POLITECNICO DI MILANO

Scuola di Ingegneria dei Sistemi
Corso di Laurea Specialistica in Ingegneria Matematica

Multiobjective Optimization For Parameter Extraction Of Power Electronics Devices

Relatore: Prof. Maurizio VERRI

Correlatore: Prof. Riccardo SACCO

Correlatore: Dr. Ivica STEVANOVIC

Correlatore: Dr. Marco BELLINI

Tesi di laurea di:

Daniele PRADA Matr. 711741

Anno Accademico 2011–2012

Daniele Prada: *Multiobjective optimization for parameter extraction of power electronics devices,* Laurea Specialistica in Ingegneria Matematica, © December 2012

PREFACE

This work stems from my internship in the Power Electronics Integration and Semiconductor groups of the Automation Devices department at ABB Switzerland Ltd, Corporate Research, Dättwil. From February 1st to October 31st , 2011, I have been working as a trainee on the subject "Parameter extraction for power electronics devices". This work was part of the research project "IGBT modeling and parameter extraction". Starting from an idea of my two ABB's supervisors, Ivica Stevanovich and Marco Bellini, I studied and developed from scratch a multi-objective optimization approach. Such an approach was so effective that a paper appeared on the subject in the "Proceedings of the IEEE Bipolar/BiCMOS Circuits and Technology Meeting, 2011", entitled "Improved Lumped Charge Model for High Voltage Power Diode and Automated Extraction Procedure" [5]. I am presently in the process of writing some further papers on this subject in collaboration with some ABB's engineers [82].

This thesis is organized as follows. Chapter 1 introduces the procedure of automatic parameter extraction for power semiconductor devices. At first, an introduction to the field of power electronics and the basic principles that govern a power diode operation are presented. Then, the different types of diode models used in circuit simulators are categorized and briefly explained. Among the whole spectrum of diode models, three are selected as a case study: Lauritzen model [68], Ma model [75] and Extended Lauritzen model [5, 82]. *The latter diode model represents an original contribution of my thesis and has been published in the paper co-authored with Marco Bellini and Ivica Stevanovic and mentioned above*. Finally, parameter extraction and refinement using a formal optimization procedure are described. At the end of this chapter, the multi-objective optimization approach for parameter extraction is briefly introduced.

Chapter 2 presents multiobjective programming and some of the nomenclature used in successive chapters. The different solution techniques for multiobjective optimization problems are classified. Some of these techniques, like the weighted sum method, are deeply described in order to show their potential drawbacks when dealing with parameter extraction problems.

In Chapter 3, evolutionary algorithms for solving multiobjective problems are discussed. I chose and implemented this particular class of solution techniques to develop a parameter extraction methodology that is potentially general enough to be applicable to all classes of models and devices. First, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is described, as it was employed to optimize a single device output characteristic at a time. Then, multi-objective evolutionary algorithms (MOEA) are extensively described. Emphasis is put on the algorithms that I implemented during the internship. Finally, some methodologies for statistical performance assessment of stochastic multiobjective optimizers are reviewed. *Material covered in this chapter is the heart of this work and represents the state of the art in the multi-objective nonlinear nonconvex optimization context. A substantial part of the described techniques and algorithms is an innovative contribution of this thesis.*

In Chapter 4 I discuss two test cases where the automated parameter extraction procedure based on MOEAs was used. Our results show that MOEAs are promising candidates to deal with parameter extraction of power semiconductor devices.

In Chapter 5 some conclusions are drawn and my view of current and future research directions is illustrated.

Appendix A provides a quick overview of the structure of the software for parameter extraction developed during the internship.

Appendix B gives the complete code of three YAML files for designing a single objective optimization, a multiobjective optimization and a MOEA comparison, in this order.

PREFAZIONE

Questa tesi è il risultato dell'esperienza di tirocinio da me svolta presso ABB Switzerland Ltd, Corporate Research, Dättwil, dal 1 Febbraio al 31 Ottobre, 2011. L'argomento del tirocinio era "estrazione parametrica per dispositivi elettronici di potenza", e faceva parte del progetto di ricerca "estrazione parametrica e modellizzazione dell'IGBT". Partendo da un suggerimento dei miei due supervisori in ABB, Ivica Stevanovich e Marco Bellini, ho studiato e sviluppato da zero un'innovativa procedura di estrazione basata su algoritmi di ottimizzazione multiobiettivo. Tale procedura si è rivelata talmente efficace che alcuni dei risultati di questo studio sono stati pubblicati su un articolo comparso in "Proceedings of the IEEE Bipolar/BiCMOS Circuits and Technology Meeting, 2011", dal titolo "Improved Lumped Charge Model for High Voltage Power Diode and Automated Extraction Procedure" [5]. Al momento, sto scrivendo un altro articolo sul medesimo argomento, in collaborazione con alcuni ingegneri di ABB [82].

La tesi è strutturata come segue. Il Capitolo 1 introduce la procedura di estrazione parametrica per modelli di dispositivi elettronici di potenza. Dopo un'introduzione generale all'elettronica di potenza, vengono richiamati i concetti fondamentali alla base del funzionamento di un diodo di potenza. Segue una breve classificazione dei diversi modelli di diodo utilizzabili in simulazioni circuitali al computer. Viene approfondito lo studio di tre particolari modelli, da me impiegati durante il tirocinio in ABB: il modello Lauritzen [68], il modello Ma [75] e il modello Lauritzen esteso [5, 82]. *L'ultimo di questi modelli rappresenta un contributo originale della mia tesi ed è stato pubblicato nell'articolo scritto in collaborazione con Marco Bellini e Ivica Stevanovic e menzionato precedentemente.* Infine, viene descritta la procedura di estrazione ed ottimizzazione parametrica per modelli di dispositivi elettronici. A conclusione del capitolo, viene brevemente introdotta la procedura di estrazione parametrica basata su algoritmi evolutivi multi-obiettivo.

Il Capitolo 2 illustra alcuni concetti chiave della programmazione multiobiettivo e una parte della notazione utilizzata nei capitoli successivi. Viene fornita una possibile classificazione degli algoritmi per la risoluzione di problemi multiobiettivo. Alcuni metodi tra i più comunemente impiegati vengono descritti a fondo per aiutare il lettore a comprendere le ragioni per cui tali metodi non sono stati da me utilizzati nella procedura di estrazione parametrica.

Il Capitolo 3 discute la classe degli algoritmi evolutivi per l'ottimizzazione multiobiettivo. La procedura di estrazione parametrica da me sviluppata si basa su questo tipo di algoritmi ed è sufficientemente generale per poter essere applicata a qualsiasi modello a parametri concentrati di un dispositivo elettronico. Inizialmente viene introdotto il metodo CMA-ES, un particolare algoritmo evolutivo utilizzato per ottimizzare una o più curve caratteristiche di un dispositivo di potenza alla volta. Segue un'ampia trattazione sugli algoritmi evolutivi multiobiettivo. Un rilievo particolare è dato ai metodi da me implementati durante il tirocinio in ABB. L'ultimo paragrafo richiama alcune tecniche statistiche per il confronto di algoritmi di ottimizzazione stocastici. *Il materiale discusso in questo capitolo rappresenta il cuore del lavoro di ricerca da me svolto e rispecchia lo stato dell'arte nel contesto dell'ottimizzazione multi-obiettivo non lineare non convessa. Una parte sostanziale delle tecniche e degli algoritmi descritti costituisce un contributo innovativo della mia tesi al problema dell'estrazione parametrica.*

Nel Capitolo 4 sono presentati due esempi di estrazione parametrica basata su algoritmi evolutivi multiobiettivo. I risultati suggeriscono che tali algoritmi consentono di raggiungere un buon accordo tra il modello di un dispositivo e i dati sperimentali.

Nel Capitolo 5 vengono tratte alcune conclusioni sull'efficacia degli algoritmi evolutivi multiobiettivo per l'estrazione parametrica e vengono proposte possibili linee di ricerca future.

L'Appendice A riporta una descrizione sintetica della struttura del software per l'estrazione parametrica sviluppato durante il tirocinio in ABB.

Nell'Appendice B è riportato il codice completo di tre file YAML impiegati nell'ambito di un'ottimizzazione a singolo obiettivo, di un'ottimizzazione multiobiettivo e di un confronto fra algoritmi evolutivi multiobiettivo.

CONTENTS

1 MODELING OF POWER SEMICONDUCTOR DEVICES 1
1.1 Overview of Power Semiconductor Devices . . . 1
1.1.1 Semiconductor Devices . . . 3
1.1.2 Power semiconductor devices . . . 4
1.2 Power Semiconductor Diode Basics . . . 8
1.2.1 Review of Basic p-n Diode Characteristics . . . 8
1.2.2 Construction and Characteristics of Power Diodes . . . 12
1.3 Why Simulate? . . . 20
1.4 Classification of Models . . . 20
1.5 The Lumped-Charge Modeling Approach . . . 22
1.5.1 Basics Concepts . . . 22
1.5.2 Diode models . . . 23
1.6 Parameter extraction . . . 38
1.6.1 Step 1: initial parameter estimation . . . 40
1.6.2 Step 2: model parameter sensitivity and parameter ranges definition . . . 45
1.6.3 Step 3: device and circuit simulation . . . 51
1.6.4 Step 4: waveform comparison . . . 54
1.6.5 Step 5: parameter optimization . . . 56

2 MULTIOBJECTIVE OPTIMIZATION 61
2.1 Problem Formulation and Solution Concepts . . . 61
2.1.1 Problems with multiple objectives . . . 61
2.1.2 Decision Space and Objective Space . . . 63
2.1.3 Notions of Optimality . . . 65
2.1.4 Orders and Cones . . . 66
2.1.5 Multiobjective optimal solutions . . . 71
2.1.6 Efficiency and Nondominance . . . 72
2.2 Properties of the Solution Sets . . . 76
2.3 Generation of the Solution Sets . . . 79
2.3.1 Scalarization methods . . . 79
2.3.2 Nonscalarizing approaches . . . 84

3 EVOLUTIONARY ALGORITHMS FOR SOLVING PARAMETER EXTRACTION PROBLEMS 91
3.1 EA Basics . . . 91
3.2 The CMA Evolution Strategy for single objective optimization . . . 94
3.2.1 Eigenvalue Decomposition of a Positive Definite Matrix . . . 95
3.2.2 The Multivariate Normal Distribution . . . 96
3.2.3 Randomized Black Box Optimization . . . 97
3.2.4 The non-elitist CMA-ES with weighted recombination . . . 98
3.2.5 A Single-Objective Elitist CMA Evolution Strategy . . . 100
3.3 Using Multi-Objective Evolutionary Algorithms . . . 102

3.3.1 Pareto Notation . . . 103
3.4 Design issues and components of Multi-Objective Evolutionary Algorithms . . . 103
3.4.1 Dominance-based ranking . . . 104
3.4.2 Diversity . . . 107
3.4.3 Elitism . . . 109
3.4.4 Constraint handling . . . 113
3.5 Structure of selected MOEAs . . . 114
3.5.1 Nondominated Sorting Genetic Algorithm-II (NSGA-II) . . . 114
3.5.2 Strength Pareto Evolutionary Algorithm 2 (SPEA2) . . . 114
3.5.3 Improved Strength Pareto Evolutionary Algorithm 2 (SPEA2+) 114
3.5.4 Pareto Archived Evolution Strategy (PAES) . . . 116
3.5.5 Covariance Matrix Adaptation for Multi-objective Optimization (MO-CMA-ES) . . . 116
3.6 Many-Objective Optimization Basics . . . 120
3.7 MOEA Performance Assessment . . . 129
3.7.1 Outperformance . . . 129
3.7.2 Stochasticity . . . 131
3.7.3 Sample Transformations . . . 132
3.7.4 Statistical Testing . . . 139

4 EXPERIMENTS 147
4.1 Problem Statement And Preliminary Studies . . . 148
4.2 Algorithmic Alternatives and Computing Environment . . . 152
4.3 First test case . . . 153
4.3.1 Choosing MOEA parameters and quality measures . . . 153
4.3.2 Design of the experiment . . . 154
4.3.3 Analysis of the results . . . 156
4.4 Second Test Case . . . 168
4.4.1 Choosing MOEA parameters and quality measures . . . 170
4.4.2 Design of the experiment . . . 170
4.4.3 Analysis of the results . . . 170

5 CONCLUSIONS AND FUTURE WORK 185

A SOFTWARE STRUCTURE 187
A.1 Building a vector of variable parameters . . . 189
A.2 Computing errors between measured and simulated data . . . 189
A.2.1 The curveFamily class . . . 189
A.2.2 The errorFamily class . . . 190
A.2.3 SSQ class hierarchy . . . 191
A.3 Reading simulation files . . . 192
A.4 Running a simulation . . . 192
A.5 Exception Handling . . . 192
A.6 Fitting of a single characteristic . . . 194
A.6.1 Designing a single objective optimization . . . 194
A.6.2 Post processing of single objective optimization results . . . 202
A.7 Concurrent fitting of several characteristics . . . 202

A.7.1 Designing a multi objective optimization 203
A.7.2 Post processing of multi objective optimization results 206
A.8 Performance comparison of several MOEAs 206

B YAML FILE EXAMPLES 209

BIBLIOGRAPHY 235

LIST OF FIGURES

Figure 1 Intrinsic, n-type and p-type semiconductors. 1
Figure 2 Comparison between semiconductors adopted in consumer electronics and in power electronics. 4
Figure 3 A simple diode rectifier. 5
Figure 4 A thyristor switched on by a trigger and off at the zero cross point. 5
Figure 5 A comparison of the losses for overhead line transmission of 1200 MW AC and High-Voltage DC (HVDC). 6
Figure 6 A gate-turn-off thyristor can be switched on and off at high frequency. 7
Figure 7 Applications using Insulated-Gate Bipolar Transistor (IGBT) technology. 7
Figure 8 Space charge density, electric field and electric potential inside a p-n junction. 9
Figure 9 Junction capacitance C_d as a function of reverse junction voltage. 12
Figure 10 Diagram of power diodes. 13
Figure 11 Electric field strength in reverse biased power diodes. 14
Figure 12 Reverse bias *i-v* characteristic of a power diode. 14
Figure 13 Characteristics of a forward biased power diode. 16
Figure 14 Diode turn on characteristics. 17
Figure 15 Definitions for the turn-off parameters of a diode. 18
Figure 16 Lauritzen model, 1991. Inductive load $i(t)$ turn off switching waveform for two different inductor values. 24
Figure 17 Charge storage locations in a p-i-n diode. 25
Figure 18 Location of charge nodes in the $P^+N^-N^+$ diode structure. . . 27
Figure 19 Location of the lumped-charge nodes and device equations. . 28
Figure 20 Charge distribution in the P^+N^- region during a reverse-recovery transient. 31
Figure 21 Diode during the turn off process. 36
Figure 22 Parameter extraction procedure. 40
Figure 23 Starting estimation of parameters τ_L and τ_H. 42
Figure 24 Sensitivity study for parameter m of Extended Lauritzen model. 46
Figure 25 Sensitivity study for parameter C_{jo} of Extended Lauritzen model. 47
Figure 26 Sensitivity study for parameter v_{jo} of Extended Lauritzen model. 48
Figure 27 Reverse recovery current synchronization. 55
Figure 28 Single objective optimization approach 58
Figure 29 Parameter extraction flow chart for a diode. 60
Figure 30 Objective functions of Example 2.3. 62
Figure 31 Objective space in Example 2.1. 64

Figure 32 Objective space in Example 2.3. . . . 65
Figure 33 Nondominated points in Example 2.3. . . . 66
Figure 34 Min-max solutions of Example 2.3. . . . 67
Figure 35 Illustration of two cones. . . . 69
Figure 36 Illustration of definitions of efficient solutions. . . . 74
Figure 37 Nondominated and weakly nondominated points. . . . 75
Figure 38 Properly nondominated point $\hat{\boldsymbol{y}}$. . . . 76
Figure 39 Efficient set, ideal, and nadir point. . . . 77
Figure 40 Nondominated points of Y and $Y + \mathbb{R}^p_{\leqq}$ are the same. . . . 78
Figure 41 Connectedness of Y_N. . . . 79
Figure 42 A set $S(\boldsymbol{\lambda}, Y)$. . . . 81
Figure 43 Properly nondominated $\hat{\boldsymbol{y}} \in Y_N$. . . . 82
Figure 44 The weighted sum method fails for nonconvex problems. . . . 83
Figure 45 Optimal solutions of ε-constraint problems. . . . 83
Figure 46 Generalized EA data structures and terminology. . . . 92
Figure 47 Key EA components. . . . 93
Figure 48 Bitwise mutation. . . . 93
Figure 49 Single-point crossover. . . . 93
Figure 50 Roulette wheel selection. . . . 94
Figure 51 Ellipsoids depicting isolines of a six different normal distributions. . . . 97
Figure 52 MOEA task decomposition. . . . 103
Figure 53 Example of NSGA-II ranking. . . . 106
Figure 54 Example of SPEA2 raw rank computation. . . . 107
Figure 55 Definition of crowding distance. . . . 108
Figure 56 Grid subdivision approach used by PAES. . . . 110
Figure 57 PAES archiving and acceptance logic. . . . 112
Figure 58 Illustration of the modification of Pareto dominance. . . . 122
Figure 59 Parallel coordinate plots of three objectives. . . . 128
Figure 60 Limitations of weak Pareto dominance. . . . 130
Figure 61 Example of empirical attainment functions. . . . 132
Figure 62 Hypervolumes of different regions. . . . 134
Figure 63 Two incomparable approximation sets A and B. . . . 135
Figure 64 A plot showing five approximation sets. . . . 137
Figure 65 Attainment surface plots for the approximation sets in Figure 64. . . . 138
Figure 66 Differences in the empirical attainment functions of two optimizers. . . . 141
Figure 67 Comparison between measured and optimized simulated capacitance characteristic. . . . 150
Figure 68 MOEA performance assessment flow. . . . 157
Figure 69 First test case: comparison of hypervolume indicator values through boxplots. . . . 159
Figure 70 First test case: comparison of additive epsilon indicator values through boxplots. . . . 160
Figure 71 First test case: convergence study for the NSGA-II and the R-NSGA-II. . . . 164

Figure 72 First test case: convergence study for the MO-CMA-ES-P and the MO-CMA-ES-P-REC. 165
Figure 73 First test case: two Pareto points found by the MO-CMA-ES-P. 166
Figure 74 First test case: another two Pareto points found by the MO-CMA-ES-P. 167
Figure 75 First test case: parallel coordinate plot of an approximation set computed by MO-CMA-ES-P. 169
Figure 76 Second test case: representation of the first four approximation sets generated by the optimizers. 171
Figure 77 Second test case: 50%-attainment surfaces of the optimizers. . 172
Figure 78 Second test case: individual differences between the probabilities of attaining different goals with the chosen optimizers. 173
Figure 79 Second test case: comparison of hypervolume indicator values through boxplots. 175
Figure 80 Second test case: comparison of additive epsilon indicator values through boxplots. 176
Figure 81 Second test case: convergence study for the NSGA-II and the R-NSGA-II. 180
Figure 82 Second test case: convergence study for the MO-CMA-ES-P and the MO-CMA-ES-P-REC. 181
Figure 83 Second test case: two Pareto points found by the MO-CMA-ES-P. 182
Figure 84 Second test case: another two Pareto points found by the MO-CMA-ES-P. 183

LIST OF TABLES

Table 1 Lauritzen model, 1991. Parameter list. 27
Table 2 Ma model, 1997. Parameter list. 34
Table 3 Extended Lauritzen model, 2011. Parameter list. 39
Table 4 Criteria and alternatives in Example 2.1. 61
Table 5 Some orders on $\mathbb{R}^p$. 68
Table 6 Feasible solutions and objective values in Example 2.47. . . . 88
Table 7 Default strategy parameters of the non-elitist CMA. 100
Table 8 List of MOEAs implemented in the Python library for parameter extraction. 105
Table 9 Selected preference relations on Pareto front approximations. 129
Table 10 Extended Lauritzen model, 2011. Parameter list. 148
Table 11 Ranges for Extended Lauritzen model parameters influencing the capacitance characteristic. 150
Table 12 Ranges for Extended Lauritzen model parameters influencing dc and transient characteristics. 151
Table 13 Description of the computing environment. 153

Table 14 First test case: average rankings of the algorithms with respect to the hypervolume indicator. 158
Table 15 First test case: average rankings of the algorithms with respect to the additive epsilon indicator. 159
Table 16 First test case: adjusted p-values for the post-hoc test of the hypervolume indicator. 162
Table 17 First test case: adjusted p-values for the post-hoc test of the additive epsilon indicator. 163
Table 18 A principal component analysis of the reference set for the first test case. 168
Table 19 Second test case: seeds provided to the experimental design procedure. 172
Table 20 Second test case: average rankings of the algorithms with respect to the hypervolume indicator. 174
Table 21 Second test case: average rankings of the algorithms with respect to the additive epsilon indicator. 175
Table 22 Second test case: adjusted p-values for the post-hoc test of the hypervolume indicator. 178
Table 23 Second test case: adjusted p-values for the post-hoc test of the additive epsilon indicator. 179

LISTINGS

Listing 1 A YAML file for optimization of diode reverse recovery waveform. . . . 209
Listing 2 A YAML file for multi-objective optimization with the NSGA-II/R-NSGA-II. . . . 215
Listing 3 A YAML file for designing a MOEA performance comparison. 231

LIST OF ALGORITHMS

2.1 Lexicographic Optimization . . . 85

3.1 Stochastic black box search. . . . 98
3.2 $(\mu/\mu_w, \lambda)$-CMA-ES . . . 99
3.3 $(1+\lambda)$-CMA-ES . . . 101
3.4 updateStepSize$(a = [\boldsymbol{x}, \bar{p}_{\text{succ}}, \sigma, \boldsymbol{p}_c, \mathbf{C}], p_{\text{succ}})$ . . . 101
3.5 updateCovariance$(a = [\boldsymbol{x}, \bar{p}_{\text{succ}}, \sigma, \boldsymbol{p}_c, \mathbf{C}], \boldsymbol{x}_{\text{step}} \in \mathbb{R}^n)$ . . . 102
3.6 NSGA-II algorithm . . . 115
3.7 SPEA2 algorithm . . . 116
3.8 SPEA2+ algorithm . . . 117
3.9 PAES algorithm . . . 117
3.10 $\mu \times (1+\lambda)$-MO-CMA-ES . . . 119
3.11 Principal Component Analysis for Multi-Objective Optimization . . . 126

4.1 Generation of initial populations for a MOEA comparison. . . . 155
4.2 Generate random seeds for MOEA runs. . . . 156

ACRONYMS

EMC Electro-Magnetic Compatibility

EMI Electro-Magnetic Interference

HVDC High-Voltage DC

FACTS Flexible AC Transmission Systems

GTO Gate Turn-Off Thyristor

IGCT Integrated Gate-Commutated Thyristor

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor

IGBT Insulated-Gate Bipolar Transistor

BiMOS Bipolar-Metal-Oxide-Semiconductor

PCB Printed Circuit Board

TCAD Technology Computer-Aided Design

SCR Silicon Controlled Rectifier

MCT MOS Controlled Thyristor

BJT Bipolar Junction Transistor

HDL Hardware Description Language

KCL Kirchoff current laws

KVL Kirchoff voltage laws

SPICE Simulation Program with Integrated Circuit Emphasis

CMA-ES Covariance Matrix Adaptation Evolution Strategy

RSS Residual Sum of Squares

RSM Residual Sum of Magnitudes

MOP Multiobjective Optimization Problem

EA Evolutionary Algorithm

EC Evolutionary Computation

GA Genetic Algorithm

ES Evolution Strategy

EP Evolutionary Programming

EVOP Evolutionary Operator

MOEA Multi Objective Evolutionary Algorithm

NFL No Free Lunch

NSGA-II Non Dominated Sorting Genetic Algorithm II

PAES Pareto Archived Evolution Strategy

MO-CMA-ES Multi-Objective Covariance Matrix Adaption Evolution Strategy

SPEA2 Strength Pareto Evolutionary Algorithm 2

SPEA2+ Improved Strength Pareto Evolutionary Algorithm 2

R-NSGA-II Reference point based Non Dominated Sorting Genetic Algorithm II

PCA Principal Component Analysis

SOP Single-objective Optimization Problem

MCDA Multi Criteria Decision Aid

EAF Empirical Attainment Function

ECDF Empirical Cumulative Distribution Function

KS Kolmogorov-Smirnov

APV adjusted p-value

SBX Simulated Binary Crossover

API Application Programming Interface

PISA Platform and Programming Language Independent Interface for Search Algorithms

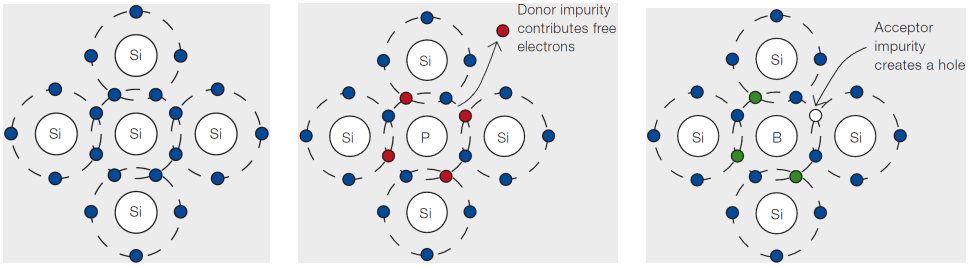
1 MODELING OF POWER SEMICONDUCTOR DEVICES

The aim of this chapter is to introduce the procedure of automatic parameter extraction for power semiconductor devices. Section 1.1 presents an introduction to the field of power electronics. Then, in Section 1.2, we focus our attention on the basic physical principles that govern a power diode operation, because this is the device we studied most extensively during our internship at ABB Corporate Research Center in Switzerland. Section 1.3 briefly discusses reasons for device and circuit simulation in power electronic system design. The simulation activity starts with the choice of a suitable model for the device under test. In Section 1.4, the different types of diode models are categorized. Section 1.5 describes three diode models that we deeply studied. The extended Lauritzen diode model is the most recent model and it is used in Chapter 4 to provide an example of automatic parameter extraction. Finally, Section 1.6 describes parameter extraction and refinement using a formal optimization procedure.

1.1 OVERVIEW OF POWER SEMICONDUCTOR DEVICES

Power electronics deals with the conversion and control of electricity using solid-state electronic switches, i.e., circuits or devices built entirely from solid materials without moving mechanical parts [77]. Solid-state electronics stems from the discovery of semiconductors.

Semiconductors, such as silicon, have electrical properties that fall somewhere between a good conductor (e.g., copper) and an insulator (e.g., rubber). If placed in a circuit they act for much of the time as insulators, forming a barrier to the flow of electrons, but sometimes, under certain conditions (elevated temperature, exposure to electromagnetic fields, etc.), they behave more like conductors, allowing electrons to flow freely.



(a) I-type semiconductor, no impurities added.

(b) N-type semiconductor, phosphorus added as impurity.

(c) P-type semiconductor, boron added as impurity.

Figure 1: Schematic diagram showing only the valence electron shell to illustrate intrinsic, p-type and n-type semiconductors [44].

The conductivity of a pure semiconductor, often called an *intrinsic* or *I-type* semiconductor (see Figure 1a), can be drastically changed by adding other elements, known as impurities, so that a new and different crystal is formed in a process called "doping". Dopants used for silicon-based semiconductors have either a three- or five-electron valency, which is one less or one more than silicon's four.

By adding small quantities of phosphorus, for example, with a valency of five, the properties of an I-type semiconductor are altered so that more free electrons are introduced, since its fifth electron remains unpaired. This creates an excess of negative electron charge carriers, leading to the creation of an *n-type* crystal (see Figure 1b). These weakly-bound electrons can move about in the crystal lattice relatively freely and can facilitate conduction in the presence of an electric field. Similarly, by adding small quantities of boron, with a valency of three, the properties of an I-type semiconductor are altered again. This time, however, it is the fourth electron of silicon that remains unsaturated when it covalently bonds with the dopant boron. Unsaturated bonds are repaired by electrons from neighboring bonds, leaving positive "holes" or *p-type* regions in the semiconductor (see Figure 1c). The continued process of repair creates a chain-like reaction that results in positively charged holes moving around the crystal. Current can be carried either by the flow of negatively charged electrons or by the flow of positively-charged "holes" in the crystal lattice of the semiconductor material. Both n- and p-type semiconductors behave like insulators below a threshold voltage, resisting current flow, but above that threshold they behave like conductors, allowing the current to flow freely. The conductivity of these n-type or p-type semiconductors can be varied between insulating and conducting by the level of dopant incorporated into the silicon lattice. To control the direction and magnitude of the current required to switch the semiconductor from an insulator to a conductor, p- and n-type semiconductors can be arranged adjacently in the same crystal, forming a junction in which the negatively charged electrons from the n-type semiconductor fill the holes resulting from unsaturated pairing in the p-type semiconductor. This creates a thin nonconductive I-type semiconductor junction at the border between more conductive p- and n-type semiconductors. This non-conductive barrier must be overcome by an external voltage source to make the junction conduct. By manipulating this non-conductive p-n junction, the electrical properties of the device can be controlled. The property and arrangement of such doped semiconductors provides the key element that led to the development of the transistor, and now forms the fundamental building block of all modern solid-state electronic devices.

In recent years advances in power semiconductor technology have produced an ever expanding array of applications. The adverse effects of global warming, resulting from the burning of fossil fuels, have played a major role in driving the increased use of power semiconductor technologies aimed at utilizing renewable energy generation and increasing energy efficiency.

Even in the very early days of electricity, transmission efficiency had an impact on the type of electricity that prevailed, i.e., direct current (DC) or alternating current (AC). Initially, for historic reasons, electric power systems were predominantly DC circuits. However, the inability to alter DC voltage levels, at that time, limited its use. Power generators were built therefore to satisfy the load on the circuit (e.g.,

at a voltage level required for lighting or motors). Inefficient transmission at such low voltages meant that these generators had to remain within a short distance of consumers.

The subsequent development of AC generators and transformers provided the much needed technology that would allow power to be stepped up to 110 kV or more, to facilitate efficient long-distance power transmission. This meant that power generators need not remain close to their end users, nor did their voltage levels need to match the class of load attached to their circuits (since step-down transformers could be used to alter the voltage to suit the load). These early developments in technology played a pivotal role in determining the nature and architecture of power transmission and distribution systems.

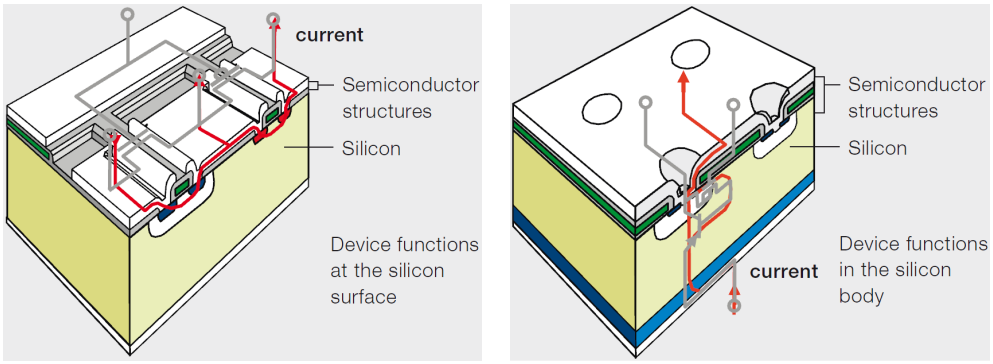
Today new demands have been placed on electric power systems, including greater energy efficiency and sustainability, yet developments in technology remain a major influence on their evolution. In the last few decades, developments in semiconductor technology have had a major impact on the architecture of the power systems that operate around the world. Innovations that have been made possible through such technology include the efficient bulk transmission of electric energy in the form of High-Voltage DC (HVDC) [99], the introduction of energy-saving variable-speed drives [100], the conversion of AC at one frequency to AC at another (50/60 Hz or 50/16.6 Hz) through the use of frequency converters, and the introduction of Flexible AC Transmission Systems (FACTS) to enhance control and increase the power transfer capability of the network [98].

1.1.1 *Semiconductor Devices*

Today the vast majority of semiconductor devices are used in the consumer electronics industry. These products include computers, DVD players, cell phones, household appliances and video games. These types of products generally operate in the nanowatt to milliwatt range. The miniaturization of such devices continues to develop with ever increasing complexity so that today's integrated circuits, known as microchips, contain hundreds of millions of switches operating at the nanowatt level. The function of these devices is typically achieved by structuring the surface area of the semiconductor material, see Figure 2a.

In addition, many low-power semiconductors are used today to modify the form of electrical energy (i.e., to modify its voltage or frequency), including:

- DC/DC converters found in most mobile devices (eg, mobile phone, mp3 player). They maintain the voltage at a fixed value, whatever the charge level of the battery.
- AC/DC converters (rectifiers) used whenever an electronic device is connected to the mains (eg, computers, televisions, game consoles).
- AC/AC converters used to change either the voltage level or the frequency. These are found in international power adapters, light dimmer switches, etc.
- DC/AC converters (inverters) used, e.g., to supply power to AC devices in a car from a DC battery.



(a) Semiconductors found in consumer electronics.

(b) Semiconductors found in power electronics.

Figure 2: Comparison between semiconductors adopted in consumer electronics and in power electronics [44].

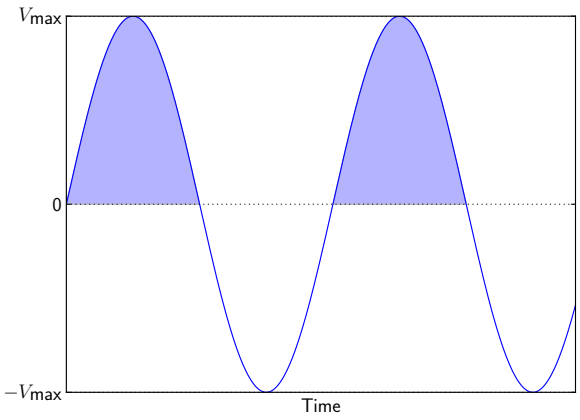
Today, similar semiconductor devices can be used to modify electric energy in the megawatt power range. They are generally silicon-based and the functionality to either block or conduct current involves the whole 3-D body of the semiconductor (see Figure 2b). Generally these devices are less visible to end users than their miniaturized cousins in the consumer electronics industry, yet they modify voltage and frequency in much the same way, only on an industrial scale, forming robust high-power switches that are either "on" or "off".

Although power electronics form a relatively small segment in the semiconductor market, rapid growth in demand for high-power semiconductor devices in the last five years has seen significant increases as new applications for this technology are recognized.

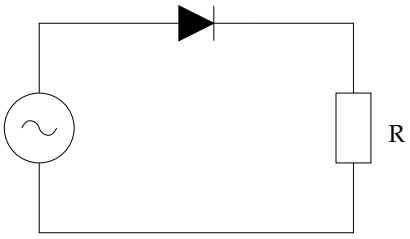
1.1.2 *Power semiconductor devices*

Power semiconductor devices first appeared in the early 1950s, e.g., with the 7 kW semiconductor diode. This device maintains the flow of electric current in one direction (called the diode's forward direction), while blocking the flow in the opposite direction (see Figure 3). Semiconductor diodes provided the first solid-state rectifiers.

In the late 1950s a new bipolar semiconductor, known as a thyristor, was developed. Thyristors are similar to diodes in that they block electric current flow in the reverse direction, but they also prevent current flow in the forward direction unless triggered to do so. In this way, the power (or current) that is supplied to a load could be controlled by triggering conductance at a particular phase of the waveform. Once switched on, the thyristor remains "on", switching "off" once per cycle when the current drops to the next zero cross point (see Figure 4). Once switched on, the thyristor behaves essentially like a diode. Since thyristors can switch power at the MW level they can be used to convert AC to DC and DC to AC for HVDC transmission. Today HVDC classic systems (with thyristors connected in series) are capable of carrying 6400 MW of power over several thousand kilometers, providing efficient methods to transport electrical energy from remote

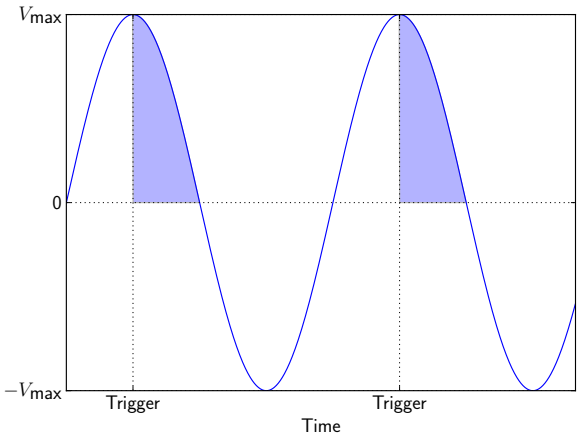


(a)

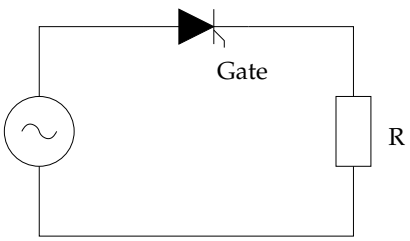


(b)

Figure 3: A simple diode rectifier.



(a)



(b)

Figure 4: A thyristor switched on by a trigger and off at the zero cross point.

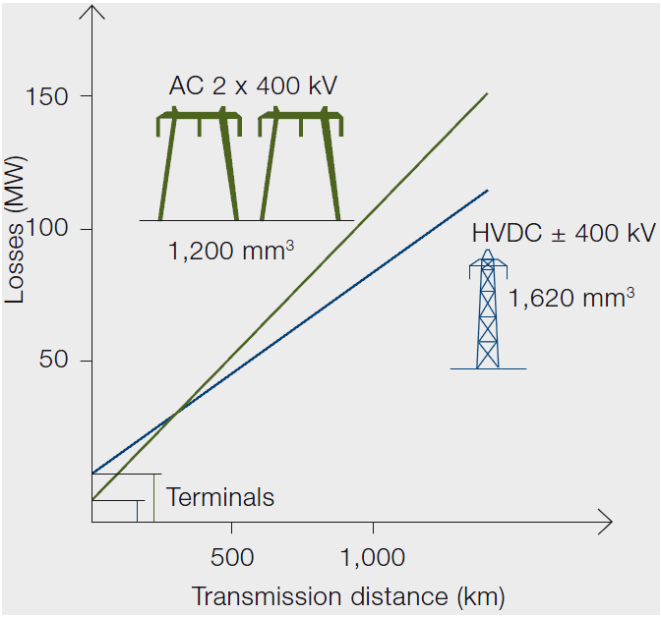


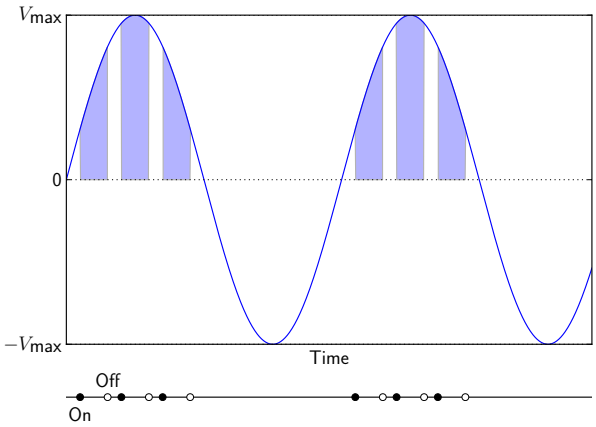
Figure 5: A comparison of the losses for overhead line transmission of 1200 MW AC and High-Voltage DC (HVDC) [44].

sources of generation to busy population centers. An HVDC transmission line has lower losses than optimized AC lines for the same power capacity. The losses in the converter stations have of course to be added, but since they are only about 0.7 percent of the transmitted power in each station, the total HVDC transmission losses come out lower than the AC losses for distances above a certain threshold (e.g., around 500 km for overhead lines, see Figure 5). In addition, HVDC is the only practical solution for subsea cable connections over 70 km.

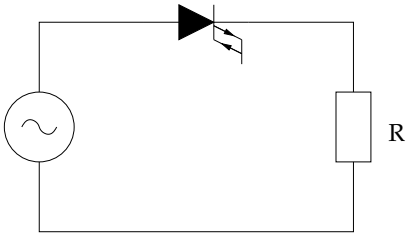
Although thyristors assembled in series can function in the several thousand MW range, a similar single thyristor can be used in the 10 MW range to modify the supply of voltage and current through a medium-voltage drive to efficiently control the speed of an industrial motor. Applications driven by electric motors account for an estimated 65 percent of all industrial energy use; however a significant portion of this energy is currently lost through the wasteful methods used to control their speeds. By altering the voltage and frequency using power electronics, the speed of an AC motor can be adjusted with much lower losses. Typical applications using variable-speed drives can reduce energy consumption by 30 to 50 percent.

Further developments in semiconductor technology have resulted in the Gate Turn-Off Thyristor (GTO), which can be switched off at an arbitrary point in the waveform, providing greater control over the power output, see Figure 6. Such devices are common in frequency converters used to alter the power frequency of the domestic grid to suit the power frequency used by electric trains and metros.

Not long after the development of the GTO, an improved type of device known as the Integrated Gate-Commutated Thyristor (IGCT) was developed. These devices, like the GTO, can be switched "on" or "off", but since their turnoff times are much faster, they can operate at much higher frequencies than GTOs. They are able to



(a)



(b)

Figure 6: A gate-turn-off thyristor can be switched on and off at high frequency.

(a) The HVDC Light converter on the BorWin alpha platform with some wind turbines.

(b) Variable speed drives.

Figure 7: Applications using Insulated-Gate Bipolar Transistor (IGBT) technology [44].

cope with high rates of voltage rise and have lower conduction losses. Today there are many thousands of drives worldwide using IGCTs.

Two decades ago, a seemingly simple variant of the Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) began to change the power electronic landscape with the creation of the Insulated-Gate Bipolar Transistor (IGBT). The IGBT is noted for high efficiency and fast switching (switching "on" and "off" multiple times per cycle) and relies on Bipolar-Metal-Oxide-Semiconductor (BiMOS) technology. These devices can be assembled in a variety of ways to modify the voltage or frequency of electrical power for a range of applications from HVDC Light® power transmission systems (see Figure 7a) to low-voltage variable-speed drives (see Figure 7b). Both variable speed drives and HVDC Light require rectifiers and converter topology. However, as with all applications, the way in which semiconductor devices are assembled in these applications determines the power rating at which they can operate.

The different types of semiconductor devices and the way they are assembled, define their suitability for a particular application. Each device is packaged, not only to maintain its integrity and performance, but also to ensure its safe operation and longevity when working in harsh environments. Power semiconductors are a key element in an increasing number of electrical applications. They allow drives to efficiently operate motors from 10 W to several hundred MW. They enable electrical energy up to 6 GW to be transmitted through HVDC lines at 800 kV. They provide the capacity for trains, cranes and elevators to run smoothly and allow renewable energy sources, such as wind turbines and large hydropower plants to connect to the grid. Even radar systems emitting high-power pulses depend on power semiconductors to securely operate air traffic.

In the next section we go into more depth of power semiconductor diode, since we will often refer to it throughout this work.

1.2 POWER SEMICONDUCTOR DIODE BASICS

Power semiconductor diodes are required to carry up to several kiloampers of current under forward bias condition and block up to several kilovolts under reverse biased condition. These extreme requirements call for important structural changes in a power diode with respect to a p-n junction diode.

1.2.1 *Review of Basic p-n Diode Characteristics*

A p-n junction diode is formed by placing p and n type semiconductor materials in intimate contact on an atomic scale. In an open circuit p-n junction diode, majority carriers from either side will defuse across the junction to the opposite side where they are in minority. These diffusing carriers will leave behind a region of ionized atoms at the immediate vicinity of the metallurgical junction. This region of immobile ionized atoms is called the *space charge region*. This process continues till the resultant electric field (created by the space charge density) and the potential barrier at the junction builds up to sufficient level to prevent any further migration of carriers. At this point the p-n junction is said to be in *thermal equilibrium condition*. In this case, the potential barrier at the junction is frequently referred to as

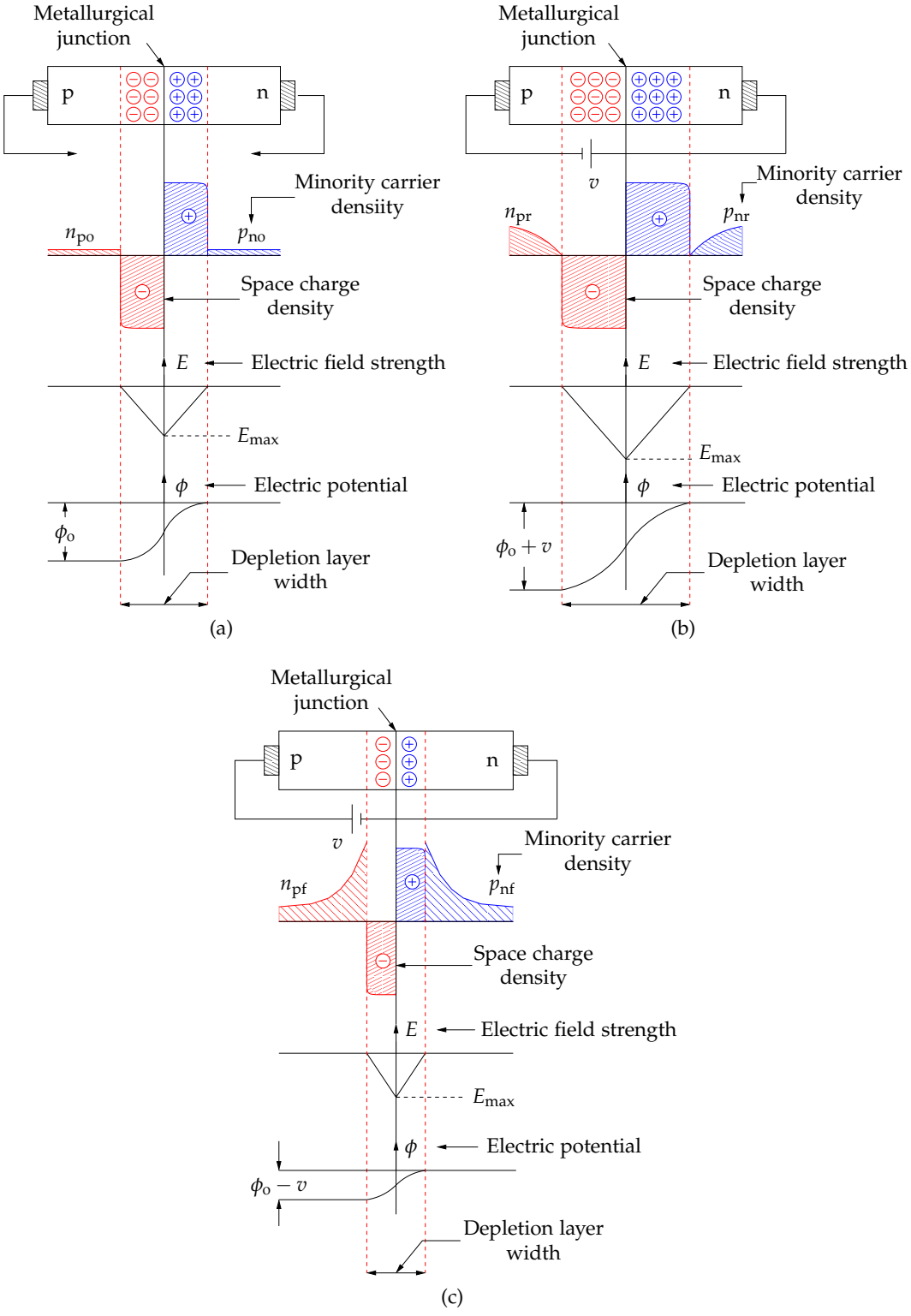


Figure 8: Space charge density, electric field and electric potential inside a p-n junction under (a) thermal equilibrium condition, (b) reverse biased condition, (c) forward biased condition.

the *built-in potential*. Variation of the space charge density, the electric field and the potential along the device are shown in Figure 8a.

When an external voltage is applied with the p side more negative than the n side, the junction is said to be under *reverse bias condition*. This reverse bias adds to the height of the potential barrier. The electric field strength at the junction and the width of the space charge region (also called the *depletion region* because of the absence of free carriers) also increases. On the other hand, free minority carrier densities will be zero at the edge of the depletion region on either side (Figure 8b).

This gradient in minority carrier density causes a small flux of minority carriers to defuse towards the depletion layer where they are swept immediately by the strong electric field into the electrical neutral region of the opposite side. This will constitute a small leakage current across the junction from the n side to the p side. There will also be a contribution to the leakage current by the electron-hole pairs generated in the space charge layer by the thermal ionization process. These two components of current together are called the *reverse saturation current* I_s of the diode. The value of I_s is independent of the reverse voltage magnitude (up to a certain level) but extremely sensitive to temperature variation.

When the applied reverse voltage exceeds some threshold value (for a given diode) the reverse current increases rapidly. The diode is said to have undergone *reverse break down*.

Reverse break down is caused by "impact ionization" as explained below. Electrons accelerated by the large depletion layer electric field due to the applied reverse voltage may attain sufficient kinetic energy to liberate another electron from the covalent bonds when it strikes a silicon atom. The liberated electron in turn may repeat the process. This cascading effect (avalanche) may produce a large number of free electrons very quickly resulting in a large reverse current. The power dissipated in the device increases and may cause its destruction. Therefore, a diode should prevented from operating in the reverse breakdown region.

When the diode is forward biased (i.e., p side more positive than n side) the potential barrier is lowered and minority carriers are injected to both sides of the junction (see Figure 8c). Two situations are relevant:

- The change in majority-carrier concentration due to the injected minority carriers is so small in the regions outside the depletion region that its effect can be neglected. Thus, there is no electric field in the n^- region if current flows, and injected carriers move only by diffusion. This condition is referred to as *low-level injection*.

- At *high-level injection*, on the contrary, there is significant alteration of the majority-carrier concentration outside the depletion region, which give rise to an electric field. Thus carrier motion is influenced by both diffusion and drift in this region. The presence of the electric field results in a voltage drop across this region. Hence only a fraction of the applied voltage drops across the junction.

The injected minority carriers eventually recombine with the majority carriers as they defuse further into the electrically neutral drift region. The excess free carrier density in both p and n side follows exponential decay characteristics. The characteristic decay length is called the "minority carrier diffusion length".

Carrier density gradients on either side of the junction are supported by a forward current i which can be expressed as

$$i(v) = I_s\left(\exp\left(\frac{qv}{kT}\right) - 1\right) \tag{1.1}$$

where

- I_s is the reverse saturation current;
- v is the voltage drop across the p-n junction;
- q is the elementary charge;
- k is the Boltzmann constant;
- T is the temperature in kelvin.

Now let us consider the charge-storage effects of the device. The importance of this effect is clear. If there were no charge storage, the device would be infinitely fast; that is, there would be no charge inertia, and currents could be changed in zero time. Instead, diodes take a finite time to make transition from reverse bias to forward bias condition (switch on) and vice versa (switch off). The two forms of charge storage are the minority-carrier injection Q_s and the space charge Q_d.

- Stored charge Q_s. The electrons injected into the p side and the holes injected into the n side of the diode not only generate the current i but also represent the charge Q_{sn} and Q_{sp} stored in the diode.

  In [1], for an abrupt p^+n junction, where dopant concentration in the anode is greater by some order of magnitude of dopant concentration in the cathode, it is shown that the following relationship holds

  $$Q_s = \tau_D \cdot i(v) \tag{1.2}$$

  where $i(v)$ is given by (1.1). The proportionality constant τ_D between the current and the charge is a time constant which represents the minimum time required to either store or remove the charge; it is called the *transit time* of the diode.

- Space charge Q_d. A change in v requires a variation in the width of the space charge region d (see Figure 8). A change in d implies a change in the charge associated with the charge layer. Thus, this charge must be moved into or out of the space-charge region to balance a change in the junction voltage. This means that a capacitance must be associated with the space-charge region.

  The equivalent differential capacitance associated with a change $\mathrm{d}v$ of the applied voltage v is then

  $$C_d = \frac{\mathrm{d}Q_d}{\mathrm{d}v} \tag{1.3}$$

  In [1] it is shown that, for an abrupt p-n junction, the following relationship holds

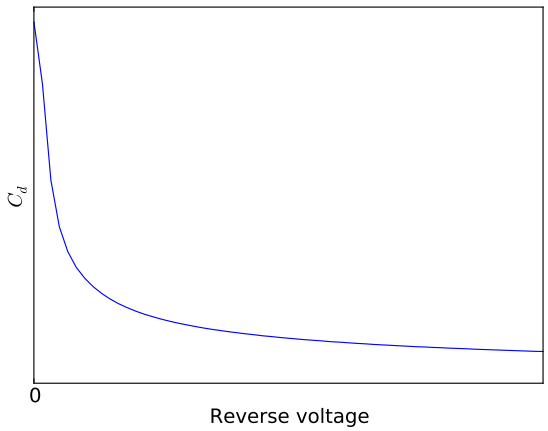


Figure 9: Junction capacitance C_d as a function of reverse junction voltage.

$$C_d = \frac{C_d(0)}{\sqrt{1 - \frac{v}{v_j}}} \tag{1.4}$$

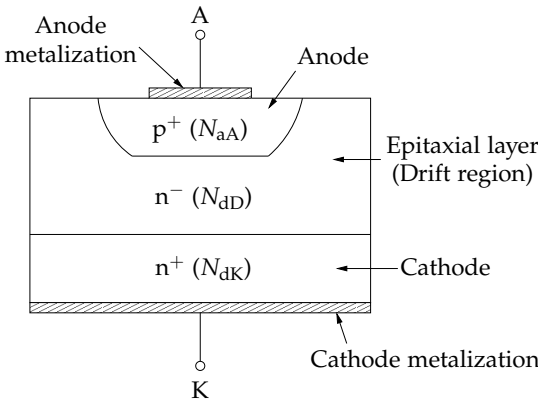
where $C_d(0)$ is the zero bias capacitance and v_j is the built-in voltage.

For reverse bias and small forward bias, the injected charge Q_s dominates; hence the effects of Q_d become negligible. This second observation is extremely important, since as v approaches v_j, the space charge-capacitance (see (1.4)) becomes infinite. In this case, (1.4) is no longer valid.

Figure 9 shows a plot of the junction capacitance as a function of reverse junction voltage.

1.2.2 *Construction and Characteristics of Power Diodes*

As mentioned before, power diodes are required to conduct several kiloampers of current in the forward direction with very little power loss while blocking several kilovolts in the reverse direction. Large blocking voltage requires wide depletion layer in order to restrict the maximum electric field strength below the "impact ionization" level. Space charge density in the depletion layer should also be low in order to yield a wide depletion layer for a given maximum electric field strength. These two requirements will be satisfied in a lightly doped p-n junction diode of sufficient width to accommodate the required depletion layer. Such a construction, however, will result in a device with high resistivity in the forward direction. Consequently, the power loss at the required rated current will be unacceptably high. On the other hand if forward resistance (and hence power loss) is reduced by increasing the doping level, reverse break down voltage will reduce. This apparent contradiction in the requirements of a power diode is resolved by introducing a lightly doped n^- "drift layer" of required thickness between two heavily doped p^+ and n^+ layers as shown in Figure 10a. Figure 10b shows a photograph of some fast power diodes.



(a) Schematic cross section.

(b) Photograph of fast power diodes [44].

Figure 10: Diagram of power diodes.

To arrive at the structure shown in Figure 10a a lightly doped n^- epitaxial layer of specified width (depending on the required break down voltage) and donor atom density (N_{dD}) is grown on a heavily doped n^+ substrate (with donor atom density N_{dK}) which acts as the cathode. Finally the p-n junction is formed by defusing a heavily doped (with acceptor atom density N_{aA}) p^+ region into the epitaxial layer. This p type region acts as the anode.

Impurity atom densities in the heavily doped cathode (N_{dK}) and anode (N_{aA}) are about of the same order of magnitude ($10^{19}\,C\,m^{-3}$) while that of the epitaxial layer (also called the drift region) is lower by several orders of magnitude ($N_{dD} \approx 10^{14}\,C\,m^{-3}$).

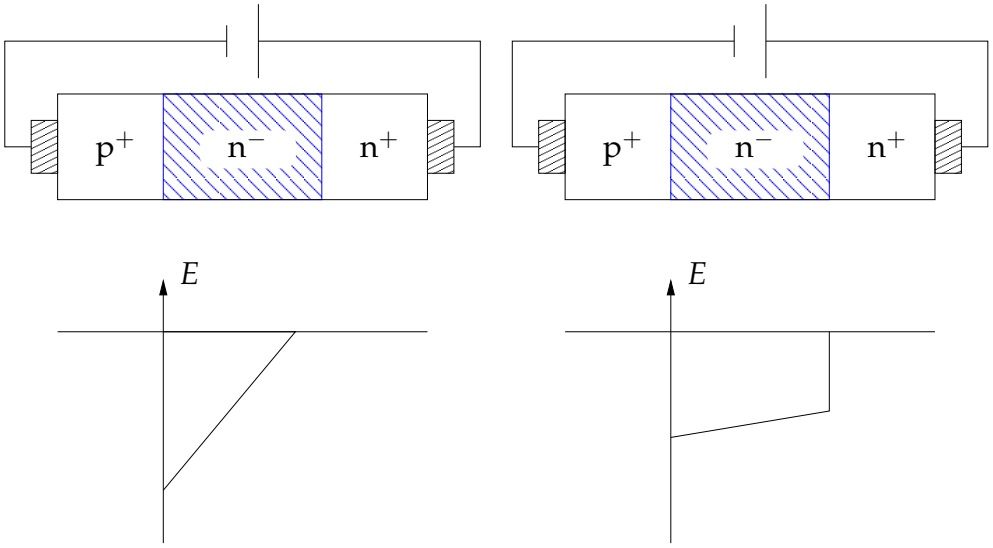
1.2.2.1 *Power Diode under Reverse Bias Condition*

As in the case of a p-n diode the applied reverse voltage is supported by the depletion layer formed at the p^+-n^- metallurgical junction. Overall neutrality of the space charge region dictates that the number of ionized atoms in the p^+ region should be same as that in the n^- region. However, since $N_{dD} \ll N_{aA}$, the space charge region almost exclusively extends into the n^- drift region. Now the physical width of the drift region d can be either larger or smaller than the depletion layer width at the break down voltage. Consequently two type of diodes exist:

- "non punch through" type;
- "punch through" type.

In "non punch through" diodes the depletion layer boundary doesn't reach the end of the drift layer. On the other hand in "punch through" diodes the depletion layer spans the entire drift region and is in contact with the n^+ cathode. However, due to very large doping density of the cathode, penetration of drift region inside cathode is negligible. Electric field strength inside the drift region of both these types of diode at break down voltage is shown in Figure 11.

In non punch through type diodes the electric field strength is maximum at the p^+-n^- junction and decrease to zero at the end of the depletion region. Where as, in the punch through construction the field strength is more uniform.



(a) Non punch through type. (b) Punch through type.

Figure 11: Electric field strength in reverse biased power diodes.

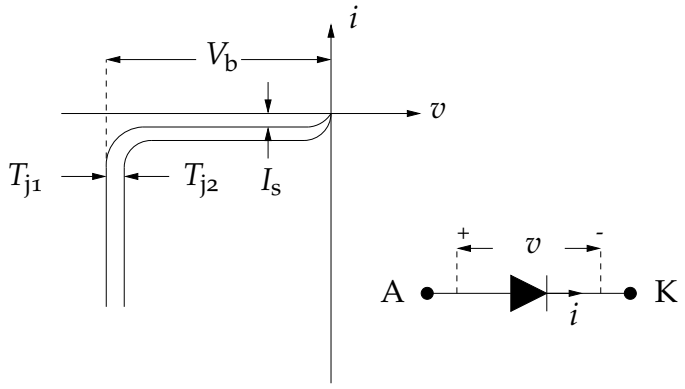


Figure 12: Reverse bias *i*-*v* characteristic of a power diode. v is the voltage drop across the p^+n^- junction, V_b is the reverse breakdown voltage, I_s is the reverse saturation current and T_j is the junction temperature ($T_{j2} > T_{j1}$).

Under reverse bias condition only a small leakage current flows in the reverse direction (i.e. from cathode to anode). This reverse current is independent of the applied reverse voltage but highly sensitive to junction temperature variation. When the applied reverse voltage reaches the break down voltage, reverse current increases very rapidly due to impact ionization and consequent avalanche multiplication process. Voltage across the device dose not increase any further while the reverse current is limited by the external circuit. Excessive power loss and consequent increase in the junction temperature due to continued operation in the reverse brake down region quickly destroys the diode. Therefore, continued operation in the reverse break down region should be avoided. A typical *I*-*V* characteristic of a power diode under reverse bias condition is shown in Figure 12.

1.2.2.2 *Power Diode under Forward Bias Condition*

As the metallurgical p^+n^- junction becomes forward biased there will be injection of excess p type carrier into the n^- side. At low level injections all excess p type carriers recombine with n type carriers in the n^- drift region. At high level of injection the excess p type carrier density distribution reaches the n^-n^+ junction and attracts electron from the n^+ cathode. This leads to electron injection into the drift region across the n^-n^+ junction. This mechanism is called "double injection".

Excess p and n type carriers recombine inside the drift region. If the width of the drift region is less than the diffusion length of carries the spatial distribution of excess carrier density in the drift region will be fairly flat and several orders of magnitude higher than the thermal equilibrium carrier density of this region. Conductivity of the drift region will be greatly enhanced as a consequence (also called "conductivity modulation").

At large forward bias, the diode current deviates from the ideal exponential characteristic given in (1.1). This deviation is due to the presence of an ohmic resistance in the diode as well as the effects of high-level injection. The effects of both ohmic resistance and high-level injection can be modeled by the ohmic resistance R_s. The value of R_s is determined by the amount the actual diode voltage deviates from the ideal exponential characteristic at a specified current. Then, if v_{AK} is the voltage applied to the diode and v is the drop across the p^+n^- junction, it can be written that

$$v = v_{AK} - R_s \cdot i. \tag{1.5}$$

Characteristics of a forward biased power diode are shown in Figure 13.

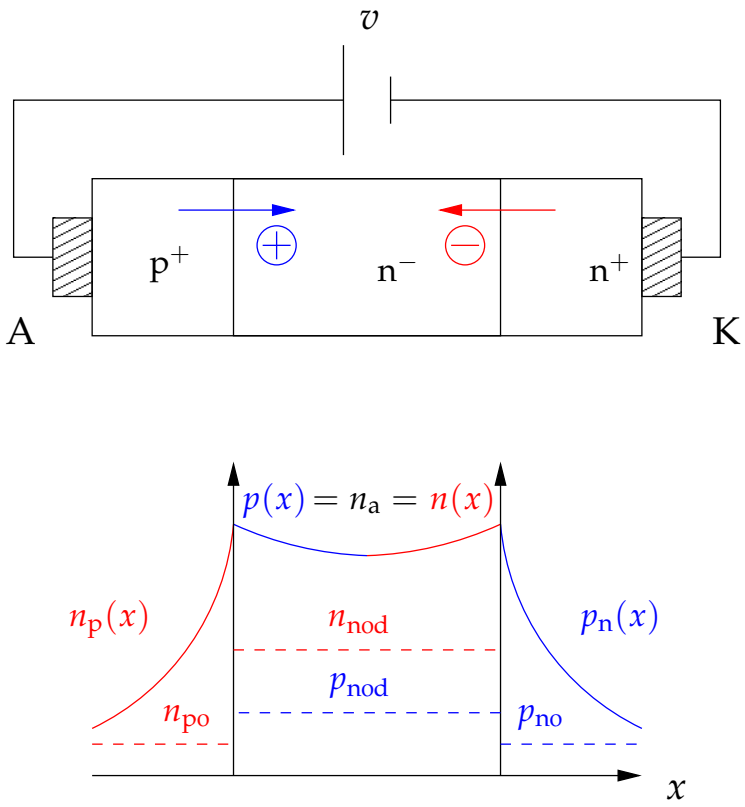
1.2.2.3 *Switching Characteristics of Power Diodes*

Power diodes take a finite time to make transition from reverse bias to forward bias condition (switch on) and vice versa (switch off).

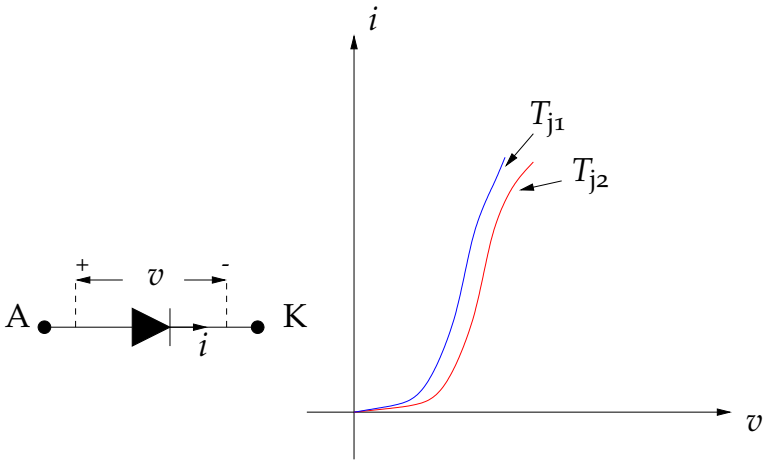
Behavior of the diode current and voltage during these switching periods are important due to the following reasons:

- Severe over voltage / over current may be caused by a diode switching at different points in the circuit using the diode.
- Voltage and current exist simultaneously during switching operation of a diode. Therefore, every switching of the diode is associated with some energy loss. At high switching frequency this may contribute significantly to the overall power loss in the diode.

Diodes can be subdivided into two main classes: Rectifier Diodes (Standard Recovery) and Fast Diodes (see 10b). Rectifier Diodes are generally used for conversion of AC (alternating current) to DC (direct current). While optimized for low conduction losses, Rectifier Diodes withstand only moderate dynamic stress in transition from conducting to the blocking state. Fast Diodes, on the other hand, are companion devices to switches in DC to AC conversion. Every switch (GTO, IGCT or IGBT) requires a complementary diode (e.g. for "free-wheeling" reactive



(a) Excess free carrier density distribution.



(b) *i*-*v* characteristic.

Figure 13: Characteristics of a forward biased power diode. T_j is the junction temperature ($T_{j2} < T_{j1}$).

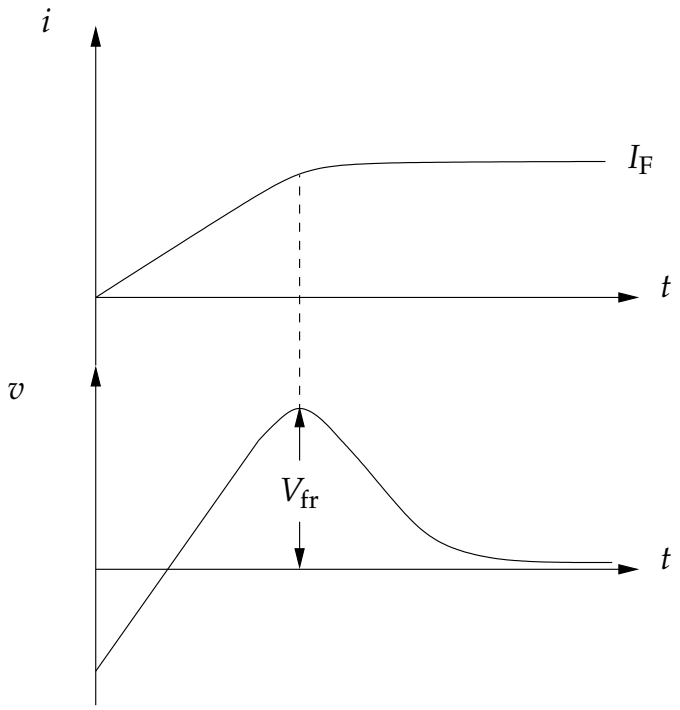


Figure 14: Forward current and voltage waveforms of a power diode during turn on operation.

power) in order to enable operation of the DC-AC conversion system with inductive loads. Fast Diodes are optimized to accept high dynamic stress (fast transition from conducting to blocking state). However, they generally have higher conduction losses than Rectifier Diodes.

A typical turn on transient is shown in Figure 14.

It is observed that the forward diode voltage during turn on may transiently reach a significantly higher value V_{fr} compared to the steady state voltage drop at the steady current I_F.

In some power converter circuits (e.g. voltage source inverter) where a free wheeling diode is used across an asymmetrical blocking power switch (i.e. asymmetric IGCT) this transient over voltage may be high enough to destroy the main power switch.

Figure 15 shows a typical turn off behavior of a power diode.

Salient features of this operation mode are:

- The diode current does not stop at zero, instead it grows in the negative direction to I_{RM} called "maximum reverse recovery current", which can be comparable to I_F. In many power electronic circuits (e.g. choppers, inverters) this reverse current flows through the main power switch in addition to the load current. Therefore, this reverse recovery current has to be accounted for while selecting the main switch.

- Voltage drop across the diode does not change appreciably from its steady state value till the diode current reaches reverse recovery level. In many power electric circuits (choppers, inverters) this may create an effective short

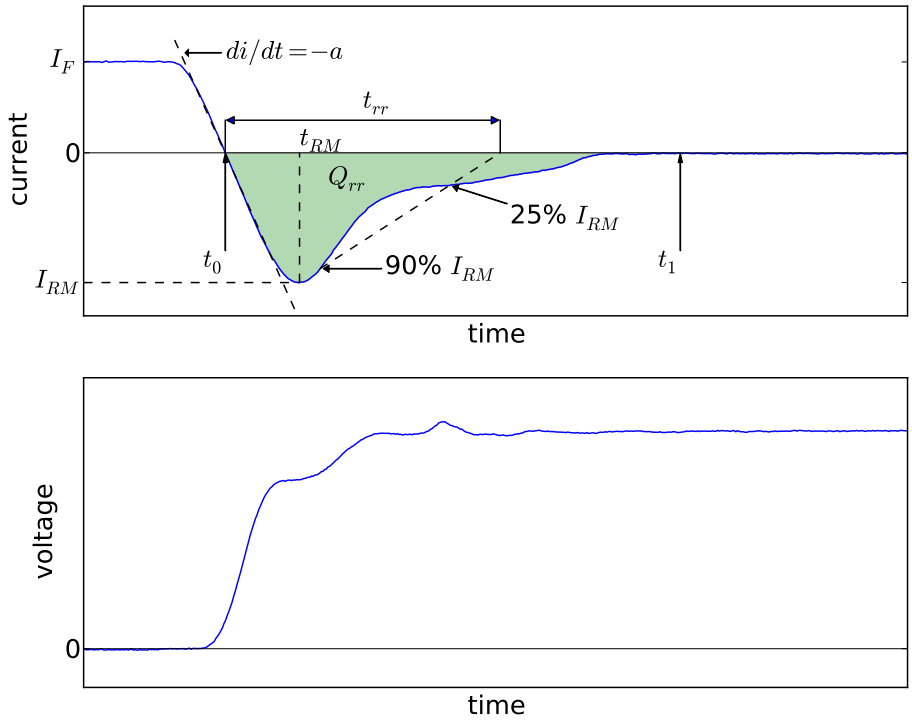


Figure 15: Definitions for the turn-off parameters of a diode.

circuit across the supply, current being limited only by the stray wiring inductance.

- Towards the end of the reverse recovery period if the reverse current falls too sharply, stray circuit inductance may cause dangerous over voltage across the device. It may be required to protect the diode using an RC snubber.

During diode turn off current and voltage exist simultaneously in the device. At high switching frequency this may result in considerable increase in the total power loss.

Important parameters defining the turn off characteristics are (see Figure 15):

- I_{RM}, maximum reverse recovery current. The peak value of the reverse current during commutation at the specified conditions.
- Q_{rr}, reverse recovery charge. The integral over time of the reverse current during commutation at the specified conditions starting at the zero-crossing of the current and ending when the reverse current has decayed to zero after the tail-current phase.
- t_{rr}, reverse recovery time. The commutation time of the diode at the specified conditions. It is measured between the current zero-crossing and the zero-crossing of a straight line drawn between 90% of the current peak on the rising flank and 25% of peak (on the falling flank).
- E_{rec}, reverse recovery energy. The energy dissipated during a single reverse recovery event. It is the integration of the product of the reverse current and voltage from t_0 to t_1 (see Figure 15) as expressed by

$$E_{rec} = \int_{t_0}^{t_1} i(t)v(t)\,\mathrm{d}t \tag{1.6}$$

In order to judge the behavior of diodes during turn-off more precisely, the circuit conditions need to be taken into consideration with respect to switching conditions [39]. In other words, the way a diode commutates from forward bias to reverse bias condition (or vice versa) depends on the external circuit. The number of power diode applications in high power systems continues to grow leading to various dynamic constraints and hence different diode designs and behaviors. A more detailed analysis of power diode physics is presented in [89]. Further reading on power diode applications can be found in [77].

In parallel to the research to scale up the diode power handling limits, extensive studies were made to model and simulate the device behavior. These two latter activities are as much important as the first one, especially for the applications engineers, since they allow to evaluate device performance without always resorting to costly and time consuming experimental approaches. Many different types of diode model exist, depending on the level of accuracy required, the capability of capturing the device physical phenomena rather than providing a simple behavioral model of it, the nature of the modeling approach (for example numerical or analytic one) etc.

In the following, different types of models are identified and briefly explained. Among the whole spectrum of modeling technique, the Lumped Charge technique is described and the key ideas are pointed out. We adopted such technique to develop an automated parameter optimization routine during our internship at ABB Corporate Research Center in Switzerland.

1.3 WHY SIMULATE?

As we have already pointed out, it is very important to model and simulate a device behavior in order to evaluate the device performance without always recurring to costly and time consuming experimental approaches.

In recent years, the simulation of discrete analog circuits has become more viable. This has come about because of the almost relentless advances in CPU power, the increased availability of device models from their manufacturers and the introduction of easy to use and affordable simulation tools.

The pressure to reduce product development time-scales has meant that for many projects the traditional bread-boarding phase is skipped altogether - with or without simulation - and circuit development is carried out on the first revisions of Printed Circuit Board (PCB). The use of simulation on a circuit or parts of a circuit can help to eliminate errors in a circuit design prior to this stage and reduce the number of PCB revisions required before the final production version is reached. Of course, to be useful, the simulation process must therefore not be too time consuming.

Computer simulation, does however, have many more uses. There are some things you can do with a simulator which cannot be achieved with practical approaches. You can remove parasitic elements, you can make non-invasive measurements that are impossible in real-life or you can run components outside of their safe operating area. These abilities make simulation a valuable tool for finding out why a particular design does not behave as expected. If the problem can be reproduced on a simulator then its cause can be much more easily identified. Even if a problem cannot be reproduced then this gives some clues. It means that it is caused by something that is not modeled, a wiring parasitic perhaps.

Simulation is extremely useful for testing ideas at the system level. Sometimes it is not easy to test a concept because the hardware to implement it is very costly or time consuming to build. It may even be that you do not know how to implement the idea in hardware at all. The alternative is to design a model and simulate it with a computer. Once it has been established that the concept is viable then attention can be given to its implementation. If it proves not to be viable, then a great deal of time will have been saved.

1.4 CLASSIFICATION OF MODELS

The most fundamental aspects that distinguish among different power device models are the model formulation technique and concept employed.

Very broadly, diode and IGBT models may be classified either as “micromodels”, or as “macromodels” [91], [87]. Micromodels are closely based on the internal device physics and, if properly formulated, should yield good accuracy over a wide

range of operating conditions. Because device physics unavoidably require mathematical equations, micromodels are also known as mathematical models. Macromodels reproduce the external behavior of the device largely by using empirical techniques without considering its geometrical nature and its internal physical processes. This external behavior is usually modeled by means of simple data-fitting empirical equations, lookup tables, or an electrical subcircuit of common components to emulate known experimental data. Because of the latter reason, macromodels have been mistakenly labeled as subcircuit models. Many so called subcircuit models are actually micromodels because they used subcircuits to simulate fairly complex physics-based mathematical equations. In our work here, the term subcircuit refers to the mode of model implementation rather than the model formulation technique.

Because of their limitations in terms of accuracy and flexibility, macromodels are rarely utilized nowadays. Micromodels are generally more computationally efficient, more accurate, and more related to the device structure and fabrication process. Micromodels can be further classified as numerical models, analytical models and hybrid models.

NUMERICAL MODELS Numerical modeling uses the partial differential set of the semiconductor physics and solves them using finite element or finite difference methods. A finite element Technology Computer-Aided Design (TCAD) simulation of the reverse recovery of a single diode can take a few minutes or tens of minutes depending on the discretization, on the circuit, and on the physical phenomena included. From the engineering point of view, the degree of accuracy that is achieved by an exact numerical model is not always necessary or even justified, in particular, if the input data, such as the doping profile, is only known with a limited accuracy. In such cases, simplified numerical models may suffice. The simplifications may be in the form of assumptions made to semiconductor physics, or in the finite-element algorithms. Numerical models are suitable for device manufacturers who want to evaluate the performance of their devices in power electronic circuit applications.

ANALYTICAL MODELS Analytical modeling relies on a set of mathematical functions to describe the devices' terminal characteristics without resorting to finite-element methods. These equations could also be device physics related to provide realistic simulations over a wide range of operating conditions. The computational overheads of analytical models are far lower than those of numerical models. Circuit simulators can solve these types of models most efficiently. Having power device models in the libraries of simulators allows the latter to function as general purpose power electronics circuits CAD tools. Analytical models are, thus, very appropriate for simulation of power electronic circuits over a large number of switching cycles.

HYBRID MODELS This formulation technique uses a combination of numerical and analytical models. A discretization approach (such as finite differences) is used to solve the semiconductor equations in the drift region only. Analytical equations are applied to the rest of the device structure. This procedure

has the advantage that a high accuracy of the charge carrier behavior may be simulated without the long execution time associated with fully numerical models. These models show good accuracy at the expense of computation efficiency which can still be too high for large system simulations or for Electro-Magnetic Interference (EMI) / Electro-Magnetic Compatibility (EMC) simulations. Hybrid models are, thus, suitable for very detailed study of device interactions with the rest of the circuit over a few switching cycles.

We wanted to use diode models which could be rather oriented to compactness and computational efficiency than extreme accuracy: in fact they are intended to be used as a first tool of analysis for the designer to evaluate the performance of a more general application. In this sense, also the complexity of the parameter extraction procedure must be accounted for in the choice of the model. Nevertheless, the model should be able to predict the most relevant physical phenomena governing the device operation.

All these factors led to the choice of the *lumped-charge modeling technique* [76] as the intended diode model.

1.5 THE LUMPED-CHARGE MODELING APPROACH

The lumped-charge modeling technique represents a systematic technique for analytical modeling. It was first introduced in 1991, based upon Linvill's lumped parameter approach [70] and the standard charge control method [1]. In the years, diode [68], [75], [5, 82], Silicon Controlled Rectifier (SCR) [74], GTO [73], MOS Controlled Thyristor (MCT) [52], MOSFET [9], Bipolar Junction Transistor (BJT) [90], IGBT [69] models have been created based on this approach.

Another important factor to be considered when choosing a model is how efficiently such a model can be described. Lumped charge models can be efficiently described with Hardware Description Language (HDL) [101], such as Verilog-A, VHDL-AMS, MAST, etc.

1.5.1 *Basics Concepts*

The lumped charge technique leads to a reduction of the model complexity, while retaining the basic structural information and internal carrier transport process of the device. Thus, it can offer a good trade off between complexity (i.e. computation times) and accuracy.

The lumped charge technique can be applied systematically to the model power semiconductors following these three steps:

- The device structure is discretized into several P- or N-type regions, each of which contains charge-storage and connection nodes. A charge-storage node is responsible for charge-carrier storage and recombination and is usually located near the center of a region; a connection node links junction voltage to the charge-carrier concentration level and is located on the junction depletion edge.

- The hole and electron charge values at each node are obtained by multiplying the local charge-carrier concentrations by the volume of the reference region, which is typically one of the lightly doped regions.
- The charge nodes are finally linked one to each other by means of the following six equation, derived from device physics and circuit theory:
  - current density equations;
  - current continuity equations;
  - charge neutrality equations;
  - Boltzmann relations (p-n junction equations);
  - Poisson equations;
  - Kirchoff current and voltage laws.
- The symbol u_{ij} is used to indicate any variable u defined between node i and node j.

The first five describe carrier distribution and transport between the charge nodes in the device; the last connects internal variables of the model equations to terminal characteristics.

This systematic approach can be followed since all power semiconductor devices can be viewed as a combination of the following representative structures:

- P^+N^- structure (or N^+P^-);
- N^-N^+ structure (or P^-P^+);
- MOS structure.

We will show how to apply the above equations to the first two structures by considering a power diode.

1.5.2 *Diode models*

During our internship in ABB Corporate Research Center in Switzerland three physics based models for the power diode has been tested:

- Lauritzen [68] model, 1991;
- Ma [75] model, 1997;
- Extended Lauritzen [5, 82] model, 2011.

These models have been developed using the lumped-charge modeling technique. Such technique provides a good trade off between accuracy and simulation speed. Moreover, extraction of model parameters is relatively fast and simple.

These three models have increasing accuracy in the order they are listed. Therefore it can be expected that they also have increasing simulation speed in the same order. This is provided that convergence problems are not encountered. In Extended Lauritzen model [5, 82], Ma model is extended while maintaining the low computational cost.

In the following, each of these three models will be discussed.

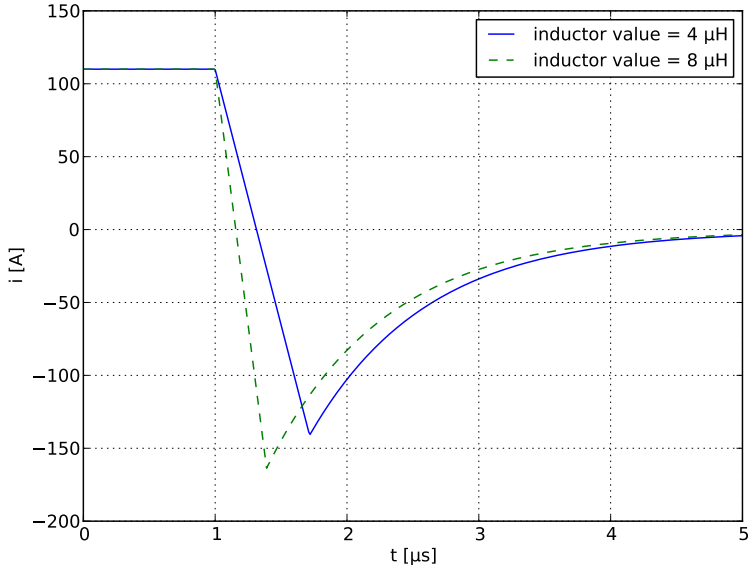


Figure 16: Lauritzen model, 1991. Inductive load [39] $i(t)$ turn off switching waveform for two different inductor values.

1.5.2.1 *Lauritzen model, 1991*

Lauritzen [68] model extends the basic charge-control diode model [1] to include a gradual reverse recovery, as shown in Figure 16. It makes the following assumptions:

- Power diode is the p-i-n diode. A p-i-n diode is a p-n junction with a doping profile tailored so that an intrinsic layer, the "i region", is sandwiched between a p layer and an n layer. In practice, however, an idealized i region is approximated by either a high-resistivity p layer or a high-resistivity n layer. The nature of the low doping in the i region causes most of the potential drop across this region.
- Hole and electron mobilities are equal.
- Base contraction due to the moving of the depletion region boundary is omitted, which results in single fixed time constant in the reverse recovery. So the voltage dependent reverse recovery is omitted.
- Forward conduction has attained a steady state.
- $\frac{\mathrm{d}i}{\mathrm{d}t}$ is a constant.
- A constant transit time is used.
- Emitter recombination due to high level injection is omitted.

For such reasons:

- Self-heating effect cannot be included.

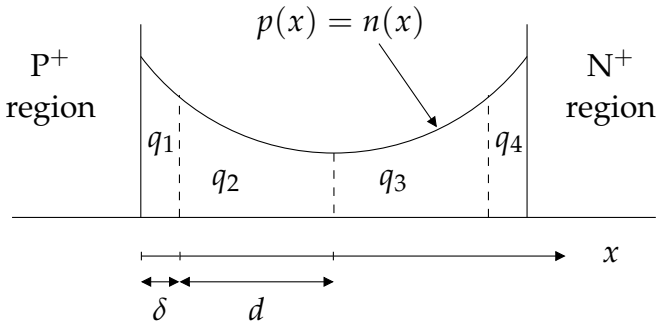


Figure 17: Charge storage locations in a p-i-n diode.

- The model should be applied to power diode with low peak inverse voltage so that base contraction may be neglected (i.e. short base diode) [106].
- The model is valid only if the ratio between the peak of the reverse current and the value of the current when it starts to decrease is less than 1 [66].

The process for deriving the model's governing equations is summarized next.

High injection condition is assumed, leading to the condition that the hole and electron distributions are equal, $n(x) = p(x)$. Since equal hole and electron mobilities are assumed, the charge distribution profiles are symmetric and the analysis can be conducted only for the charges on one side (referring to Figure 17, they are q_1 and q_2, which represent the lumped excess charges in their respective regions). Here $q_1 = qA\delta(p_1 - p_{io})$ and $q_2 = qA\delta(p_2 - p_{io})$, where

- q is the electron unit charge;
- A the junction area;
- δ and d the two charge storage regions;
- p_1 and p_2 the average hole concentrations in the regions corresponding to q_1 and q_2;
- p_{io} the equilibrium hole concentration.

High level diffusion is the mechanism for charge redistribution from q_1 to q_2, while holes are injected directly into q_1 from the p^+-i junction on the left. The discretized diffusion equation is

$$i(t) = -qA2D_p\frac{dp}{dx} = \frac{qA2D_p(p_1 - p_2)}{\frac{\delta}{2} + \frac{d}{2}}. \tag{1.7}$$

Here, D_p is the high level diffusion constant. To prevent an arbitrarily large current from flowing between q_1 and the external leads, the width δ must be made small compared to the width d. Taking the limit for $\delta \to 0$, (1.7) becomes

$$i(t) = \frac{q_0 - q_2}{T_{12}} \tag{1.8}$$

where $q_0 = qAd(p_1 - p_{io})$ and $T_{12} = d^2/(4D_p)$ is the approximate diffusion time across region d. Applying the current continuity equation to lumped charge node 2 yields

$$0 = \frac{\mathrm{d}q_2}{\mathrm{d}t} + \frac{q_2}{\tau} - \frac{q_0 - q_2}{2T_{12}}. \tag{1.9}$$

Finally, the excess charge q_0 can be determined by multiplying the p^+-i junction equation times qAd

$$q_0 = \frac{I_s \tau}{2}\left[\exp\left(\frac{v}{2V_T} - 1\right)\right]. \tag{1.10}$$

where $I_s = qA2dp_{io}/\tau$ and V_T is the *thermal voltage*. The thermal voltage depends on absolute temperature T, the magnitude of the electrical elementary charge q and the Boltzmann constant k as

$$V_T = \frac{kT}{q}. \tag{1.11}$$

Renaming $q_M = 2q_2$, $T_M = 2T_{12}$ and $q_E = 2q_0$, the model equations become:

$$i(t) = \frac{q_E - q_M}{T_M} \tag{1.12a}$$

$$0 = \frac{\mathrm{d}q_M}{\mathrm{d}t} + \frac{q_M}{\tau} - \frac{q_E - q_M}{T_M} \tag{1.12b}$$

$$q_E = I_s \cdot \tau \cdot \left[\exp\left(\frac{v}{n \cdot V_T} - 1\right)\right] \tag{1.12c}$$

In (1.12c) the emission coefficient n replaces the factor 2 in (1.10) to generalize the equation and make it similar to the Simulation Program with Integrated Circuit Emphasis (SPICE) equation [1]. For a complete diode model, the equations for the junction capacitance, temperature dependence of I_s and parasitic series resistance can be added. Junction capacitance and temperature dependence of I_s are modeled with the standard SPICE equations [1], while the series resistance R_s is modeled with (1.5).

The list of the model parameters is reported in Table 1. Section 1.6 describes parameter extraction and refinement using a formal optimization procedure.

1.5.2.2 *Ma model, 1997*

This diode model improves Lauritzen [68] model. It makes the following assumptions:

- during forward condition, the drift current is determined by the lowest carrier concentration in the region, and the higher concentration carrier can be ignored;
- during reverse recovery, drift current is negligible;
- average carrier lifetime is used in the base region.

SYMBOL	DESCRIPTION	UNIT
C_{jo}	Zero bias junction capacitance	F
F_c	Forward bias depletion capacitance coefficient	
I_s	Saturation current	A
m	Grading coefficient	
n	Forward emission coefficient	
R_s	Series resistance	Ω
T_{nom}	Parameter measurement temperature	°C
T_M	Diffusion transit time	s
τ	Minority carrier lifetime	s
v_j	Built-in potential	V
XTI	Saturation current temperature exponent	

Table 1: Lauritzen model, 1991. Parameter list.

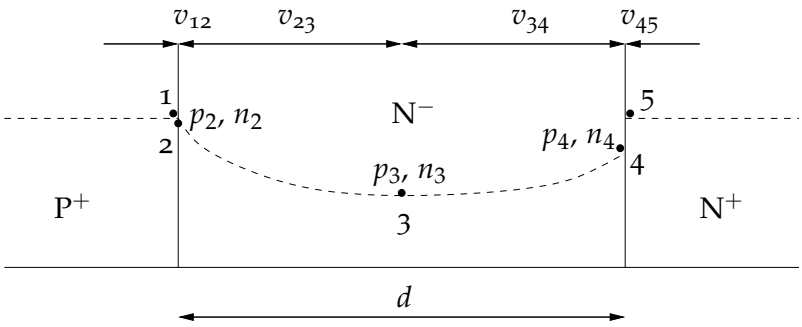


Figure 18: Location of charge nodes in the $P^+N^-N^+$ diode structure. Electron and hole carrier concentration variables at each node and voltage drop along the device are also shown.

Its main features are:

- it accounts for DC and switching characteristics in low and high level injection conditions (with and without emitter recombination);
- the voltage dependent reverse recovery is included;
- no self-heating effect is included;
- for the previous reasons the model applies to p-ν-n diode at all conditions where self-heating is not significant, and the spatial variation of the carrier lifetime in the base region is not excessive.

A typical one-dimensional forward-biased on-state charge carrier distribution in a $P^+N^-N^+$ is shown in Figure 13a.

Since the lightly doped N^- region stores the excess carriers, which determine the switching characteristics of the device, three nodes, indicated by 2, 3, and 4 in Figure 18, are chosen. Nodes 2 and 4 are connection nodes, and node 3 is a

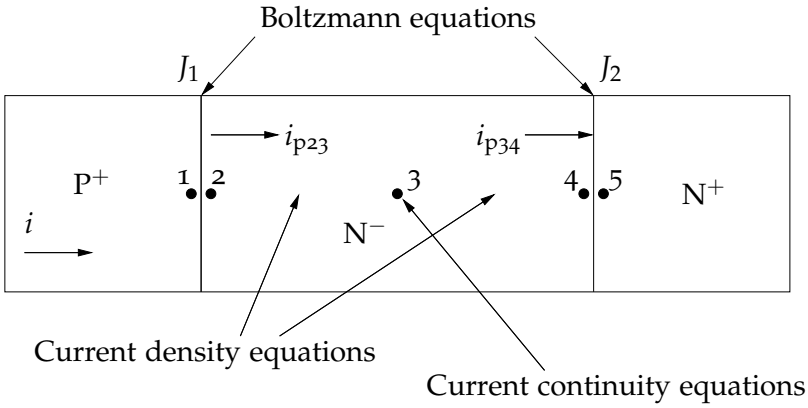


Figure 19: Locations of the lumped-charge nodes and device equations in a 1-D $P^+N^-N^+$ diode structure.

charge-storage node. Carriers in the heavily doped end regions do not penetrate deeply beyond the vicinity of the junctions. Hence, single-charge nodes 1 and 5 are sufficient. They serve as both connection and charge-storage nodes. At this point, the physical picture of the charge distribution in Figure 18 can be abstracted in the standard lumped-charge nodal representation depicted in Figure 19.

The process for converting the six groups of the fundamental semiconductor and circuit equations into the lumped-charge equations using the nodal assignment is outlined next.

CURRENT DENSITY EQUATIONS The one dimensional current density equation for holes is given by the following drift-diffusion relation [89]

$$j_p = qp\mu_p E - qD_p \frac{dp}{dx} \tag{1.13}$$

Then, due to the discrete nature of the lumped charge models, we must express the current as a function of the carrier concentration at each node

$$J_{p23} = q\frac{p_2 + p_3}{2}\mu_p \frac{v_{23}}{d_{23}} - qD_p \frac{p_2 - p_3}{d_{23}} \tag{1.14}$$

where the first term on the right hand side is the linearized drift component (in which the averaged carrier density has been used), the second term is the linearized diffusion one, and d_{23} is the distance between node 2 and 3. For simplicity the following approximation is used

$$\frac{p_2 + p_3}{2} \to p_3. \tag{1.15}$$

There can be several orders of magnitude difference between carrier concentrations at adjacent nodes. The term where this approximation is made is the drift term of the transport equation. The drift current is determined by the lowest carrier concentration in the region, and the higher concentrations are not so important. Thus, the approximation (1.15) is reasonable for forward conduction. In reverse recovery, the drift current is usually negligible.

Typically, the accuracy loss by these approximations can be compensated by parameter extraction [75].

Multiplying and dividing the previous equation by the N^- region width d we obtain

$$J_{p23} = d \cdot q p_3 \frac{D_p}{V_T} \frac{v_{23}}{d \cdot d_{23}} + d \cdot q D_p \frac{p_2 - p_3}{d \cdot d_{23}} \tag{1.16}$$

where the thermal voltage $V_T = D_p/\mu_p$. A new parameter T_{p23} is defined as the approximate hole diffusion transit time

$$T_{p23} = \frac{d_{23} \cdot d}{D_p} \tag{1.17}$$

The hole current equation in lumped-charge form can now be derived as

$$i_{p23} = A \cdot J_{p23} = \frac{q_{p3}}{T_{p23}} \frac{v_{23}}{V_T} + \frac{q_{p2} - q_{p3}}{T_{p23}} \tag{1.18}$$

where A is the device active area and $q_{pi} = qdAp_i$. An analogous derivation can be made for i_{p34}, i_{n23} and i_{n34}:

$$i_{p34} = \frac{q_{p3} - q_{p4}}{T_{p34}} + \frac{q_{p3}}{T_{p34}} \frac{v_{34}}{V_T} \tag{1.19}$$

$$i_{n23} = \frac{q_{n3} - q_{n2}}{T_{n23}} + \frac{q_{n3}}{T_{n23}} \frac{v_{23}}{V_T} \tag{1.20}$$

$$i_{n34} = \frac{q_{n4} - q_{n3}}{T_{n34}} + \frac{q_{n3}}{T_{n34}} \frac{v_{34}}{V_T} \tag{1.21}$$

$$\tag{1.22}$$

In order to simplify the model, and consequently the parameter extraction procedure, the charge node 3 can be assumed to be in the center of the drift region. In this way T_{p23} and T_{p34} become equal and, considering the parameter $b = \mu_n/\mu_p$, $T_{n23} = T_{n34} = T_{p23}/b$.

CONTINUITY EQUATIONS The continuity equations for holes is [89]

$$\frac{\partial p}{\partial t} = (G_p - R_p) - \frac{1}{q} \nabla \cdot J_p \tag{1.23}$$

where $(G_p - R_p)$ is the net hole generation/recombination rate. Since the total charge in the N^- region can be approximated by q_{p3}, the discretized form of the continuity equation becomes

$$i_{p23} - i_{p34} = \frac{q_{p3} - Q_{Bp}}{\tau_3} + \frac{\mathrm{d}q_{p3}}{\mathrm{d}t} \tag{1.24}$$

The first term on the right-hand side represents charge-carrier recombination (with $Q_{Bp} = qAdp_{no}$ being the thermal equilibrium hole charge in the N^-

base region) and the second term charge variation with time. An averaged carrier lifetime τ_3 is used. The electron continuity equation for the N^- base region is similar.

Current continuity equations are not written for connections nodes 2 and 4, since the whole base charge is modeled with q_{p3}. However, to include the effect of end region recombination, the continuity equation can be written for node 1 and 5 in Figure 19 [72]. End region recombination becomes important at high current levels, since it tends to lower the injection of carriers into the drift region, resulting in higher voltage drop across it. Neglecting the charge-storage effects in the end region, due to the high doping level, and treating the injection of minority carriers as a low level one:

$$i_{n23} = \frac{1}{\tau_1}\left[q_{n2}\exp\left(\frac{v_{12}-\phi_{12}}{V_T}\right) - Q_{Pn}\right] \tag{1.25}$$

$$i_{p34} = \frac{1}{\tau_5}\left[q_{p4}\exp\left(\frac{v_{45}-\phi_{45}}{V_T}\right) - Q_{Np}\right] \tag{1.26}$$

where

- parameters τ_1 and τ_5 are equal to the products of minority carrier lifetimes in the end regions times the diffusion volume of the end regions divided by the base region volume;
- Q_{Pn} and Q_{Np} are the electron and hole background doping charges in the P^+ and N^+ regions, respectively;
- ϕ_{12} and ϕ_{45} are the junction built-in potentials.

CHARGE NEUTRALITY EQUATIONS Outside the junction depletion region, electron and hole charges are related by the charge neutrality equations. Given the small dielectric relaxation time (≈ 1 ps), the electron charges in the quasi-neutral region are assumed to be instantaneously equal to the injected excess hole charges plus the ionized background doping charge Q_B:

$$q_{n2} = q_{p2} + Q_B \tag{1.27a}$$

$$q_{n3} = q_{p3} + Q_B \tag{1.27b}$$

$$q_{n4} = q_{p4} + Q_B \tag{1.27c}$$

P-N JUNCTION EQUATIONS The hole concentration in the drift region is regulated by the P-N junction equation, which relates the hole concentrations on each side of the junction and the voltage drop across it

$$p_2 = p_{no}\exp\left(\frac{v_{12}}{V_T}\right) \tag{1.28}$$

p_{no} being the equilibrium hole concentration in the drift region. Multiplying by the region volume gives the charge at node 2 and, following a similar approach, also the charge at node 4 in Figure 19 is determined:

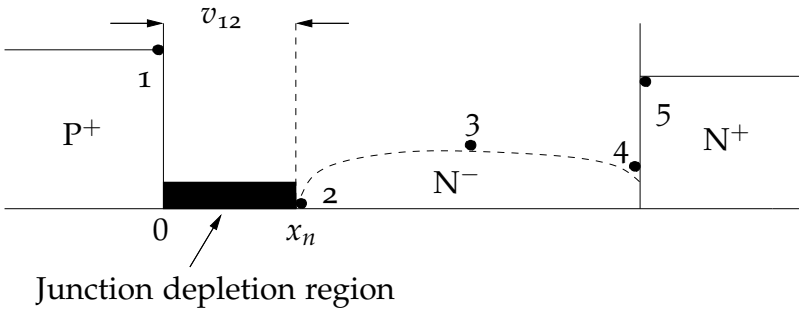


Figure 20: Charge distribution in the P^+N^- region during a reverse-recovery transient (the scale is relative).

$$q_{p2} = Q_{Bp} \exp\left(\frac{v_{12}}{V_T}\right) \tag{1.29}$$

$$q_{n4} = Q_B \exp\left(\frac{v_{45}}{V_T}\right) \tag{1.30}$$

POISSON EQUATION The Poisson equation is used to describe the effects of variation of junction-depletion width under reverse bias voltage [89]

$$-\frac{d^2v}{dx^2} = \frac{\rho(x)}{\varepsilon_{Si}} \tag{1.31}$$

Here, ε_{Si} is the silicon dielectric constant and $\rho(x)$ the charge density. As a reverse bias is incremented, the depletion region expands into the drift region and squeezes the charges into a smaller volume as shown in Figure 20.

The positive charges inside the depleted region are immobile ionized donors (n_{no}) and holes (p_i) injected from the P^+ region, which here we assume to travel at the saturation velocity v_{sat}

$$|i_{p23}| = qAp_i v_{sat}. \tag{1.32}$$

Integrating (1.31) twice over the depleted region and rearranging terms gives

$$x_n = \left(\frac{2\varepsilon_{Si}(\phi_{12} - v_{12})}{q\left(n_{no} + \frac{|i_{p23}|}{qAv_{sat}}\right)}\right)^{1/2} \tag{1.33}$$

where x_n is the depletion boundary (see Figure 20). According to the above equation, since the depletion width changes with the applied voltage, the transit time T_{p23}, defined in (1.17), also become variable and dependent on $(d - x_n)$:

$$l = \left(\frac{\phi_{12} - v_{12}}{\phi_B \left(1 + \frac{|i_{p23}|}{I_B}\right)} \right)^{1/2} \tag{1.34}$$

$$T_{p23} = T_{po}(1-l)^2 \tag{1.35}$$

where:

$$\phi_B = \frac{q n_{no} d^2}{2\varepsilon_{Si}} \tag{1.36}$$

$$I_B = q A n_{no} v_{sat}. \tag{1.37}$$

KCL AND KVL Kirchoff current laws (KCL) and Kirchoff voltage laws (KVL) are annexed to the model in order to link the device internal behavior with the external terminal quantities. Referring to Figure 18 and Figure 19:

$$v = v_{12} + v_{23} + v_{34} + v_{45} \tag{1.38}$$

$$i = i_{n23} + i_{p23} \tag{1.39}$$

$$i = i_{n34} + i_{p34} \tag{1.40}$$

The complete set of diode model equations ready for implementation follows:

$$i_{p23} = \frac{q_{p2} - q_{p3}}{bT_n} + \frac{q_{p3}}{bT_n}\frac{v_{23}}{V_T} \tag{1.41}$$

$$i_{n23} = \frac{q_{p3} - q_{p2}}{T_n} + \frac{Q_B + q_{p3}}{T_n}\frac{v_{23}}{V_T} \tag{1.42}$$

$$i_{p34} = \frac{q_{p3} - q_{p4}}{bT_n} + \frac{q_{p3}}{bT_n}\frac{v_{34}}{V_T} \tag{1.43}$$

$$i_{n34} = \frac{q_{p4} - q_{p3}}{T_n} + \frac{Q_B + q_{p3}}{T_n}\frac{v_{34}}{V_T} \tag{1.44}$$

$$i_{n23} = \frac{1}{\tau_1}\left[(q_{p2} + Q_B)\exp\left(\frac{v_{12} - \phi_{12}}{V_T}\right) - Q_{Pn}\right] \tag{1.45}$$

$$i_{p23} - i_{p34} = \frac{q_{p3} - Q_{Bp}}{\tau_3} + \frac{dq_{p3}}{dt} \tag{1.46}$$

$$i_{p34} = \frac{1}{\tau_5}\left[q_{p4}\exp\left(\frac{v_{45} - \phi_{45}}{V_T}\right) - Q_{Np}\right] \tag{1.47}$$

$$q_{p2} = Q_{Bp}\exp\left(\frac{v_{12}}{V_T}\right) \tag{1.48}$$

$$q_{p4} = Q_B\left[\exp\left(\frac{v_{45}}{V_T}\right) - 1\right] \tag{1.49}$$

$$l = \left(\frac{\phi_{12} - v_{12}}{\phi_B \left(1 + \frac{|i_{p23}|}{I_B}\right)} \right)^{1/2} \tag{1.50}$$

$$T_n = T_{no}(1-l)^2 \tag{1.51}$$

$$v = v_{12} + v_{23} + v_{34} + v_{45} \tag{1.52}$$

$$i = i_{n23} + i_{p23} \tag{1.53}$$

$$i = i_{n34} + i_{p34} \tag{1.54}$$

Neutrality equations (1.27) have been used to eliminate the electron charges in the above model equations.

- Equations (1.41)-(1.44) are electron and hole current equations between nodes 2 and 3 and nodes 3 and 4. T_n is the electron transit time and equal to T_{p23}/b.
- Equation (1.45) combines the p-n junction equation at J_1 and the simplified continuity equations for node 1, where charge-storage effects are neglected due to the short carrier lifetime observed in the heavily doped P^+ end region. Equation (1.47) is similar to (1.45), but for J_2 and node 5 in the N^+ end region.
- Equation (1.46) is identical to (1.24).
- Equations (1.48) and (1.49) are the P-N junction equations, identical to (1.29) and (1.30).
- Equations (1.50) and (1.51) account for the J_1 junction-depletion width variation effects.
- Equations (1.52)–(1.54) apply KVL and KCL to the model.
- The diffusion capacitance is embedded in the charge transport equations, namely, current (1.41)-(1.44) and continuity (1.45)-(1.47). The depletion capacitance is embedded in the Poisson equations (1.34)-(1.51).

For a complete diode model, the equations for the junction capacitance and parasitic series resistance can be added. Junction capacitance can be modeled with the standard SPICE equations [1], while for series resistance R_s equation (1.5) can be used.

Under most circumstances, the model equations can be solved directly for the dc current-voltage (I-V) characteristic as well as for the turn-on and turn-off transients. These solutions can be used to extract model parameters.

Some of the model parameters are shown in Table 2. Section 1.6 describes parameter extraction and refinement using a formal optimization procedure.

1.5.2.3 *Extended Lauritzen model, 2011*

Extended Lauritzen [5, 82] model extends Ma [75] model for high voltage power diodes by taking into account impact ionization.

Impact ionization occurs if the electric field in a junction is high enough, such that a noticeable number of electrons or holes in the statistical distribution gain sufficient kinetic energy that they can lift a valence electron by impact into the conduction band. Each ionizing carrier generates a pair of a free electron and hole, which again can generate further electron-hole pairs, thus giving rise to an avalanche event. Impact and avalanche generation are therefore used as synonymous expressions. Impact ionization is represented by *impact ionization rates* α_n, α_p defined as the number of electron-hole pairs generated per electron or hole per length of the path which the assembly travels with drift velocity v_n or v_p, respectively. The number of electron-hole pairs generated per unit of time, i.e. the *avalanche generation rate* G_{av}, is then given by

SYMBOL	DESCRIPTION	UNIT
C_{jo}	Zero bias junction capacitance	F
F_{c}	Forward bias depletion capacitance coefficient	
m	Grading coefficient	
R_{s}	Series resistance	Ω
T_{nom}	Parameter measurement temperature	°C
v_{j}	Built-in potential	V
α	Hole mobility coefficient	
$T_{R_{\mathrm{s}}}$	Temperature dependent coefficient of series resistance	K^{-1}
$T_{E_{\mathrm{r}}}$	Temperature dependent coefficient of E_{r}	K^{-1}
T_{τ_3}	Temperature dependent coefficient of τ_3	
T_{no}	Electron transit time	s
τ_3	Averaged carrier lifetime	s
Q_{B}	Thermal equilibrium electron charge in the N^- region C	
Q_{Bp}	Thermal equilibrium hole charge in the N^- region	C
E_{r}	Symmetric end-region recombination parameter	s C
NN	Low level ideality parameter	
NE	Medium-high level ideality parameter	
ϕ_{B}, I_{B}	Voltage-dependent reverse-recovery parameters	V, A
b	Mobility ratio of electrons and holes	

Table 2: Ma model, 1997. Parameter list.

$$G_{\mathrm{av}} = \alpha_{\mathrm{n}} \cdot n \cdot v_{\mathrm{n}} + \alpha_{\mathrm{p}} \cdot p \cdot v_{\mathrm{p}} = \frac{1}{q}(\alpha_{\mathrm{n}} j_{\mathrm{n}} + \alpha_{\mathrm{p}} j_{\mathrm{p}}) \tag{1.55}$$

On the right-hand side, the densities of the field currents are replaced by the total current densities, the diffusion currents are neglected because of the high fields. The avalanche generation rate appears in carrier continuity equations.

Much experimental and theoretical work has been carried out to determine the ionization rates and their field dependency, which is very strong. Experimental ionization rates were found by Chynoweth [10] to follow the relationship

$$\alpha_{\mathrm{n,p}} = a_{\mathrm{n,p}} \exp(-b_{\mathrm{n,p}}/E(x)) \tag{1.56}$$

where E is the electric field component in the direction of current flow. The parameters $a_{\mathrm{n}}, a_{\mathrm{p}}$ and $b_{\mathrm{n}}, b_{\mathrm{p}}$ are constants that depend upon the semiconductor material and the temperature. The experimental determination is done usually by measuring *carrier multiplication factors* M_{n} and M_{p}, consisting of double integrals over the ionization rates [71], which then have to be extracted.

With the avalanche generation rate (1.55) the continuity equations [71] for electrons and holes in the stationary one-dimensional case take the form:

$$-\frac{\mathrm{d}j_{\mathrm{p}}}{\mathrm{d}x} = \alpha_{\mathrm{p}} j_{\mathrm{p}} + \alpha_{\mathrm{n}} j_{\mathrm{n}} + qG \tag{1.57a}$$

$$\frac{\mathrm{d}j_{\mathrm{n}}}{\mathrm{d}x} = \alpha_{\mathrm{p}} j_{\mathrm{p}} + \alpha_{\mathrm{n}} j_{\mathrm{n}} + qG \tag{1.57b}$$

$$\tag{1.57c}$$

where G is the thermal generation rate and $\alpha_{\mathrm{p}}, \alpha_{\mathrm{n}}$ are as before the field dependent impact ionization rates. Equations (1.57) agree with the convention used in Figure 21.

The total current density

$$j = j_{\mathrm{p}} + j_{\mathrm{n}} \tag{1.58}$$

is independent of x under stationary conditions.

For a qualitative understanding we assume at first that the ionization rates are equal, $\alpha_{\mathrm{p}} = \alpha_{\mathrm{n}} = \alpha$, as is found e.g. in GaAs. Then equations (1.57) with (1.58) simplify to

$$-\frac{\mathrm{d}j_{\mathrm{p}}}{\mathrm{d}x} = \alpha j + qG = \frac{\mathrm{d}j_{\mathrm{n}}}{\mathrm{d}x} \tag{1.59}$$

Integrating over the depleted region gives

$$j_{\mathrm{p}}(0) - j_{\mathrm{p}}(x_{\mathrm{n}}) = j - j_{\mathrm{ns}} - j_{\mathrm{ps}} = \int_0^{x_{\mathrm{n}}} \alpha \, \mathrm{d}x + q x_{\mathrm{n}} G \tag{1.60}$$

since $j_{\mathrm{p}}(0) = j - j_{\mathrm{ns}}$ and $j_{\mathrm{p}}(x_{\mathrm{n}}) = j_{\mathrm{ps}}$ where $j_{\mathrm{ns}}, j_{\mathrm{ps}}$ are the minority carrier saturation current densities entering the depletion layer from the neutral regions. The current density j follows as

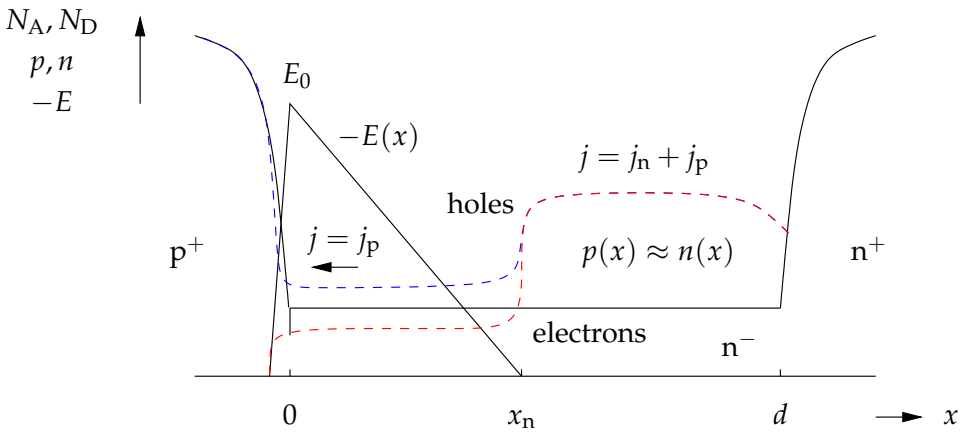


Figure 21: Diode during the turn off process.

$$j = \frac{j_{ns} + j_{ps} + j_{sc}}{1 - \int_0^{x_n} \alpha \, dx} \tag{1.61}$$

where $j_{sc} = q x_n G$. As shown by this equation, the effect of avalanche can be expressed by a multiplication factor

$$M = \frac{1}{1 - \int_0^{x_n} \alpha \, dx} \tag{1.62}$$

which enhances the three components of the thermal current density.

If $\alpha_n \neq \alpha_p$ as in the case of Si and most other semiconductors, the calculation is much more complicated. In [71] it is shown that the avalanche multiplication can be expressed generally by double integrals over the ionization rates. As expected, the three components of the thermal blocking current (numerator of (1.61)) are enhanced for $\alpha_n \neq \alpha_p$ each by a different multiplication factor and hence the leakage current density is generally given by

$$j = M_n j_{ns} + M_{sc} j_{sc} + M_p j_{ps} \tag{1.63}$$

Under reverse bias condition, when the applied reverse voltage reaches the break down voltage impact ionization causes the reverse current to increase very abruptly. In this way the blocking ability is lost.

Avalanche generation can also take place during fast switching, since free carriers, which have conducted the forward current before, are still present in the device and enhance the electric field. This stored charge is partially removed during the voltage increase, and it flows as hole current through the space charge region.

Figure 21 illustrates the process in a simplified way. Between the p-n junction at the position $x = 0$ and x_n the space charge region has extended, for supporting the applied voltage. Between x_n and the end of the lowly doped region exists a plasma zone, in which $n(x) \approx p(x)$ holds. The effects at the right side shall be neglected in this first approximation.

Through the space charge the current flows as hole current, $j = j_p$. The density of holes p can be calculated from the current density at this instant

$$p = \frac{j}{q v_{\mathrm{sat(p)}}} \tag{1.64}$$

In this equation $v_{\mathrm{sat(p)}}$ is the saturation drift velocity of holes under the condition of high fields. It amounts in silicon to approximately $1.0 \times 10^7\,\mathrm{cm\,s^{-1}}$ and is close to the saturation drift velocity of electrons $v_{\mathrm{sat(n)}}$. A current density j of $100\,\mathrm{A\,cm^{-2}}$ leads to $p = 8.2 \times 10^{13}\,\mathrm{cm^{-3}}$, which is already in the order of the background doping of the drift region of a power diode. The hole density p can no longer be neglected.

Holes have the same polarity as the positively charged ionized donors, hence their density now adds to the background doping n_{no} to an effective doping N_{eff}

$$N_{\mathrm{eff}} = n_{\mathrm{no}} + p \tag{1.65}$$

With the Poisson equation, N_{eff} determines the gradient of the electric field

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \frac{q}{\varepsilon_{\mathrm{Si}}}(n_{\mathrm{no}} + p) \tag{1.66}$$

so that $\mathrm{d}E/\mathrm{d}x$ is increased. With this, the field shape is steeper, E_0 is increased and the voltage, which drops across the space charge of width x_{n}, is increased in the first instance. However, E_0 can rise only up to the avalanche field strength E_{c}. E_{c} will now be reached at an applied voltage far below the specified rated blocking voltage of the device and avalanche will set in. This process, which is now dominated by free carriers, is called *dynamic avalanche*. It leads to the characteristic slight bend in the voltage waveform and to the characteristic rounded current pulse. Refer to [71] for more information about dynamic avalanche in fast diodes.

Most diode compact model do not account for impact ionization and therefore are characterized by very "angular" current waveforms with noticeable discontinuities in $\frac{\mathrm{d}i}{\mathrm{d}t}$ after the maximum reverse current I_{RM}. These effects can be seen by comparing Figure 15 and Figure 16. Inclusion of impact ionization significantly improves the precision of the reverse recovery current and voltage transients that can be a source of concerns for EMI/EMC simulations.

Extended Lauritzen model follows these steps to extend Ma model and then include dynamic avalanche:

- the Chynoweth law (1.56) is used to calculate avalanche ionization rates;
- in (1.56) the coefficients a and b are considered equal for holes and electrons;
- only the dominant of the holes moving through the space charge region is taken into account.
- the field $E(x)$ during reverse recovery through the $\mathrm{N^-}$ layer is calculated through

$$E(x) = \frac{q n_{\mathrm{no}} x}{\varepsilon_{\mathrm{Si}}} \tag{1.67}$$

This leads to a simplified expression for the impact ionization current

$$i_{ii} = i_{p23} a_p x_n \exp(-b_p / E(x_n)) \tag{1.68}$$

where i_{p23} has the same meaning as in Ma model.

This simplified expression can result in excessively large impact ionization at very large electric field E. A simple way to control this undesirable behavior without degrading the speed and convergence of the model is introducing a field dependent lifetime $\tau_3(E)$ at very high electric fields through the following expression

$$\tau_3(E) = \frac{\tau_H - \tau_L}{2} \frac{\tanh\left(k\left(\frac{E}{E_{th}} - 1\right)\right)}{\tanh k} + \frac{\tau_H + \tau_L}{2} \tag{1.69}$$

where τ_3 takes values from τ_L to τ_H. The speed of change between the two is defined by the parameter k and E_{th} is a threshold value of E. For a complete diode model, the equations for the junction capacitance and parasitic series resistance can be added. Junction capacitance can be modeled with the standard SPICE equations [1], while for series resistance R_s equation (1.5) can be used.

Some of the model parameters are listed in Table 3. It is the same set of Ma model except for:

- τ_L and τ_H, which replace τ_3;
- E_{th}, a threshold value of electric field;
- a_p and b_p, which model impact ionization.

Model parameter can be estimated from datasheets or from a single turn-off measurement.

Usually this procedure lacks in accuracy, i.e. it may fail to achieve very precise fitting of the voltage and current waveforms in reverse recovery, no matter which model we use. Therefore some refinement of the parameters must take place in order to improve the accuracy of simulations. Employing conventional optimization techniques to improve the model fitting is not trivial. This is particularly true when using Ma model or Extended Lauritzen model, since model equations are strongly coupled and very small changes in the parameters lead to very different results.

The next section describes parameter extraction and refinement using a formal optimization procedure.

1.6 PARAMETER EXTRACTION

In current work, the lumped-charge modeling approach has been followed, since it provides a good trade off between accuracy and simulation speed.

Moreover, model parameters may be fast extracted from datasheet and from measurements, as documented in [69] or [75], for example. This procedure lacks in accuracy. The initial estimated parameters cannot usually yield good fitting waveforms, so they must be refined.

The parameter extraction process can be summarized in the following list and in Figure 22:

SYMBOL	DESCRIPTION	UNIT
C_{jo}	Zero bias junction capacitance	F
F_c	Forward bias depletion capacitance coefficient	
m	Grading coefficient	
R_s	Series resistance	Ω
T_{nom}	Parameter measurement temperature	°C
v_j	Built-in potential	V
α	Hole mobility coefficient	
T_{R_s}	Temperature dependent coefficient of series resistance	K^{-1}
T_{E_r}	Temperature dependent coefficient of E_r	K^{-1}
T_{τ_3}	Temperature dependent coefficient of τ_3	
T_{no}	Electron transit time	s
τ_L	Carrier lifetime at low electric field values	s
τ_H	Carrier lifetime at high electric field values	s
Q_B	Thermal equilibrium electron charge in the N^- region	C
Q_{Bp}	Thermal equilibrium hole charge in the N^- region	C
NN	Low level ideality parameter	
NE	Medium-high level ideality parameter	
E_r	Symmetric end-region recombination parameter	s C
ϕ_B, I_B	Voltage-dependent reverse-recovery parameters	V, A
b	Mobility ratio of electrons and holes	
E_{th}	Threshold value of electric field	$V\,m^{-1}$
a_p, b_p	Impact ionization parameters	m^{-1}, $V\,m^{-1}$

Table 3: Extended Lauritzen model, 2011. Parameter list.

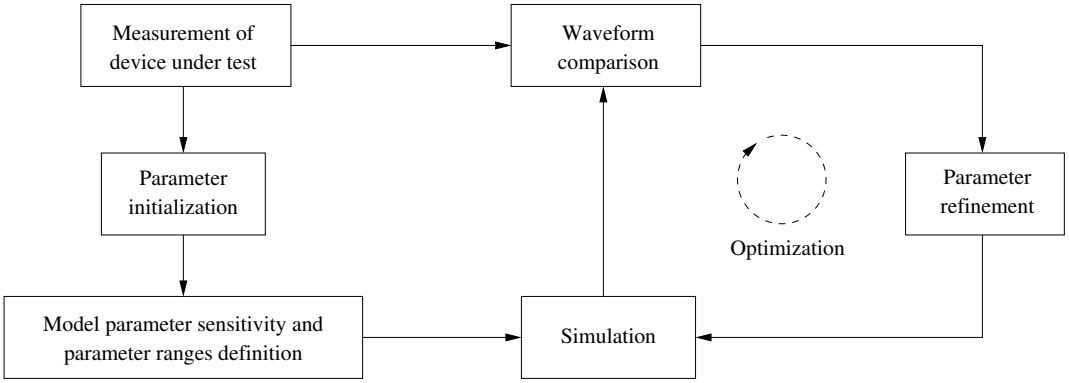


Figure 22: Diagram showing the procedure of parameter extraction.

1. Initial estimate of parameters made from device datasheets and basic measurements. Typically, for diode, measurements comprise steady-state junction capacitance vs reverse junction voltage characteristics, dc *i-v* characteristics and turn-off waveforms.

2. Study of the influence of model parameters on device characteristics. Usually some model parameters have a greater influence on some characteristics than on others. Concerning parameter ranges, they are not given sometimes, than we have to define them.

3. Simulation of circuit behavior using parameters estimated.

4. Comparison of simulated and measured characteristics to produce an error value.

5. Variation of the parameter values to minimize the error value.

Step 5 is the optimization procedure. Once the error value has been determined, the parameters are varied, and the simulation in step 3 re-executed to produce a characteristic valid for that set of parameters. This is again compared to measured characteristic in step 4, and the optimization continues accordingly. Once termination criteria are satisfied, the optimization procedure stops. If this process were manually carried out, it would be laborious, time consuming and it would require extensive engineering expertise to arrive at meaningful parameter sets. An automated parameter extraction procedure is essential.

Let us now describe the parameter extraction method more in detail.

1.6.1 *Step 1: initial parameter estimation*

Some methodologies that can be followed to find initial estimates of diode model parameters are:

- Manual tuning based on the empirical value range.

- Extrapolation using manufacturer's datasheets or measurements if they are available. In this case, we may still need to correct parameter values manually,

since the equations used for extrapolation can return rough, or even not-physical, results. It is therefore fundamental to have a clear understanding of the physical meaning of model parameters in order to understand if they need to be corrected.

To make the initial extraction procedure practical, the two methods have been employed jointly.

Parameter sets of Lauritzen, Ma and Extended Lauritzen models are given in Table 1, Table 2 and Table 3, respectively. Refer to Figure 15 for the definition of some turn-off parameters of a diode.

The following steps describe the parameter estimation, as we implemented it.

1. *Grading coefficient* m. As a starting guess, we set m to its value for an abrupt p-n junction, $m = 0.5$ [1].

2. *Diode junction capacitance at zero bias* C_{jo}. We set $C_{\mathrm{jo}} = C_{\mathrm{d}}(0)$, where $C_{\mathrm{d}}(0)$ is the junction capacitance at zero bias. If $C_{\mathrm{d}}(0)$ is not available from measurements, we can extrapolate or interpolate them to estimate it.

3. *Built-in potential* v_{jo}. Typically, v_{jo} may range from 0.2 to 1 V [1]. We estimate it by minimizing the sum of squared errors between the actual capacitance data and the corresponding SPICE model [1]

$$C_{\mathrm{d}}(v) = \frac{C_{\mathrm{d}}(0)}{\left(1 - \frac{v}{v_{\mathrm{jo}}}\right)^{m}}. \tag{1.70}$$

   where v is the reverse bias and m is the grading coefficient.

4. *Active die area* A. This parameter is usually given in the diode datasheets. Otherwise, since the maximum current density J is for most power diodes from 100 A cm^{-2} to 150 A cm^{-2} [8], the active die area A can be roughly estimated from the average forward dc current in the datasheet using

$$A = \frac{I_{\mathrm{F}}}{J} \tag{1.71}$$

5. *Minority carrier lifetime* (τ in Lauritzen model, τ_3 in Ma model). Neglecting the recombination effect in the reverse charge, equation (1.72) is used to make an initial estimate τ_0 for the high-level lifetime

$$\tau_3 \approx \tau_0 = \frac{Q_{\mathrm{rr}}}{I_F}. \tag{1.72}$$

   In the case that the reverse recovery charge Q_{rr} in the datasheet is given for a temperature other than room temperature, the parameter τ_0 can be scaled accordingly [104].

6. *Value of carrier lifetime at low and high electric field* τ_{L} and τ_{H}, respectively. Actual power diodes under inductive turn-off conditions at high rail voltages show the phenomenon of dynamic avalanche. This effect leads to the characteristic slight bend in the current waveform after the minimum current I_{RM}.

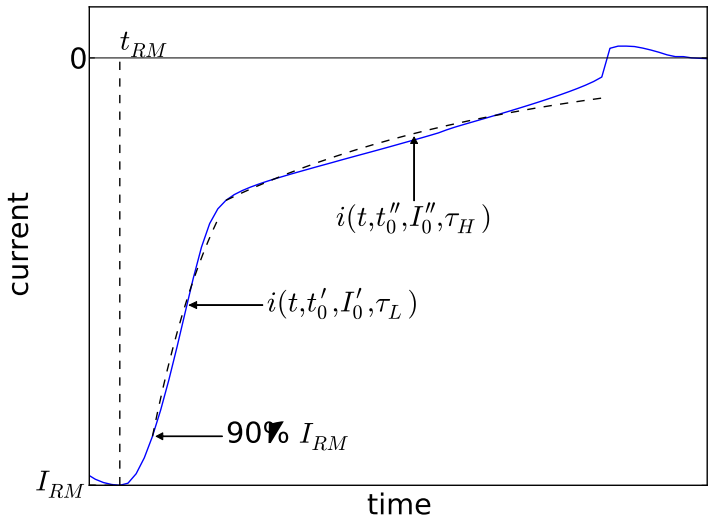


Figure 23: Starting estimation of parameters τ_L and τ_H.

Parameter τ_L is therefore estimated by minimization of the sum of squared residuals between the actual current data and the following model

$$i(t) = I_0 \exp\left(-\frac{t-t_0}{\tau_L}\right) \tag{1.73}$$

being $I_0 = 90\%$ of the reverse current peak on the rising flank and t_0 the instant when it occurs. Time t varies from t_0 to the instant where the waveform seems to start bending (see Figure 23). Parameter τ_H is estimated in a similar way. In this case I_0 is the value of the reverse current when it stops bending and t_0 is the corresponding instant. Time t varies from t_0 to the end of the switching waveform, generally. If the waveform exhibits a *step* decrease on the rising flank, the exponential model (1.73) (with τ_H instead of τ_L and with proper values of t_0, I_0) is fitted up to the step (see Figure 23).

7. *Reverse recovery time constant* τ_{rr}. After the peak of the reverse recovery current I_{RM}, the current decreases to zero approximately in an exponential way, with a time constant τ_{rr}, under inductive turn-off conditions. The parameter τ_{rr} can be approximately considered as a constant only when the diode is turned off at low reverse voltage, for instance, 10% of its breakdown voltage [75]. When this approximation cannot be made (see Figure 15), it is difficult to define τ_{rr}.

   In [68], it is shown that the following relationship holds

$$I_{RM} = a(\tau_3 - \tau_{rr})\left[1 - \exp\left(-\frac{t_{RM}}{\tau_3}\right)\right] \tag{1.74}$$

   All parameters can be determined from an experimental waveform:

   - current fall slope a;
   - peak reverse current I_{RM} and time t_{RM} at which it occurs.

When the parameter τ_{rr} cannot be easily defined from reverse recovery characteristic, we estimated it with (1.74).

8. *Electron transit time* T_{no}. In [75], it is shown that

$$\frac{1}{\tau_{rr}} = \frac{n}{bT_n} + \frac{1}{\tau_3} \tag{1.75}$$

where

- n is the forward emission coefficient of Lauritzen model. It is also called "ideality factor". We set it to its high level value $n = 2$ [75].
- b is the mobility ratio. We set it to its silicon value $b = 3$ [75].

Parameters T_n and T_{no} are equal under the same assumption for considering τ_{rr} as a constant [75]. We estimated T_{no} with equation (1.75) even when the latter assumption was not fulfilled.

9. *Transit time of Lauritzen model* T_M. This parameter can be estimated from equation (1.75) assuming that $\frac{n}{b} = 1$.

10. *Saturation current* I_s. This parameter is usually given in datasheets. In the case it is given for a temperature other than the temperature used for circuit simulation, it can be scaled using its temperature dependence in the SPICE model [1]

$$I_s(T_2) = I_s(T_1)\left(\frac{T_2}{T_1}\right)^{XTI/n} \exp\left[-\frac{qE_g(300)}{kT_2}\left(1 - \frac{T_2}{T_1}\right)\right]$$

where

- T_1 is the temperature in the datasheet at which I_s has been measured;
- T_2 is the temperature used for circuit simulation;
- XTI is the saturation current temperature exponent ($XTI = 3$ for a pn-junction [1]);
- n is the forward emission coefficient;
- $E_g(300)$ is the energy gap at 300 K.

Saturation current in a power diode is extremely sensitive to temperature variation.

11. *Series resistance* R_s. A starting guess of R_s can be computed by minimizing the deviation of the actual diode voltage data from the ideal exponential characteristic given by equation (1.5).

12. *Diode base width* d. If it has not been measured, it can be approximated with equation (1.76) [75]

$$d = \sqrt{2T_{no}D_n} \tag{1.76}$$

where $D_n = \mu_n V_T$ is the electron diffusion constant. Temperature dependence of the electron mobility μ_n should be taken into account [104].

13. *Thermal equilibrium electron charge in the base* Q_B and *medium-high level ideality factor* NE. These two parameters set the current in the medium to high level injection region [92]. If this data is available, Q_B and NE can be manually tuned until a fairly enough match is obtained. If this data is not available, we can estimate Q_B in the following way [75]:

    a) Estimate the diode base volume Vol_B: $Vol_B = A \cdot d$.

    b) Estimate the electron charge Q_B: $Q_B = qn_{no}Vol_B$, being n_{no} the base background doping. Parameter n_{no} is assumed to be $10^{14}\,\mathrm{cm}^{-3}$.

    The parameter NE may be set to the high level injection value of the forward emission coefficient n, i.e., $NE = 2$ [75]. In our experience, this is a very rough estimate.

14. *Thermal equilibrium hole charge in the base* Q_{Bp} and *low level ideality factor* NN. These two parameters set the current and conductance in the low level injection region [92]. They can be manually tuned until a good fit of low level injection data is obtained. Otherwise, an estimate Q_{Bp} can be devised from equations given in [75]:

    a) Estimate the diode base volume Vol_B.

    b) Estimate the electron charge Q_B.

    c) Estimate the hole charge Q_{Bp}: $Q_{Bp} = (Vol_B q n_i)^2/Q_B$, where n_i is the intrinsic carrier concentration at the given temperature [71].

    The parameter NN may be set to the low level injection value of the forward emission coefficient n, i.e., $NN = 1$ [75], in order to have a rough starting guess.

15. *Symmetric end-region recombination parameter* E_r. It is quite hard to find an initial estimate of this parameter. In [75] E_r is defined as

$$E_r = \tau_E Q_E$$

    where $\tau_E = \tau_1 = \tau_5$ and the end-region charge Q_E is given by

$$Q_E = Q_B \exp\left(\frac{\phi_{45}}{V_T}\right) = Q_{Bp} \exp\left(\frac{\phi_{12}}{V_T}\right)$$

    We assumed $\tau_1 = \tau_5 \approx \tau_3$ and took $\phi_{12}, \phi_{45} \in [0.2, 1], \phi_{12} > \phi_{45}$ since ϕ_{12}, ϕ_{45} are related to the built-in potential v_j. We want to stress that this procedure only helps to understand the order of magnitude of E_r.

16. *Voltage-dependent reverse recovery parameters* $I_B, R_B = \phi_B/I_B$. Approximate values for I_B and R_B are computed from (1.36) and (1.37). The hole saturation velocity v_{sat} which appears in (1.37) is given by $v_{sat} = 1.62 \times 10^8/T^{0.52}$ $\mathrm{cm\,s}^{-2}$ [71].

17. *Impact ionization rates* a_p, b_p. Initial guesses for these two parameters can be computed from the following relations

$$a_p = a \cdot \frac{\varepsilon_{Si}}{qn_{no}d}, \qquad b_p = b \cdot d$$

Proper values of ionization rates a and b can be found in [80]. Manual adjustment of these parameters may be needed.

18. *Threshold electric field* E_{th}. It is the threshold field at which impact ionization takes place. See [80] for an example of how this parameter may be initially extracted.

1.6.2 *Step 2: model parameter sensitivity and parameter ranges definition*

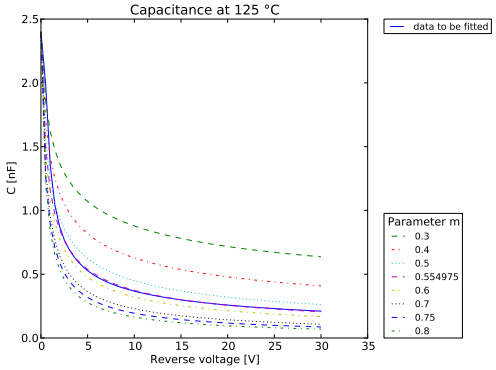
In general, the equations of the model of an electronic device are coupled and small changes in each parameter lead to different results in each output characteristic. However, some parameters may have a stronger influence on the shape of some characteristics than others. Identifying these relations helps to divide the overall problem of parameter extraction into smaller and hence simpler problems where one or more characteristics are fitted by using only a subset of the model parameters at a time. We followed two methodologies to carry out this study:

- DC and transient analysis of the model equations. Usually, every scientist introducing a new model for a device derives static and transient characteristics from the model equations. Looking at the analytic expression of characteristics provides a first hint for understanding the influence of model parameters.
- Model parameter sensitivity. Starting from an estimation of the model parameters, circuit simulations are executed repeatedly for several values of one parameter at a time. Output characteristics are then plotted to see how strong the influence of a parameter on each characteristic is.

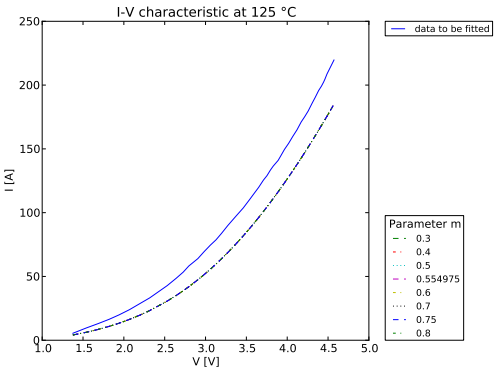
In Figure 24–26 it is shown how the grading coefficient m, the zero bias junction capacitance C_{jo} and the built-in potential v_{jo} influence the capacitance, DC and reverse recovery characteristics of a power diode. We can see that these three parameters determine the shape of the capacitance characteristic much more than that of the other characteristics. If we would plot similar figures for the other parameters, we could see that they do not influence the capacitance characteristic very much. We could therefore infer that the junction capacitance depends only on m, C_{jo} and v_{jo}. Actually, the same conclusion can be drawn by looking at the SPICE model for the junction capacitance (1.70). The junction capacitance C_d could then be fitted independently from DC and reverse recovery characteristics, i.e., we could extract m, C_{jo} and v_{jo} from reverse bias C_d and then keep them constant when fitting the other characteristics.

From the discussion on the junction capacitance, it should be clear that sensitivity analysis is useful to understand the parameter extraction problem better. However, its results have to be interpreted cautiously. Some of the reasons are explained below:

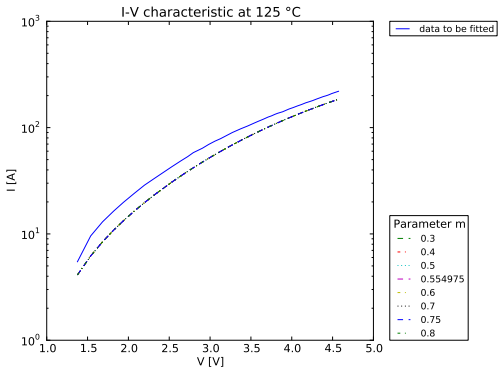
- Results may depend on the starting point used for the analysis. In this case, if the initial parameter set is a particularly bad estimate, there is the risk to make misleading conclusions about how a characteristic is influenced by a parameter. Usually, this does not happen when considering capacitance. In our experience, it always appear to depend only on m, C_{jo} and v_{jo} and to be decoupled from DC and transient characteristics.



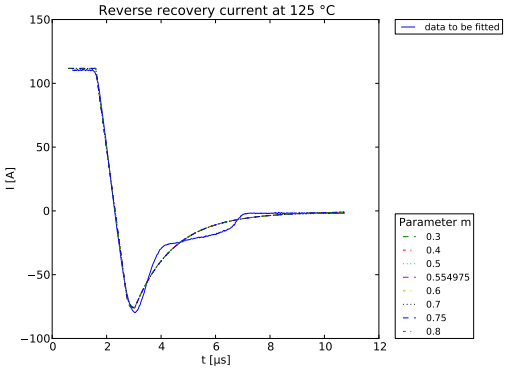
(a) Junction capacitance vs reverse bias voltage



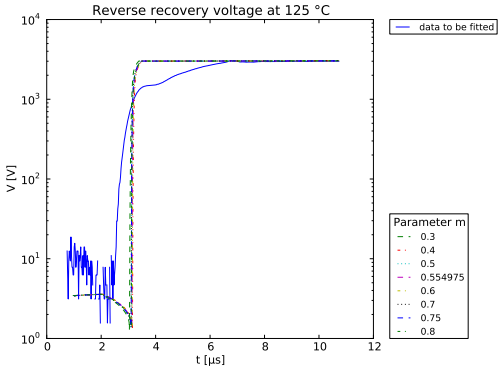
(b) DC current vs voltage



(c) DC log of current vs voltage

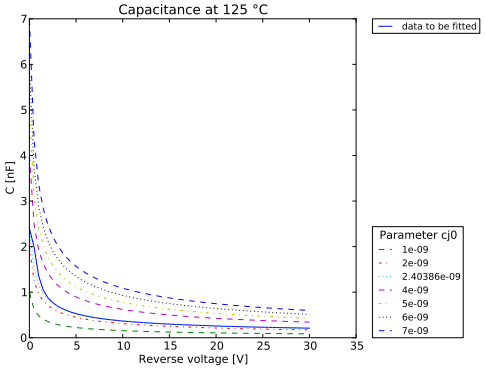


(d) Reverse recovery current vs time

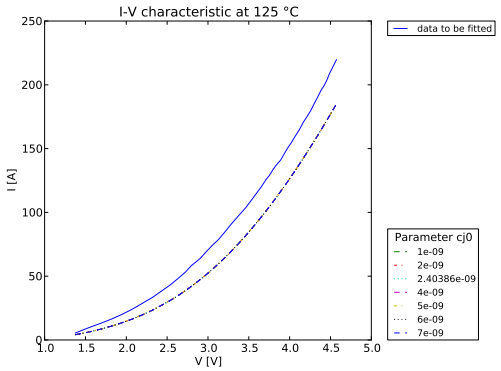


(e) Log of the absolute value of reverse recovery voltage vs time

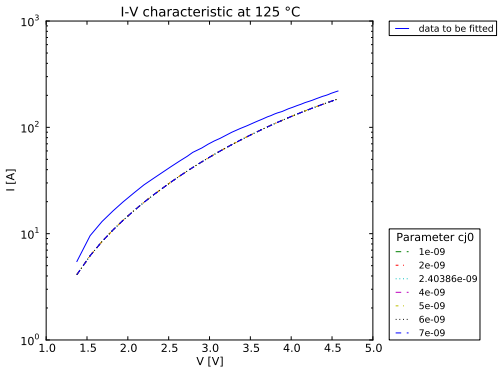
Figure 24: Sensitivity study for parameter m of Extended Lauritzen model.



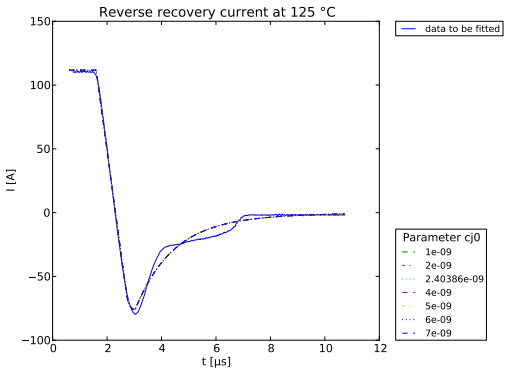
(a) Junction capacitance vs reverse bias voltage



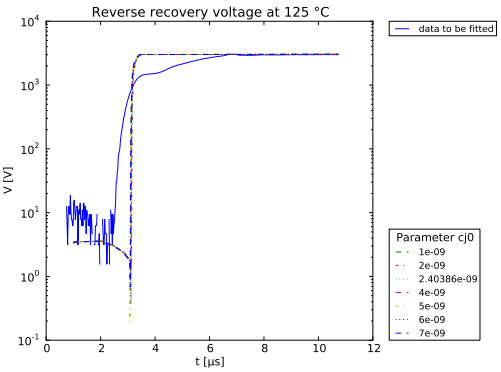
(b) DC current vs voltage



(c) DC log of current vs voltage

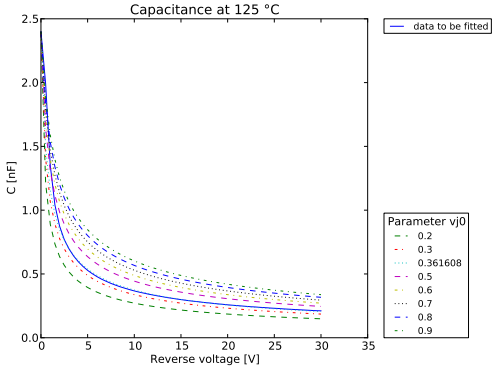


(d) Reverse recovery current vs time

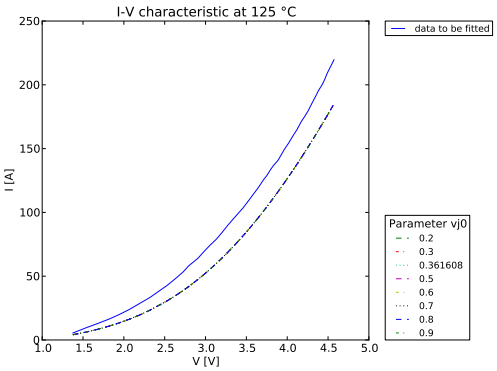


(e) Log of the absolute value of reverse recovery voltage vs time

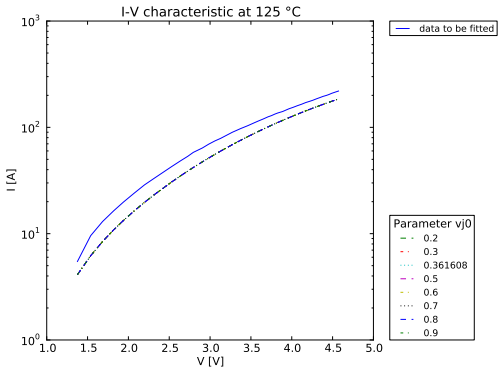
Figure 25: Sensitivity study for parameter C_{j0} of Extended Lauritzen model.



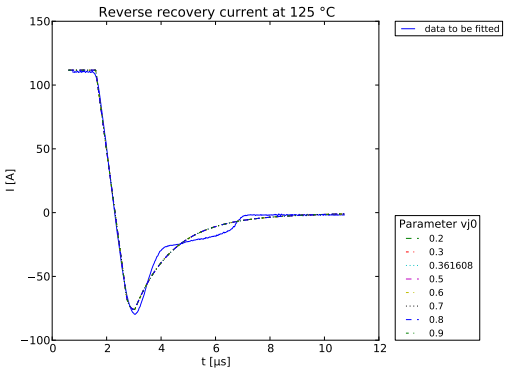
(a) Junction capacitance vs reverse bias voltage



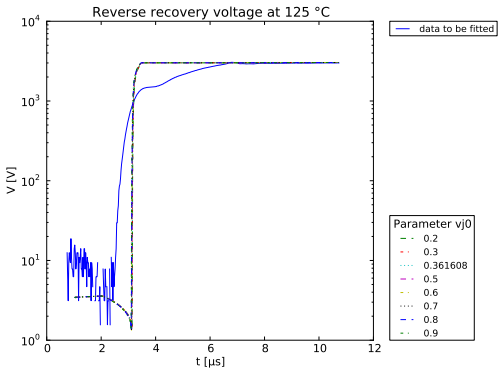
(b) DC current vs voltage



(c) DC log of current vs voltage



(d) Reverse recovery current vs time



(e) Log of the absolute value of reverse recovery voltage vs time

Figure 26: Sensitivity study for parameter v_{j0} of Extended Lauritzen model.

- Some parameters may influence several characteristics. For example, if we consider Ma model, both T_{no} and τ_3 have a strong effect on both the $i - v$ and the reverse recovery current characteristics. In these cases, sensitivity analysis does not provide enough evidence to decide if a parameter should be used to fit one characteristic and/or the others.

- More parameters than what it is expected seem to have an effect on a characteristic. This happens when model equations are strongly coupled, for example. Considering all the parameters selected by a sensitivity analysis in order to fit a characteristic may be not a good idea. The larger the set of parameters which are varied, the harder the exploration of the parameter space for the optimizer used to fit measurements. This is due to the exponential increase in volume associated with adding extra dimensions to a mathematical space. This phenomenon is known as *curse of dimensionality* [102].

  Moreover, a useless inclusion of some parameters can allow bad estimates of some of them to be compensated by incorrect variations of other parameters. A variation is said *incorrect* if the final parameter value does not have a physical meaning.

Two possible approaches to help understanding the influence of model parameters on output characteristics are: a deep understanding of the meaning of model parameters and an analytical reading of the papers where device models are described. Every good paper introducing a new model usually includes a section describing parameter extraction for the model itself. Of course, what is stated in such papers has to be adapted to the situation of the end-user.

It is worth pointing out that performing sensitivity analysis is essential when fitting one characteristic at a time. We call this fitting strategy a *single objective* approach. Sensitivity analysis becomes less important when a *multi objective* approach is adopted, which is a procedure which tries to fit several characteristics at the same time. Both the single and the multi objective approaches will be described later.

The last problem we consider in this section is the definition of model parameter ranges. If empirical value ranges are not known, they have to be computed. We performed this search as a succession of single objective optimizations, following these steps:

1. initial estimation of model parameters;

2. selection of the most influential parameters for each characteristic, on the basis of sensitivity analysis;

3. for each selected parameter, definition of a *narrow* range containing the initial estimate;

4. variation of the selected parameters in order to minimize the error between measured and simulated data;

5. parameter range expansion until a satisfactory solution is found by the optimizer:

6. repetition of this procedure for each characteristic.

Some observations are needed:

- The definition of a *narrow* range for a parameter depends on the parameter itself. For example, the DC *i-v* characteristic is extremely sensitive to the series resistance R_s. A good initial range for R_s would then be one decade wide at most. Instead for another parameter whose value is known with a bigger margin of error, like the electron charge in the base Q_B for example, we could use an initial range two-decade wide.
- It is better to start with narrow rather than wide ranges. If wide ranges are used an optimizer may return a not-physical combination of model parameters more easily.
- The error minimization is carried out with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which is an evolutionary algorithm for difficult non-linear non-convex optimization problems in continuous domain. This algorithm will be described in Section 3.2. Being CMA a stochastic optimizer, if it is applied several times to the same problem, each time a different solution may be returned. Therefore, it could be useful to run CMA several times for the same initial parameter set.
- The CMA-ES turns out to be a particularly reliable and highly competitive evolutionary algorithm for global optimization [45]. However, in order to lower the probability to find just a local minimum, several initial parameter sets should be used.
- When defining the termination criterion of CMA, tolerances should not be too strict, otherwise it may happen that big parameter variations do not correspond to a big error decrease. This can be detected by plotting parameter and objective values vs the number of objective function evaluations.
- If a parameter returned by the optimizer is close to its lower or upper bound, a better solution could be found by making that bound less strict. The situation can be complicated if several parameters are close to their lower or upper bound. In this case, it is necessary to decide which bounds should be relaxed, while taking into consideration the effect that the new solution could have on the shape of other characteristics, and hence, on value ranges of the corresponding most influential subset of model parameters.
- If a value range has to be widened, small bound relaxations should be tried first. Moreover, if the solution corresponding to a bound relaxation is expected to be not physical, that bound should not be relaxed.
- Bound relaxation should be carried out until at least one of the following conditions is fulfilled:
  - the solution found by the optimizer yields a good match between measurements and simulations and it is not close to parameter boundaries;
  - a further bound relaxation would not produce a physical solution.

From these observations, it follows that the process of computing parameter ranges must be carefully monitored and it could be laborious and time consuming.

1.6.3 *Step 3: device and circuit simulation*

Any model formulated would require a circuit simulator as a vehicle for simulation. Lumped charge models can be implemented in Verilog-A language. Verilog-A is a language for defining analog models with a high level of abstraction. It is the continuous-time subset of Verilog-AMS [12].

The method of adding a Verilog-A model varies from one simulator to another. Our work has been carried out in SIMetrix Micron-VX [17] and ngspice [14].

The aim of this subsection is to explain some of the causes of simulation failure we experienced when using SIMetrix and ngspice.

1.6.3.1 *Simulation and convergence issues in SIMetrix Micron-VX*

SIMetrix Micron-VX is a commercial analog circuit simulator aimed at integrated circuit designers. It offers support for popular IC design transistor models and real-time noise analysis allowing the study of noise in large signal systems such as mixers, oscillators and switched capacitor circuits. Full documentation on using SIMetrix simulator is available online [17].

SIMetrix uses a compiler to translate Verilog-A source into program code using the "C" language. This in turn is compiled into a shared library which has extension .sxdev. This library is then loaded into the SIMetrix memory image, making the new device ready for use. A SIMetrix Micron-VX license is required to load a .sxdev file. This implies that:

- A VX license can only be used by one simulation at a time. So it is not possible to have two different simulations running on the same machine exactly concurrently.
- After loading a .sxdev file, a VX license is released. So if we have to run several simulations on a local machine and we are using a network license, each time the license is released after a simulation, it can be caught by another computer in the same network.

In both the cases, the program execution is stopped until a local, or a network, license is available once again. This may happen several times because extracting model parameters with an automated procedure requires thousands of simulations to be run. The total execution time could significantly increase.

A simulation can also fail for other reasons. For example, in transient and DC analyses, SIMetrix uses the Newton-Raphson method [79] to solve nonlinear equations arising from circuit description: it consists in linearizing the equation around a suitable initial guess and solving for it. Then the method is iterated until an appropriate solution is found, based on the tolerance and precision requirements.

The Newton-Raphson algorithm requires some conditions to converge to a solution [79]:

- the nonlinear equation must have a solution (an isolated one);

- the equations must be continuous;
- the algorithm needs the equation's derivatives;
- the initial approximation must be close enough to the solution.

The Newton-Raphson algorithm terminates when both of the following conditions hold:

1. Node voltage differences between the last iteration k and the current one $(k+1)$ satisfy

$$\left|v_i^{(k+1)} - v_i^{(k)}\right| \leq RELTOL \cdot \max\left\{\left|v_i^{(k+1)}\right|, \left|v_i^{(k)}\right|\right\} + VNTOL \tag{1.77}$$

2. Differences between the value of the nonlinear function defining the current computed for last voltage and the linear approximation of the same current computed with the actual voltage satisfy

$$\left|\widehat{i_{\text{branch}}^{(k+1)}} - i_{\text{branch}}^{(k)}\right| \leq RELTOL \cdot \max\left(\widehat{i_{\text{branch}}^{(k+1)}}, i_{\text{branch}}^{(k)}\right) + ABSTOL \tag{1.78}$$

where

- $\widehat{i_{\text{branch}}}$ indicates the linear approximation of the current;
- $RELTOL$ is the relative tolerance for voltage and currents;
- $VNTOL$ is the absolute node voltage tolerance;
- $ABSTOL$ is the absolute current branch tolerance.

Relation (1.77) is called *update criterion*. This is very important in nodes characterized by a high impedance, such as a reverse-biased p-n junction: in fact, there is a large range of voltages that result in the current through the junction being less than the absolute current tolerance. In this situation, the update criterion is more important than the residual criterion for maintaining the accuracy of the solution.

Relation (1.78) is named the *residual criterion*, very important in nodes with small impedance, such as a forward biased p-n junction. In fact, small changes in voltage across the junction result in large changes in the current through it. In such circuits, the residual criterion is more important than the update criterion for maintaining the accuracy of the solution.

In DC analyses, if a solution cannot be found with the default method, most circuit simulators use continuation techniques to try resolving the DC operating point. These all work by repeating the iterative process while gradually varying some circuit parameter. The circuit parameter is chosen so that at its start value the solution is known or trivial and at its final value the solution is the operating point that is required.

Continuation techniques available in SIMetrix are [16]:

- source stepping;
- GMIN stepping;

- pseudo transient analysis.

For a DC analysis to fail, and assuming the default settings are being used, the standard solving method and all the continuation methods must fail.

DC analysis is the basis for the principle of operation of the transient one. In transient analysis, the nonlinear first-order differential equations representing the circuit behavior are first written. Then, the time derivatives appearing in these equations are replaced by their finite difference approximations (known as *integration formulas*) which discretize time. This allows to consider the transformed equations, at each discretized instant, as a time independent nonlinear set of equations. Then, a solution can be found at each time instant with the same iterative algorithm used in the DC analysis. Each determined solution represents the initial condition for the following analysis.

At every instant, the error introduced by the time-step choice is estimated and used to refine the time-step prediction.

Basically there are four reasons for convergence failure in transient analysis.

1. there is no solution to the circuit within the numerical range of the computer;
2. one or more device models contains a discontinuity or a discontinuity in its first derivative;
3. the circuit has a discontinuity caused by undamped regenerative action;
4. the solution matrix is ill-conditioned and the machine does not have sufficient accuracy to solve it up to the required tolerance.

1 and 3 are circuit problems. 2 is usually a software problem that the user can do little about. 4 can be caused by:

- very small resistors especially if they are not connected to ground;
- very large capacitors especially if they are not connected to ground;
- very large inductors especially if they are not connected to ground;
- circuit forced to use very small time steps perhaps because of fast rise/fall times;
- very large currents/voltages;
- very high gain loops.

By "very" in the above we mean extreme. 1000 V is not a very large voltage but 1000 MV volts is. 1 mΩ is not particularly small but 1 pΩ is.

Two other reasons why a DC or a transient simulation may fail are:

- Singular matrix errors. A singular matrix error occurs when the circuit does not have a unique and finite solution. For example, a circuit containing a floating capacitor does not have a unique DC solution as the capacitor can be at any voltage. Also circuits with shorted inductors, voltage sources or a combination of both will fail with this error.

- "Time step too small" error. It means that, because of the nature of a circuit, to achieve the required accuracy, a time step smaller than the minimum permissible was needed.

Many other errors can cause a simulation to fail. They will be discussed in Appendix A.

1.6.3.2 *ngspice*

ngspice is a mixed-level/mixed-signal circuit simulator. Its code is based on three open source software packages: Spice3f5, Cider1b1 and Xspice. ngspice is part of gEDA project [13], a full GPL'd suite of Electronic Design Automation tools. Unlike SIMetrix, ngspice does not suffer from any license-related problem, since it is an open source simulator.

Verilog-A models can be integrated into ngspice by using the ADMS compiler developed by Laurent Lemaitre (`http://www.noovela.com`).

Methods used to carry out DC and transient analyses are similar to those implemented in SIMetrix. Hence, the reasons why a circuit simulation fails are analogous as well.

Continuation techniques available in ngspice are [14]:

- source stepping;
- GMIN stepping.

A more detailed description of the errors that can occur when using ngspice is reported in Appendix A.

1.6.3.3 *Handling non-convergence situations*

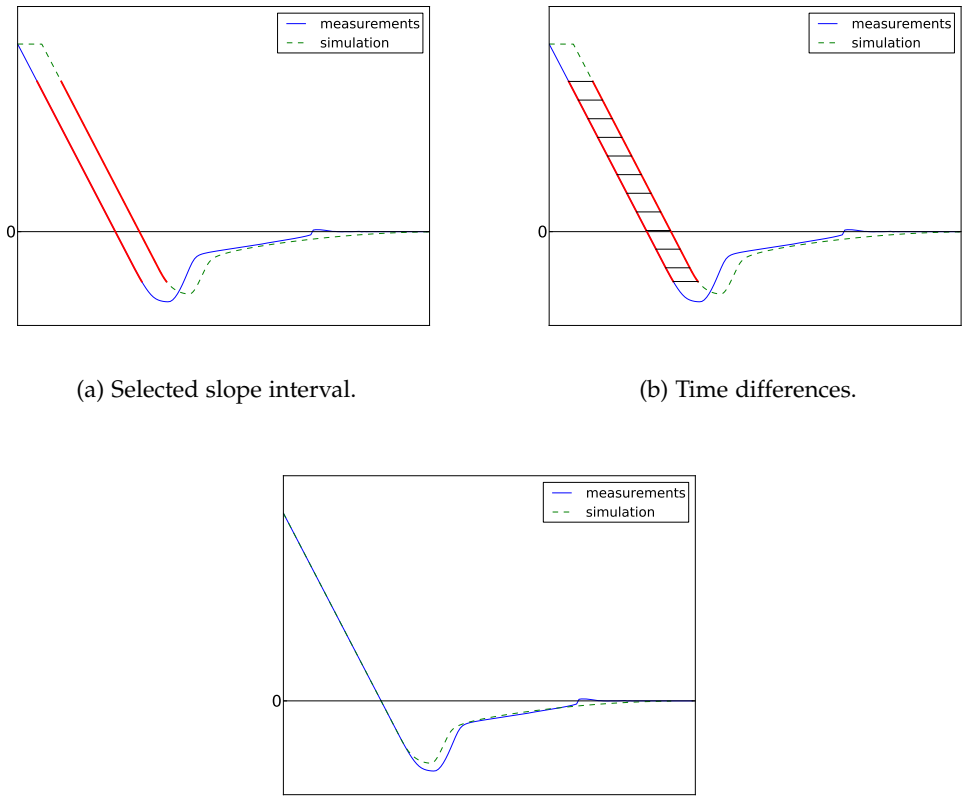
In this work, the parameter extraction procedure has been carried out through evolutionary algorithms. These methods try to find the parameter values which minimize the error value by introducing random parameter variation [11]. This helps to search the parameter space completely and relatively quickly. A side effect is that combinations of model parameters which cause non-convergence situations like those we described before, are found very often.

In Appendix A we will describe which strategies have been implemented to handle simulation failures in order to prevent a parameter extraction process to stop prematurely.

1.6.4 *Step 4: waveform comparison*

One possible method of comparison of the measured and simulated characteristics is using the salient points of the waveforms, e.g., diode reverse recovery current, reverse recovery time or IGBT switching time. This gives a small number of points to match, but can suffer from low accuracy. Most importantly, the waveforms may differ quite substantially during the switching instants, especially in $\mathrm{d}i/\mathrm{d}t$ and $\mathrm{d}v/\mathrm{d}t$. This would give a large error in estimated power during simulation.

A much more accurate method of comparison is to calculate the sum of the squared errors at each voltage or time point in the characteristics. This requires:



(a) Selected slope interval.

(b) Time differences.

(c) Aligned characteristics.

Figure 27: Visual representation of the steps of the algorithm for automatic synchronization of measured and simulated reverse recovery current.

- Normalization of transient data, so that the conditions imposed such as supply voltage and load current do not affect the consistency of the parameter extraction, allowing comparison between different operating conditions.
- Accurate synchronization of switching waveforms, otherwise the errors in $\mathrm{d}i/\mathrm{d}t$ and $\mathrm{d}v/\mathrm{d}t$ will be significant and lead to inaccuracy of the power loss estimates during simulation.
- A method to evaluate simulated characteristics at each voltage or time point of measured data.

Synchronization is needed in order to match both the time scales and the slopes ($\mathrm{d}i/\mathrm{d}t$ and $\mathrm{d}v/\mathrm{d}t$) of transient characteristics. In this work, we are interested only in automatic synchronization techniques. Let us describe the algorithm for the automatic synchronization of diode reverse recovery current waveform:

1. Find the slope of the falling flank of both the measured and simulated current as between 80% of maximum and minimum.
2. Find the slope interval such that the measured characteristic can be interpolated over the simulated characteristic (see figure 27a).

3. Interpolate the measured time over the range of the simulated time.
4. Compute the differences between the interpolated time instants and the simulated time instants (some of them are shown in figure 27b).
5. Compute the average of the differences and apply this shift to simulated data. Simulated and measured data are now synchronized (see figure 27c).

It may be shown that the synchronization strategy we used corresponds to a least squared approach.

Implementation of the synchronization algorithm is robust against rugged simulated characteristics. Voltage waveforms are aligned in a similar way.

The next step is interpolating simulated data over the range of measured data and then subtracting interpolated characteristics from measurements. A possible strategy is to calculate the Residual Sum of Squares (RSS)

$$\mathrm{RSS} = \sum_{n=1}^{N} (y_{\mathrm{interp\_sim}}(n) - y_{\mathrm{meas}}(n))^2$$

where N is the number of samples. The RSS assigns more significance to any large errors between measured and simulated waveforms. It will then focus more on matching the switching losses than the on-state losses. This is because the instantaneous on-state losses are much smaller in magnitude, and become smaller still when squared. Using the Residual Sum of Magnitudes (RSM) rather than the RSS may help to redress this imbalance

$$\mathrm{RSM} = \sum_{n=1}^{N} |y_{\mathrm{interp\_sim}}(n) - y_{\mathrm{meas}}(n)|$$

Many other error estimators may be defined. They will be presented in Appendix A. However, in our experience the RSS and the RSM are the most effective estimators.

1.6.5 *Step 5: parameter optimization*

Optimization techniques rely on finding the minimum of an *objective function*. This is specific to a particular problem, and must be a function of the system parameters. In our case, an objective function is the error between a simulated waveform and its experimental counterpart. The solution we look for is a diode parameter set which minimizes the error between, possibly all, the measured and the simulated characteristics. Therefore the goal is the simultaneous minimization of several objective functions. A problem where p objective functions have to be minimized simultaneously is called a Multiobjective Optimization Problem (MOP). Expressions like multicriteria optimization, multiperformance or vector optimization problem can also be used.

A first approach to multiobjective optimization that can be suited to the parameter extraction problem is a succession of optimizations of one characteristic at a time, until convergence is obtained. Each optimization returns values for a particular subset of the model parameters which have been selected through the sensitivity analysis. Using an expression which will be introduced in Chapter 2, we could

interpret this technique as a *nonscalarizing approach*. An advantage of this method is that it can be carried out with a single objective optimization method, like the Levenberg-Marquardt [79] or the CMA-ES algorithm. Being CMA a stochastic optimizer, it is able to find the global optimum more effectively than a deterministic method. However, since each single optimization tries to minimize only a subset of the model parameters, it is not generally true that the complete procedure could find an optimal solution with respect to all the characteristics. The concept of *optimal solution* in the context of multiobjective optimization will be introduced in Chapter 2. The quality of the solution found by this first approach then depends heavily on the number of waveforms being optimized and on the quality of the initial parameter settings. As the number of characteristics to be optimized is increased, or the quality of parameter initial guesses is decreased, the method will either fail to converge to a meaningful solution or be trapped in whatever local minimum is nearest the given starting guesses.

This method could even fail to converge if the output characteristics, or the correspondent objective functions, are *conflicting*. Two or more waveforms are conflicting if there is no parameter set achieving the minimum error for all of them at the same time. In this situation, the best we can do is to find a set of solutions that are *nondominated* with respect to each other. While moving from one nondominated solution to another, there is always a certain amount of sacrifice in one objective(s) to achieve a certain amount of gain in the other(s). This concept will be better explained in Chapter 2. If the characteristics are conflicting, the method of successive single objective optimizations could jump from one local optimum to another without converging. An example of this behavior is shown in Figure 28.

Another possible parameter extraction methodology is to combine the individual objective functions into a single composite function or move all but one objective to the constraint set and then use a single objective optimizer. In the former case, determination of a single objective is possible with methods such as the weighted sum method, but the problem lies in the proper selection of the weights to characterize the decision-maker's preferences. Compounding this drawback is that scaling amongst objectives is needed and small perturbations in the weights can sometimes lead to quite different solutions. Moreover, if the MOP is nonconvex, there exist solutions that cannot be generated by the weighted sum approach (see Section 2.3.1.1). In the latter case, the problem is that in order to move objectives to the constraint set, a constraining value must be established for each of the former objectives. This can be rather arbitrary. In both cases, an optimization method would return a single solution rather than a set of solutions that can be examined for trade-offs.

A third approach is to determine an entire optimal solution set or a representative subset. As we have already said, if objective functions are conflicting, a solution is considered optimal if it is nondominated, that is, it cannot be improved in any objective without getting worse in at least one other objective. Such a solution is also called a *Pareto optimal* solution.

For the parameter extraction problem finding elements of the Pareto set exactly is impossible due to numerical complexity of resulting optimization problems. For this reason, an approximated description of this set becomes an appealing alter-

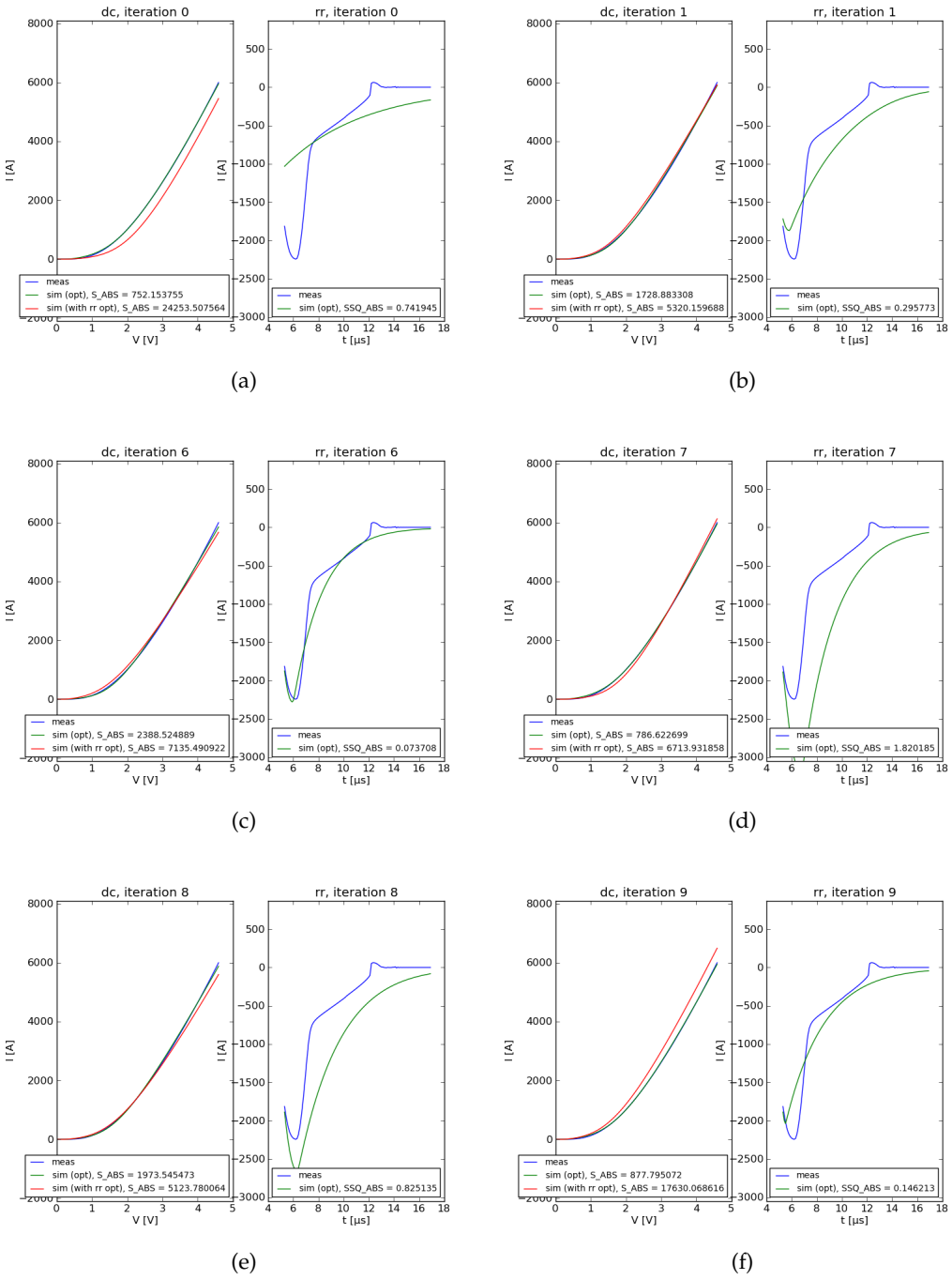


Figure 28: Some iterations of the parameter extraction method for Ma diode model through a succession of single objective optimizations. For each row, the left figure shows DC measured data, the simulated characteristic corresponding to the DC optimal parameter set and the simulated characteristic for the reverse recovery optimal parameter set. The right figure shows measured reverse recovery current and the simulated characteristic given by the corresponding optimal parameter set.

native. Population-based metaheuristics are a particular class of approximation techniques. They have some advantages which include:

- they are not affected by MOP convexity or nonconvexity;
- they return a whole set of approximated Pareto solutions.

In our opinion, *population-based metaheuristics are the best approach to deal with the parameter extraction problem*. They will be presented in Chapter 3. Of course, the No Free Lunch (NFL) Theorem [103] implies that population-based metaheuristics are not a universal robust solution technique for all MOPs. Then, there could be parameter extraction problems which could be more effectively solved with other approaches.

During our internship at ABB Corporate Research Center in Switzerland we developed a Python library for automated parameter extraction of power electronics devices which is mainly based on these techniques. Chapter 4 reports results obtained by using this software for parameter extraction of Extended Lauritzen model.

Let us conclude this chapter with a closer look at the implemented extraction flow for diode compact models (shown in Figure 29):

- Compute first initial parameter estimates.
- Perform sensitivity analysis and define parameter value ranges.
- Check if capacitance characteristic can be fitted independently from DC and reverse recovery waveforms or not. In the first case, capacitance is fitted and the corresponding parameters are kept constant. Afterwards, DC and reverse recovery characteristics are fitted following a successive single optimizations approach or a multiobjective metaheuristic approach. In the second case, all the characteristics have to be fitted jointly.
- At the end, check if reverse recovery current and voltage are conflicting or if a good fit of reverse recovery current ensures a good fit of reverse recovery voltage as well. In the last case, repeat the whole extraction process without considering the reverse recovery voltage.

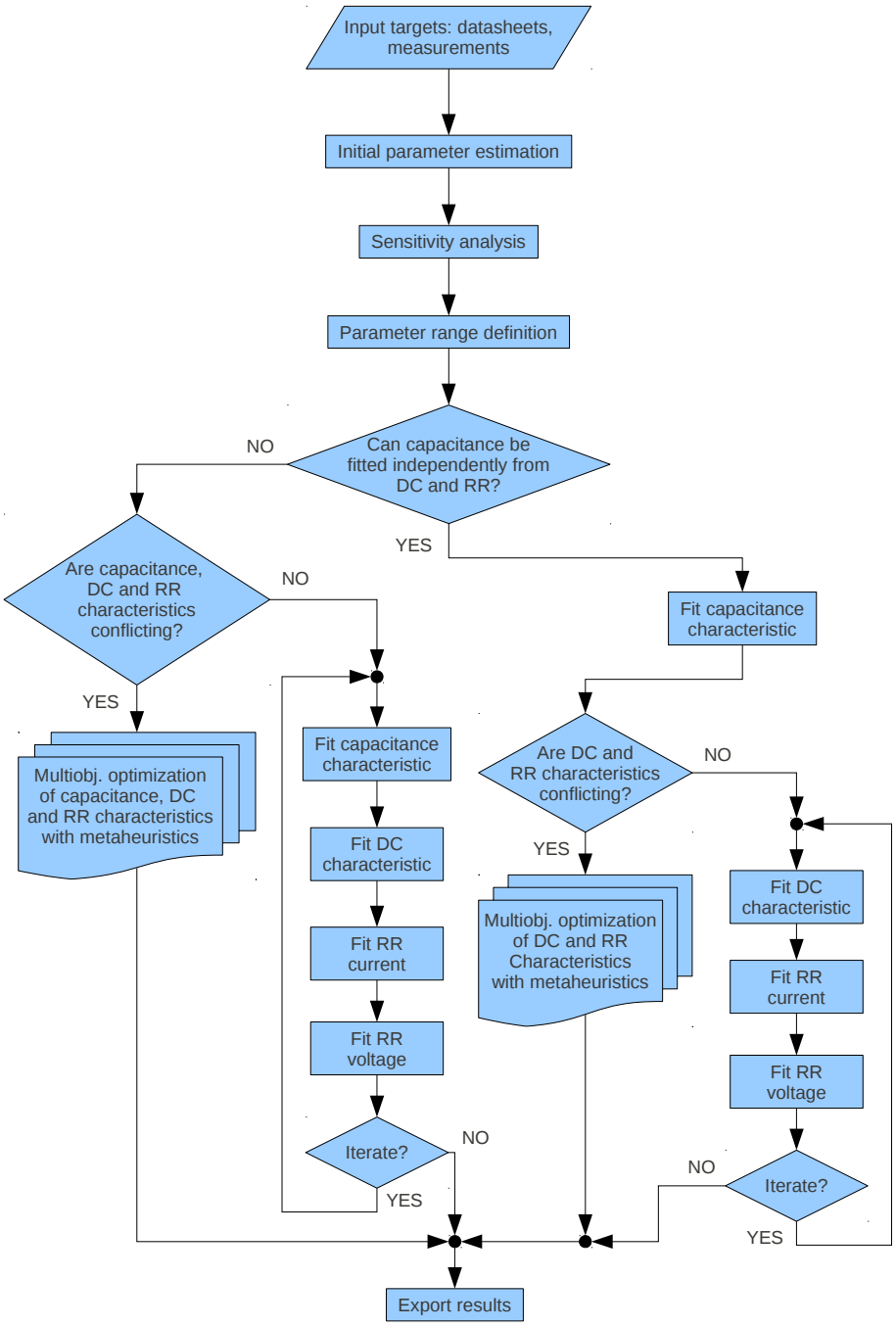


Figure 29: Parameter extraction flow chart for a diode.

2 MULTIOBJECTIVE OPTIMIZATION

The aim of this chapter is to give an overview of multiobjective programming. All these concepts will be used to state the parameter extraction problem as a MOP and to describe the approach that has been developed to solve it. In Sections 2.1 and 2.2 we review theoretical foundations of multiobjective programming. Most of the material in these sections can be found in [30] where the results are presented in a more extensive way. In Section 2.1 we define MOPs and relevant solution concepts. Section 2.2 contains a summary of properties of the solution sets. In Section 2.3, some methods for generating individual elements or subsets of the solution sets are collected.

2.1 PROBLEM FORMULATION AND SOLUTION CONCEPTS

2.1.1 *Problems with multiple objectives*

In order to understand how easily problems with multiple objectives can arise in real world problems, let us consider the following examples of *decision problems*, where a "good" or "best" solution has to be chosen among a set of "alternatives", whose quality is measured according to certain criteria.

Example 2.1. We want to buy a new car and have identified four models we like: an Audi A4, an Alfa Romeo Giulietta, a Great Wall Hover, a Hyundai Genesis. The decision will be made according to price, petrol consumption, and power. We prefer a cheap and powerful car with low petrol consumption. In this case, we face a decision problem with four alternatives (or feasible solutions) and three objectives. The characteristics of the four cars are shown in Table 4 (data is invented).

How do we decide, which of the four cars is the "best" alternative, when the most powerful car is also the one with the highest petrol consumption, so that we cannot buy a car that is cheap as well as powerful and fuel efficient. However, we observe that with any one of the three criteria alone the choice is easy.

		Criteria		
		Price (Euros)	Consumption $\left(\frac{1}{100\,\mathrm{km}}\right)$	Power (kW)
Alternatives	Audi	46 000	14.1	195.0
	Alfa Romeo	26 000	7.8	125.0
	Great Wall	18 000	15.0	93.0
	Hyundai	34 000	15.5	223.0

Table 4: Criteria and alternatives in Example 2.1.

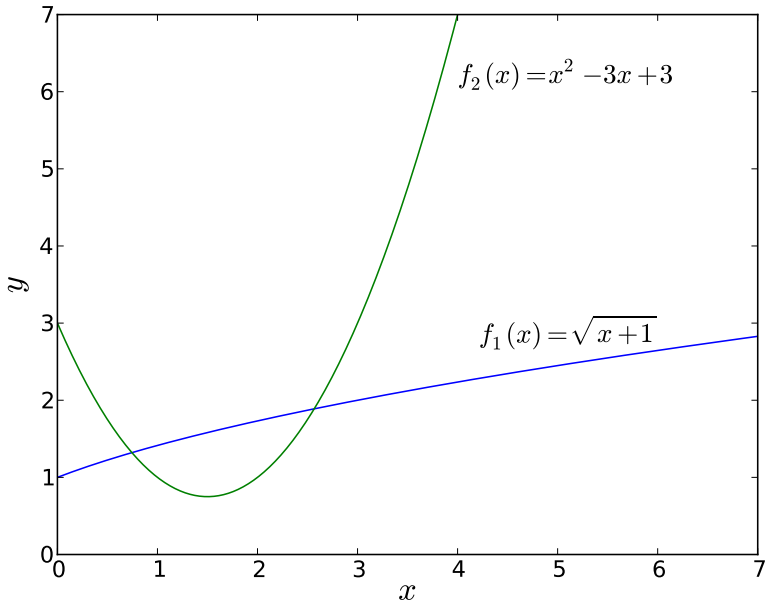


Figure 30: Objective functions of Example 2.3.

Example 2.2. In the assignment of a number of university exams into a limited number of time period there are three main groups of people who are affected by the results of the process:

- administration of the university: usually sets the minimum standards to which the timetables must confirm;
- departments: may have demands for specific classrooms for examinations, may prefer to place large exams early in the timetables, etc;
- students: usually prefer exams to be spread out in time and to have a break between consecutive exams, may be concerned about the order in which exams are scheduled, etc.

Consequently, the quality of a timetable can be assessed from various points of view. Usually the problem is subject to several constraints which can be split into two categories: hard and soft constraints. Hard constraints must not be violated under any circumstances. Soft constraints are desirable but can be violated if necessary. They express the priorities of the three categories presented above.

Criteria are defined with respect to the soft constraints that are imposed on specific problems. Each criterion expresses a measure of the violation of the corresponding constraint. The goal is the simultaneous minimization of all these criteria. However, it is clear that the criteria may be partially or totally conflicting. For example, two exams which should be scheduled immediately before/after each other may have common students, but this is usually not desirable from a student's point of view.

Example 2.3. Consider a mathematical problem with two criteria and one decision variable. The criteria or objective functions, which we want to minimize simultaneously over the nonnegative real line, are

$$f_1(x) = \sqrt{x+1} \quad \text{and} \quad f_2(x) = x^2 - 3x + 3 \tag{2.1}$$

plotted in Figure 30. We want to solve the optimization problem

$$\text{“}\min_{x \geq 0}\text{”}(f_1(x), f_2(x)). \tag{2.2}$$

The question is, what are the “minima” and the “minimizers” in this problem? Note that again, for each function individually the corresponding optimization problem is easy: $x_1 = 0$ and $x_2 = 1.5$ are the (unique) minimizers of f_1 and f_2 on $x \in \mathbb{R} \colon x \geq 0$, respectively.

We see that in Example 2.1, if we move from one alternative to another, there is always a certain amount of sacrifice in one objective(s) to achieve a certain amount of gain in the other(s). That is to say, there is no car which is better than the others in any criterion at the same time. In Example 2.3 all x in $[0, 1.5]$, where one of the functions is increasing, the other is decreasing. Historically, the first reference to address such situations of conflicting objectives is usually attributed to Pareto [81]. In honor of Pareto, these alternatives are today often called *Pareto optimal solutions* of multiple objective optimization problems. The expression *efficient solutions* can also be used.

2.1.2 *Decision Space and Objective Space*

The fundamental notions of decision (or variable) and objective (or criterion) space are informally introduced next.

Let us consider Example 2.1 again, where we consider price and petrol consumption only for the moment. We can illustrate the criterion values in a two-dimensional coordinate system.

From Figure 31 it is easy to see that Great Wall and Alfa Romeo are the efficient choices. For both there is no alternative that is both cheaper and consumes less petrol. In addition, both Hyundai and Audi are more expensive and consume more petrol than Alfa Romeo.

We call $X = \{\text{Great Wall, Alfa Romeo, Hyundai, Audi}\}$ the *feasible set*, or the set of alternatives of the decision problem. The space, of which the feasible set X is a subset, is called the *decision space*.

If we denote price by f_1 and petrol consumption by f_2, then the mappings $f_i \colon X \to \mathbb{R}$ are criteria or objective functions and the optimization problem can be stated mathematically as in Example 2.3

$$\text{“}\min_{x \in X}\text{”}(f_1(x), f_2(x)). \tag{2.3}$$

The image of X under $\boldsymbol{f} = (f_1, f_2)$ is denoted by

$$Y := \boldsymbol{f}(X) := \{\, y \in \mathbb{R}^2 \mid y = \boldsymbol{f}(x) \text{ for some } x \in X \,\}$$

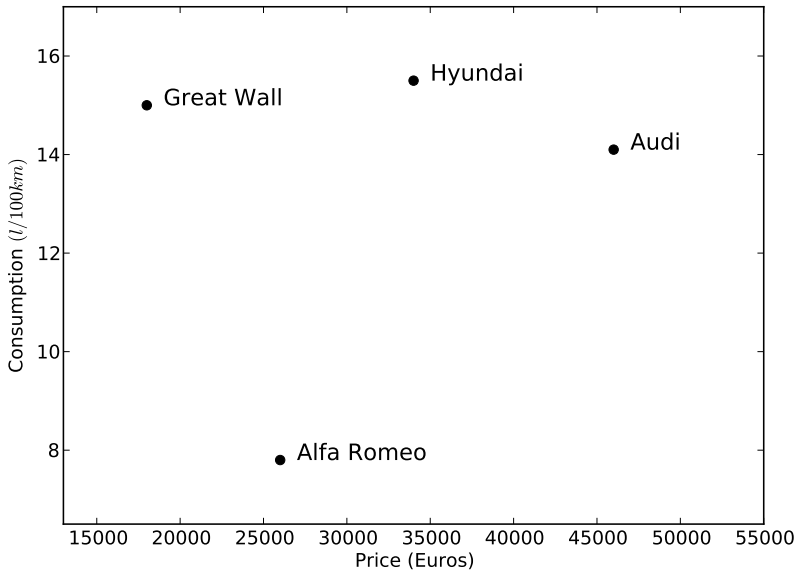


Figure 31: Objective space in Example 2.1.

and referred to as the image of the feasible set, or the feasible set in objective space. The space from which the objective values are taken is called the *objective* (or *criterion*) *space*.

In Example 2.3 the feasible set is

$$X = \{\, x \in \mathbb{R} \mid x \geq 0 \,\} \tag{2.4}$$

and the objective functions are

$$f_1(x) = \sqrt{x+1} \quad \text{and} \quad f_2(x) = x^2 - 3x + 3. \tag{2.5}$$

The decision space is $\mathbb{R}$ because $X \subset \mathbb{R}$. The criterion space is $\mathbb{R}^2$. To obtain the image of the feasible set in criterion space we substitute y_1 for $f_1(x)$ and y_2 for $f_2(x)$ to get $x = y_1^2 - 1$ (solving $y_1 = \sqrt{1+x}$ for x). Therefore we obtain $y_2 = (y_1^2 - 1)^2 - 3(y_1^2 - 1) + 3 = y_1^4 - 5y_1^2 + 7$. The graph of this function (shown in Figure 32) is the analogue of Figure 31 for Example 2.1. Note that $x \geq 0$ translates to $y_1 \geq 1$, so that $Y := \boldsymbol{f}(X)$ is the part of the graph to the right of the vertical line $y_1 = 1$.

Computing the minimum of y_2 as a function of y_1, we see that the efficient solutions $x \in [0, 1.5]$ found before correspond to values of $y_1 = f_1(x) \in [1, \sqrt{2.5}]$ and $y_2 = f_2(x) \in [0.75, 3]$. These points on the graph of $y_2(y_1)$ with $1 \leq y_1 \leq \sqrt{2.5}$ and $0.75 \leq y_2 \leq 3$ are called *nondominated points*.

The right angle attached to the efficient point $\hat{\boldsymbol{y}} = (\hat{y}_1, \hat{y}_2)$ illustrates that there is no other point $\boldsymbol{y} \in \boldsymbol{f}(X)$, $\boldsymbol{y} \neq \hat{\boldsymbol{y}}$ such that $y_1 \leq \hat{y}_1$ and $y_2 \leq \hat{y}_2$. This is true for the image under $\boldsymbol{f}$ of any $x \in [0, 1.5]$. This observation confirms the definition of nondominated points as the image of the set of efficient points under the objective function mapping.

In the examples, we have seen that we will often have many efficient solutions of a multicriteria optimization problem. Obviously, a final choice has to be made among efficient solutions. This aspect of decision making, the support of decision

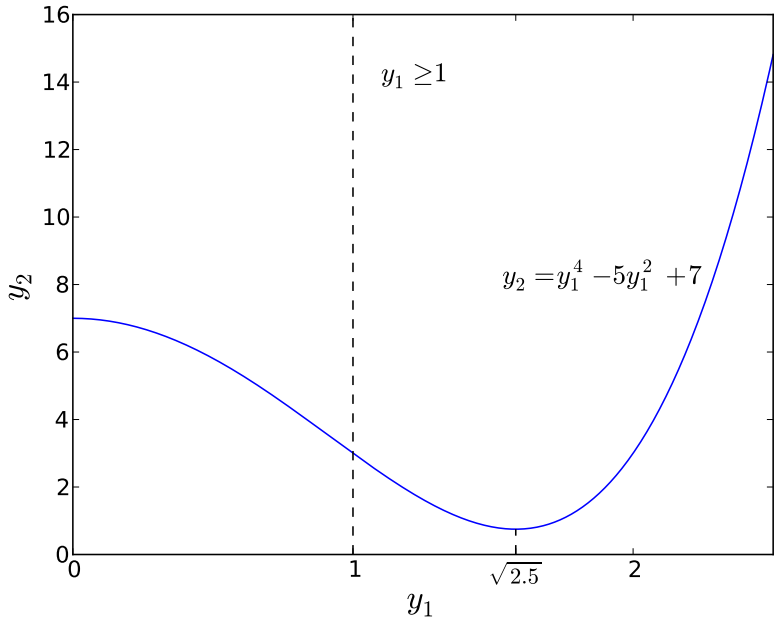


Figure 32: Objective space in Example 2.3.

makers in the selection of a final solution from a set of mathematically "equally optimal" solutions, is often referred to as Multi Criteria Decision Aid (MCDA).

Although finding efficient solutions is the most common form of multicriteria optimization, the field is not limited to that concept. There are other possibilities to cope with multiple conflicting objectives.

2.1.3 *Notions of Optimality*

So far, the minimization in multiobjective optimization problems has been written in quotation marks

$$\text{“}\min_{x \in X}\text{”}(f_1(x), \ldots, f_p(x)) \tag{2.6}$$

because we can associate different interpretations with the "min".

For example we may want to find the set of efficient solutions (Pareto optimal solutions), that is, those solutions that cannot be improved in any objective without getting worse in at least one other objective

$$\{\, x \in X \mid \nexists \hat{x} \in X \text{ subject to (s.t.) } f_k(\hat{x}) \leq f_k(x)\ \forall k = 1, \ldots, p \ \wedge\ \boldsymbol{f}(\hat{x}) \neq \boldsymbol{f}(x) \,\} \tag{2.7}$$

We can imagine situations in which there is a ranking among the objectives. In Example 2.1, price might be more important than petrol consumption, this in turn more important than power. This means that even an extremely good value for petrol consumption cannot compensate for a slightly higher price. Then the criterion vectors $(f_1(x), f_2(x), f_3(x))$ are compared lexicographically (see Table 5 for a definition of the lexicographic order and Section 2.3.2.1 for more on lexicographic optimization) and we want to solve

$$\text{lexmin}_{x \in X}(f_1(x), f_2(x), f_3(x)). \tag{2.8}$$

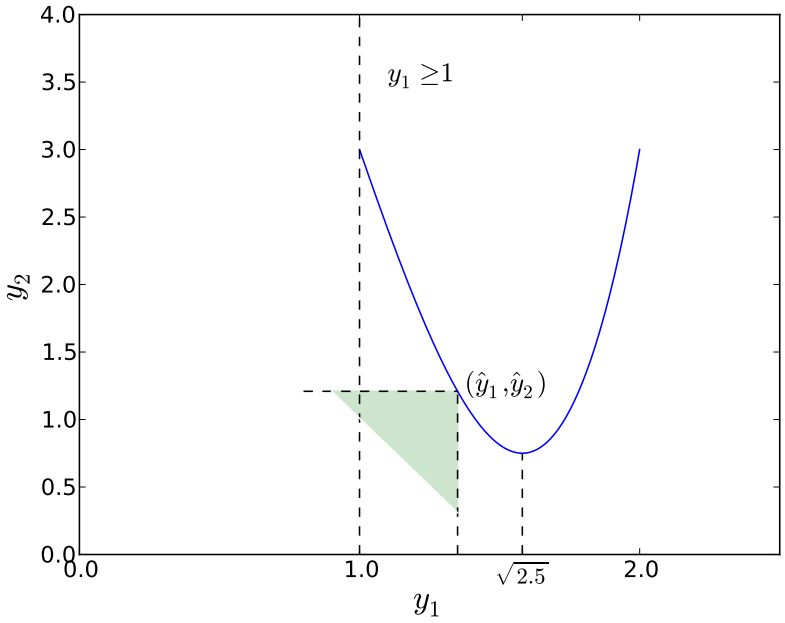


Figure 33: Nondominated points in Example 2.3.

In Example 2.1 we should choose the Great Wall because for this ranking of objectives it is the unique optimal solution (the cheapest).

Let us assume that in Example 2.3 the objective functions measure some negative impacts of a decision (environmental pollution, etc.) to be minimized. We might not want to accept a high value of one criterion for a low value of the other. It is then appropriate to minimize the worst of both objectives. Accordingly we would solve

$$\min_{x \geq 0} \max_{i=1,2} f_i(x). \tag{2.9}$$

This problem is illustrated in Figure 34, where the solid line shows the maximum of f_1 and f_2. The optimal solution of the problem is obtained for $x \approx 0.7445$, see Figure 34.

In the last two examples, we got unique optimal solutions, and there are no incomparable values. And indeed, in the min-max example one could think of this problem as a single objective optimization problem. However, both have to be considered as multicriteria problems, because the multiple objectives are in the formulation of the problems. Thus, in order to define the meaning of "min", we have to define how objective function vectors $(f_1(x), \ldots, f_p(x))$ have to be compared for different alternatives $x \in X$. The different possibilities to do that arise from the fact that for $p \geq 2$ there is no canonical order on $\mathbb{R}^p$ as there is on $\mathbb{R}$. Therefore weaker definitions of orders have to be used.

2.1.4 *Orders and Cones*

In this section we will first introduce binary relations and some of their properties to define several classes of orders. The second main topic is cones, defining sets of nonnegative elements of $\mathbb{R}^p$. They can be used to derive a geometric interpretation

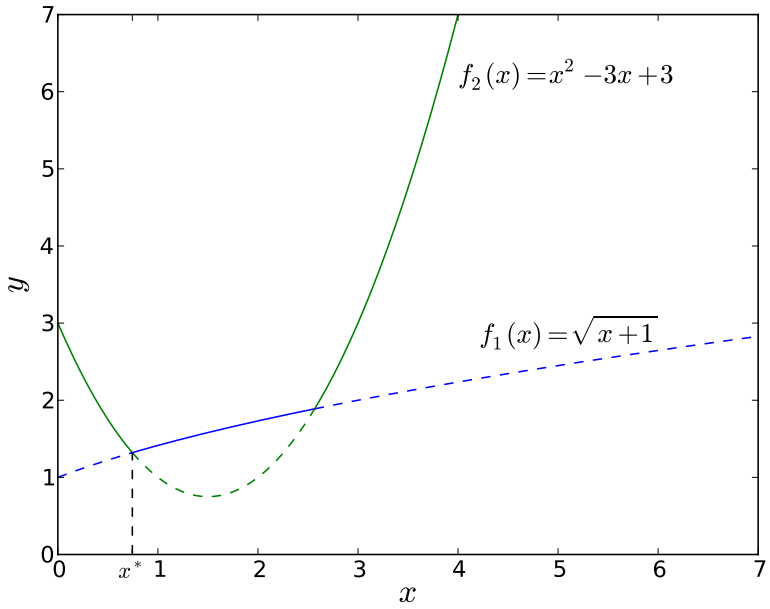


Figure 34: Min-max solutions of Example 2.3.

of orders. An indication of the relationship between orders and cones has already been shown in Figure 33, where we used a cone (the negative orthant of $\mathbb{R}^2$) to confirm that $\hat{\boldsymbol{y}}$ is nondominated.

Let S be any set. A *binary relation* on S is a subset R of $S \times S$.

Definition 2.4. A binary relation R on S is called:

- *reflexive* if $(s,s) \in R \quad \forall s \in S$;
- *irreflexive* if $(s,s) \notin R \quad \forall s \in S$;
- *symmetric* if $(s^1,s^2) \in R \Longrightarrow (s^2,s^1) \in R \quad \forall s^1,s^2 \in S$;
- *asymmetric* if $(s^1,s^2) \in R \Longrightarrow (s^2,s^1) \notin R \quad \forall s^1,s^2 \in S$;
- *antisymmetric* if $(s^1,s^2) \in R \wedge (s^2,s^1) \in R \Longrightarrow s^1 = s^2 \quad \forall s^1,s^2 \in S$;
- *transitive* if $(s^1,s^2) \in R \wedge (s^2,s^3) \in R \Longrightarrow (s^1,s^3) \in R \quad \forall s^1,s^2,s^3 \in S$;
- *negatively transitive* if $(s^1,s^2) \notin R \wedge (s^2,s^3) \notin R \Longrightarrow (s^1,s^3) \notin R \quad \forall s^1,s^2,s^3 \in S$;
- *connected* if $(s^1,s^2) \in R \vee (s^2,s^1) \in R \quad \forall s^1,s^2 \in S$ with $s^1 \neq s^2$;
- *strongly connected* (or *total*) if $(s^1,s^2) \in R \vee (s^2,s^1) \in R \quad \forall s^1,s^2 \in S$.

Definition 2.5. A binary relation R on a set S is:

- an *equivalence relation* if it is reflexive, symmetric and transitive;
- a *preorder* (*quasi-order*) if it is reflexive and transitive.

NOTATION	DEFINITION	NAME
$\boldsymbol{y}^1 \leqq \boldsymbol{y}^2$	$y_k^1 \leqslant y_k^2 \quad k = 1, \dots, p$	weak componentwise order
$\boldsymbol{y}^1 \leq \boldsymbol{y}^2$	$y_k^1 \leqslant y_k^2 \quad k = 1, \dots, p; \boldsymbol{y}^1 \neq \boldsymbol{y}^2$	componentwise order
$\boldsymbol{y}^1 < \boldsymbol{y}^2$	$y_k^1 < y_k^2 \quad k = 1, \dots, p$	strict componentwise order
$\boldsymbol{y}^1 \leq_{\text{lex}} \boldsymbol{y}^2$	$y_{k^*}^1 < y_{k^*}^2$ or $\boldsymbol{y}^1 = \boldsymbol{y}^2$	lexicographic order
$\boldsymbol{y}^1 \leq_{\text{MO}} \boldsymbol{y}^2$	$\max\limits_{k=1,\dots,p} y_k^1 \leq \max\limits_{k=1,\dots,p} y_k^2$	max-order

Table 5: Some orders on $\mathbb{R}^p$.

If R is a preorder the pair (S, R) is called a *preordered set*. In the context of (pre)orders we shall write $s^1 \preceq s^2$ as shorthand for $(s^1, s^2) \in R$ and $s^1 \npreceq s^2$ for $(s^1, s^2) \notin R$ and indifferently refer to the relation R or the relation $\preceq$. According to Ehrgott [30] this notation can be read as "preferred to".

Given any preorder $\preceq$, two other relations can be defined, $\prec$ and $\sim$:

$$s^1 \prec s^2 \iff s^1 \preceq s^2 \wedge s^2 \npreceq s^1 \tag{2.10}$$

$$s^1 \sim s^2 \iff s^1 \preceq s^2 \wedge s^2 \preceq s^1 \tag{2.11}$$

Actually, $\prec$ and $\sim$ can be seen as the strict preference and equivalence (or indifference) relation, respectively, associated with the preference defined by preorder $\preceq$.

The most import classes of relations in multicriteria optimization are partial orders and strict partial orders, which are introduced now.

Definition 2.6. A binary relation $\preceq$ is called

- *partial order* if it is reflexive, transitive and antisymmetric;
- *strict partial order* if it is asymmetric and transitive.

Throughout this work, we use several orders on the Euclidean space $\mathbb{R}^p$. These notations are not unique in multiobjective literature therefore they should always be checked when consulting another source. Let

$$\boldsymbol{y}^1 = (y_1^1, \dots, y_p^1), \boldsymbol{y}^2 = (y_1^2, \dots, y_p^2) \in \mathbb{R}^p,$$

and if $\boldsymbol{y}^1 \neq \boldsymbol{y}^2$ let $k^* := \min \{ k \mid y_k^1 \neq y_k^2 \}$. Notations and names in Table 5 will be used for the most common ((strict) partial) orders on $\mathbb{R}^p$ appearing in this work.

With the (weak, strict) componentwise orders, we define subsets of $\mathbb{R}^p$ as follows:

- $\mathbb{R}^p_{\geqq} := \{ \boldsymbol{y} \in \mathbb{R}^p \mid \boldsymbol{y} \geqq \boldsymbol{0} \}$, the nonnegative orthant of $\mathbb{R}^p$;
- $\mathbb{R}^p_{\geq} := \{ \boldsymbol{y} \in \mathbb{R}^p \mid \boldsymbol{y} \geq \boldsymbol{0} \} = \mathbb{R}^p_{\geqq} \setminus \{\boldsymbol{0}\}$;
- $\mathbb{R}^p_{>} := \{ \boldsymbol{y} \in \mathbb{R}^p \mid \boldsymbol{y} > \boldsymbol{0} \} = \operatorname{int} \mathbb{R}^p_{\geqq}$, the positive orthant of $\mathbb{R}^p$.

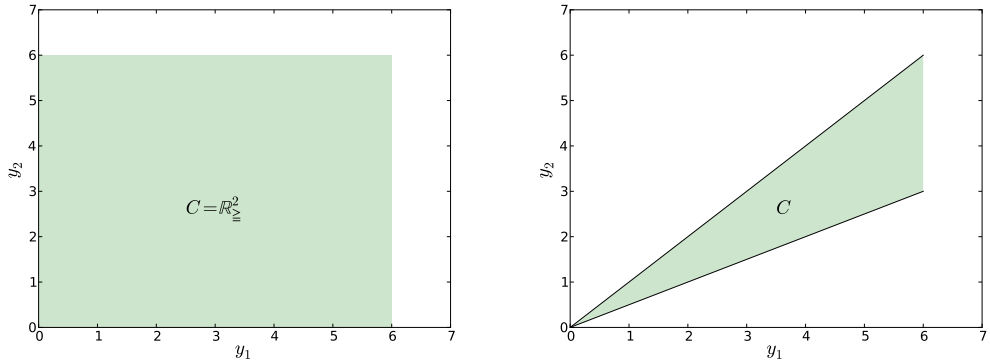
Note that for $p = 1$ we have $\mathbb{R}_{\geq} = \mathbb{R}_{>}$. Throughout this work, for $S \subseteq \mathbb{R}^n$ or $S \subseteq \mathbb{R}^p$

- $\operatorname{int}(S)$ is the interior of S.

The notion of cone in $\mathbb{R}^p$ ($\mathbb{R}^2$ for purpose of illustration) can be used to derive a geometric interpretation of properties of orders. Since (partial) orders can be used to define "minimization" or "maximization", this interpretation makes it possible to analyze multicriteria optimization problems geometrically [30].

Definition 2.7. A subset $C \subseteq \mathbb{R}^p$ is called a *cone*, if $\alpha \boldsymbol{d} \in C \; \forall \boldsymbol{d} \in C$ and $\forall \alpha \in \mathbb{R}$, $\alpha > 0$.

Example 2.8. Figure 35a shows the cone $C = \left\{ \boldsymbol{d} \in \mathbb{R}^2 \;\middle|\; d_k \geq 0, k = 1,2 \right\} = \mathbb{R}^2_{\overset{\leq}{=}}$. This is the cone of nonnegative elements of the weak componentwise order. Figure 35b shows a smaller cone $C \subset \mathbb{R}^2_{\overset{\leq}{=}}$.



(a) (b)

Figure 35: Illustration of two cones.

Let $S, S_1, S_2 \subset \mathbb{R}^p$ and $\alpha \in \mathbb{R}$. We denote by:

$$\alpha S := \{ \alpha s \mid s \in S \} \tag{2.12}$$

$$S_1 + S_2 := \left\{ s^1 + s^2 \;\middle|\; s^1 \in S_1, s^2 \in S_2 \right\}. \tag{2.13}$$

In particular, $-S = \{ -s \mid s \in S \}$. If $S_1 = \{s\}$ is a singleton, we also write $s + S_2$ instead of $\{s\} + S_2$. These notations do not involve any set arithmetic, e.g. $2S \neq S + S$ in general.

Definition 2.9. A cone $C \subseteq \mathbb{R}^p$ is called:

- *nontrivial* or *proper* if $C \neq \emptyset$ and $C \neq \mathbb{R}^p$;
- *convex* if $\alpha \boldsymbol{d}^1 + (1-\alpha)\boldsymbol{d}^2 \in C \; \forall \boldsymbol{d}^1, \boldsymbol{d}^2 \in C$ and $\alpha \in (0,1)$;
- *pointed* if for $\boldsymbol{d} \in C, \boldsymbol{d} \neq 0, -\boldsymbol{d} \notin C$, i.e., $C \cap (-C) \subseteq \{0\}$.

Given an order relation R on $\mathbb{R}^p$, we can define a set

$$C_R := \left\{ \boldsymbol{y}^2 - \boldsymbol{y}^1 \;\middle|\; \boldsymbol{y}^1 R \boldsymbol{y}^2 \right\}. \tag{2.14}$$

Proposition 2.10. *Let R be compatible with scalar multiplication, i.e., $\forall \; (\boldsymbol{y}^1, \boldsymbol{y}^2) \in R$ and $\alpha > 0$ it holds that $(\alpha \boldsymbol{y}^1, \alpha \boldsymbol{y}^2) \in R$. Then C_R defined in (2.14) is a cone.*

Example 2.11. Let us consider the weak componentwise order on $\mathbb{R}^p$. Here $\boldsymbol{y}^1 \leqq \boldsymbol{y}^2$ if and only if $y_k^1 \leq y_k^2 \; \forall \; k = 1, \ldots, p$ or $y_k^2 - y_k^1 \geq 0 \; \forall \; k = 1, \ldots, p$. Therefore

$$C_{\leqq} = \{ \boldsymbol{d} \in \mathbb{R}^p \mid d_k \geq 0, k = 1, \ldots, p \} = \mathbb{R}^p_{\leqq}.$$

Let us consider Definition (2.14) with $\boldsymbol{y}^1 \in \mathbb{R}^p$ fixed, i.e.,

$$C_R(\boldsymbol{y}^1) = \left\{ \boldsymbol{y}^2 - \boldsymbol{y}^1 \;\middle|\; \boldsymbol{y}^1 R \boldsymbol{y}^2 \right\}.$$

If R is an order relation, $\boldsymbol{y}^1 + C_R(\boldsymbol{y}^1)$ is the set of elements of $\mathbb{R}^p$ that $\boldsymbol{y}^1$ has preference over.

The set $C_R(\boldsymbol{y})$ is the same for all $\boldsymbol{y} \in \mathbb{R}^p$ if the order relation R is compatible with addition, i.e. if $(\boldsymbol{y}^1 + \boldsymbol{z}, \boldsymbol{y}^2 + \boldsymbol{z}) \in R \; \forall \; \boldsymbol{z} \in \mathbb{R}^p$ and $\forall \; (\boldsymbol{y}^1, \boldsymbol{y}^2) \in R$.

Proposition 2.12. *If R is compatible with addition and $\boldsymbol{d} \in C_R$, then $0R\boldsymbol{d}$.*

Proposition 2.12 means that if R is compatible with addition, the sets $C_R(\boldsymbol{y}), \boldsymbol{y} \in \mathbb{R}^p$, do not depend on $\boldsymbol{y}$.

Theorem 2.13. *Let R be a binary relation on $\mathbb{R}^p$ which is compatible with scalar multiplication and addition. Then the following statements hold:*

- *$0 \in C_R$ if and only if R is reflexive;*
- *C_R is pointed if and only if R is symmetric;*
- *C_R is convex if and only if R is transitive.*

Example 2.14. The weak componentwise order $\leqq$ is compatible with addition and scalar multiplication. $C_{\leqq} = \mathbb{R}^p_{\leqq}$ contains 0, is pointed, and convex.

The max-order $\leq_{\text{MO}}$ is compatible with scalar multiplication, but not with addition (e.g. $(-3, 2) \leq_{\text{MO}} (3, 1)$, but this relation is reversed when adding $(0, 3)$). Furthermore, $\leq_{\text{MO}}$ is reflexive, transitive, but not antisymmetric (e.g. $(1, 0) \leq_{\text{MO}} (1, 1)$ and $(1, 1) \leq_{\text{MO}} (1, 0)$).

We have defined cone C_R given relation R. We can also use a cone to define an order relation. Let C be a cone. Define R_C by

$$\boldsymbol{y}^1 R_C \boldsymbol{y}^2 \iff \boldsymbol{y}^2 - \boldsymbol{y}^1 \in C. \tag{2.15}$$

Proposition 2.15. *Let C be a cone. Then R_C defined in (2.15) is compatible with scalar multiplication and addition in $\mathbb{R}^p$.*

Theorem 2.16. *Let C be a cone and let R_C be as defined in (2.15). Then the following statements hold:*

- *R_C is reflexive if and only if $0 \in C$;*
- *R_C is antisymmetric if and only if C is pointed;*
- *R_C is transitive if and only if C is convex.*

Theorems 2.13 and 2.16 show equivalence of some partial orders and pointed convex cones containing $\mathbf{0}$. Thanks to these two results we can give a geometric interpretation to multiobjective optimization problems.

2.1.5 *Multiobjective optimal solutions*

By the choice of an order $\preceq$ on $\mathbb{R}^p$ we can define the meaning of "min" in the problem formulation

$$\text{“}\min_{x\in X}\text{”}\boldsymbol{f}(x) = \text{“}\min_{x\in X}\text{”}(f_1(x),\ldots,f_p(x)). \tag{2.16}$$

With the multiple objective functions we can evaluate objective value vectors

$$(f_1(x),\ldots,f_p(x)).$$

However, we have seen that these vectors $\boldsymbol{y} = \boldsymbol{f}(x), x \in X$ are not always compared in objective space, i.e., $\mathbb{R}^p$, directly.

In Example 2.3 we have formulated the optimization problem

$$\min_{x\in X}\max_{i=1,2} f_i(x). \tag{2.17}$$

That is, we have used a mapping $\vartheta\colon \mathbb{R}^2 \to \mathbb{R}$ from objective space $\mathbb{R}^2$ to $\mathbb{R}$, where the min in (2.17) is defined by the canonical order on $\mathbb{R}$.

In general, the objective function vectors are mapped from $\mathbb{R}^p$ to an ordered space, e.g. $(\mathbb{R}^P, \preceq)$, where comparisons are made using the order relation $\preceq$. This mapping is called the *model map*.

Without loss of generality, we can assume that every multiobjective optimization problem can be stated as (2.16) since maximizing an objective function $g(x)$ is equivalent to minimizing $-g(x)$. Then a MOP can be formally stated as a tuple

$$((X, \mathbb{R}^p, \boldsymbol{f}), \vartheta, (\mathbb{R}^P, \preceq)),$$

where:

- X is the feasible set;
- $\mathbb{R}^p$ the objective space;
- $\boldsymbol{f} = (f_1,\ldots,f_p)\colon X \to \mathbb{R}^p$ is the objective function vector;
- $(\mathbb{R}^P, \preceq)$ is the ordered set;
- ϑ is the model map.

Example 2.17. Let us look at a problem of finding efficient solutions

$$\min_{x\geq 0}(\sqrt{x+1}, x^2 - 4x + 1). \tag{2.18}$$

Here the feasible set is $X = \{\, x \mid x \geq 0 \,\} = \mathbb{R}_{\geqq}$, the objective function vector is $\boldsymbol{f} = (f_1, f_2) = (\sqrt{x+1}, x^2 - 4x + 1)$, and the objective space is $\mathbb{R}^p = \mathbb{R}^2$. Since we compare objective function vectors componentwise, the model map is given by $\vartheta(\boldsymbol{y}) = \boldsymbol{y}$ and denoted "id", the *identity mapping*, henceforth. The ordered set is then $(\mathbb{R}^P, \preceq) = (\mathbb{R}^2, \leq)$. The problem (2.18) is classified as

$$((\mathbb{R}_{\geqq}, \mathbb{R}^2, \boldsymbol{f}), \mathrm{id}, (\mathbb{R}^2, \leq)). \tag{2.19}$$

Example 2.18. If we have a ranking of objectives as described in Example 2.1 in Section 2.1.1, we compare objective vectors lexicographically. See Table 5 for a definition of lexicographic order. In Example 2.1

$$X = \{\text{Great Wall, Alfa Romeo, Hyundai, Audi}\}$$

is the set of alternatives (feasible set), f_1 is price, f_2 is petrol consumption, and f_3 is power. We define $\vartheta(\boldsymbol{y}) = (y_1, y_2, -y_3)$ (note that more power is preferred to less). The problem is then classified as

$$((X, \mathbb{R}^3, \boldsymbol{f}), \vartheta, (\mathbb{R}^3, \leq_{\text{lex}})). \tag{2.20}$$

Definition 2.19. A feasible solution $x^* \in X$ is called an *optimal solution* of a multiobjective optimization problem

$$((X, \mathbb{R}^p, \boldsymbol{f}), \vartheta, (\mathbb{R}^P, \preceq))$$

if there is no $x \in X, x \neq x^*$ such that

$$\vartheta(\boldsymbol{f}(x)) \preceq \vartheta(\boldsymbol{f}(x^*)). \tag{2.21}$$

For an optimal solution x^*, $\vartheta(\boldsymbol{f}(x^*))$ is called an *optimal value* of the MOP.

Since we are often dealing with orders which are not total, a positive definition of optimality, like $\vartheta(\boldsymbol{f}(x^*)) \preceq \vartheta(\boldsymbol{f}(x)) \; \forall \, x \in X$, is not possible in general. Moreover, for specific choices of ϑ and $(\mathbb{R}^P, \preceq)$, specific names for optimal solutions and values are commonly used, such as efficient solutions or lexicographically optimal solutions.

We now check Definition 2.19 with Examples 2.17 and 2.18.

Example 2.20. With the problem $((\mathbb{R}_{\geqq}, \mathbb{R}^2, \boldsymbol{f}), \mathrm{id}, (\mathbb{R}^2, \leq))$ the optimality definitions reads: there is no $x \in X, x \neq x^*$, such that $\boldsymbol{f}(x) \leq \boldsymbol{f}(x^*)$, i.e., $f_k(x) \leq f_k(x^*) \, \forall \, k = 1, \ldots, p$, and $\boldsymbol{f}(x) \neq \boldsymbol{f}(x^*)$. This is efficiency as we know it.

Example 2.21. For $((X, \mathbb{R}^3, \boldsymbol{f}), \vartheta, (\mathbb{R}^3, \leq_{\text{lex}}))$ with $\vartheta(\boldsymbol{y}) = (y_1, y_2, -y_3)$, $x^* \in X$ is an optimal solution if there is no $x \in X, x \neq x^*$, such that

$$(f_1(x), f_2(x), -f_3(x)) \leq_{\text{lex}} (f_1(x^*), f_2(x^*), -f_3(x^*)). \tag{2.22}$$

2.1.6 *Efficiency and Nondominance*

Here we focus on multiobjective optimization problems of finding efficient solutions

$$\min_{x \in X} (f_1(x), \ldots, f_p(x)) \tag{2.23}$$

The model map is given by the identity mapping introduced in Section 2.1.5. The image of the feasible set X under the objective function mapping $\boldsymbol{f}$ is denoted as $Y := \boldsymbol{f}(X)$.

Definition 2.22. A feasible solution $\hat{x} \in X$ is called *efficient* or *Pareto optimal* if there is no other $x \in X$ such that $\boldsymbol{f}(x) \leq \boldsymbol{f}(\hat{x})$. If $\hat{x}$ is efficient, $\boldsymbol{f}(\hat{x})$ is called *nondominated point*. If $x^1, x^2 \in X$ and $\boldsymbol{f}(x^1) \leq \boldsymbol{f}(x^2)$ we say x^1 *dominates* x^2 and $\boldsymbol{f}(x^1)$ *dominates* $\boldsymbol{f}(x_2)$. The set of all efficient solutions $\hat{x} \in X$ is denoted X_E and called the *efficient set* or the *Pareto optimal set*. The set of all nondominated points $\hat{\boldsymbol{y}} = \boldsymbol{f}(\hat{x}) \in Y$, where $\hat{x} \in X_E$, is denoted Y_N and called the *nondominated set* or the *Pareto front*.

In Chapter 3, the terms *efficient* and *nondominated* are equally referred to elements of X and Y. However, it should be clear from the context what we are talking about.

Several other, equivalent, definitions of efficiency are frequently used. In particular, $\hat{x}$ is efficient if:

1. there is no $x \in X$ such that $f_k(x) \leq f_k(\hat{x})$ for $k = 1, \ldots, p$ and $f_i(x) < f_i(\hat{x})$ for some $i \in \{1, \ldots, k\}$;
2. there is no $x \in X$ such that $\boldsymbol{f}(x) - \boldsymbol{f}(\hat{x}) \in -\mathbb{R}^p_{\leqq} \setminus \{0\}$;
3. $\boldsymbol{f}(x) - \boldsymbol{f}(\hat{x}) \in \mathbb{R}^p \setminus \left\{-\mathbb{R}^p_{\leqq} \setminus \{0\}\right\} \quad \forall\, x \in X$;
4. $\boldsymbol{f}(X) \cap \left(\boldsymbol{f}(\hat{x}) - \mathbb{R}^p_{\leqq}\right) = \{\boldsymbol{f}(\hat{x})\}$;
5. there is no $\boldsymbol{f}(x) \in \boldsymbol{f}(X) \setminus \{\boldsymbol{f}(\hat{x})\}$ with $\boldsymbol{f}(x) \in \boldsymbol{f}(\hat{x}) - \mathbb{R}^p_{\leqq}$;
6. $\boldsymbol{f}(x) \leqq \boldsymbol{f}(\hat{x})$ for some $x \in X$ implies $\boldsymbol{f}(x) = \boldsymbol{f}(\hat{x})$.

With the exception of the last, these definitions can be illustrated graphically. Definition 2.22 and equivalent definitions 1., 4., and 5. consider $\boldsymbol{f}(\hat{x})$ and check for images of feasible solutions to the left and below (in direction of $-\mathbb{R}^p_{\leqq}$) of that point. See Figure 36a. In equivalent definitions 2. and 3., through $\boldsymbol{f}(x) - \boldsymbol{f}(\hat{x})$, the set $Y = \boldsymbol{f}(X)$ is translated so that the origin coincides with $\boldsymbol{f}(\hat{x})$, and the intersection of the translated set Y with the negative orthant is checked. This intersection contains only $\boldsymbol{f}(\hat{x})$ if $\hat{x}$ is efficient. See Figure 36b.

Nondominated points are defined by the componentwise order on $\mathbb{R}^p$. When we use the the weak and strict componentwise order instead, we obtain definitions of strictly and weakly nondominated points, respectively.

Definition 2.23. A feasible solution $\hat{x} \in X$ is called *weakly efficient* or *weakly Pareto optimal* if there is no $x \in X$ such that $\boldsymbol{f}(x) < \boldsymbol{f}(\hat{x})$, i.e. $f_k(x) < f_k(\hat{x})\, \forall\, k = 1, \ldots, p$. The point $\hat{\boldsymbol{y}} = \boldsymbol{f}(\hat{x})$ is then called *weakly nondominated*.

A feasible solution $\hat{x} \in X$ is called *strictly efficient* or *strictly Pareto optimal* if there is no $x \in X, x \neq \hat{x}$ such that $\boldsymbol{f}(x) \leqq \boldsymbol{f}(\hat{x})$. The weakly (strictly) efficient and nondominated sets are denoted $X_{wE}(X_{sE})$ and Y_{wE}, respectively.

From the definitions it is clear that

$$Y_N \subset Y_{wN} \tag{2.24}$$

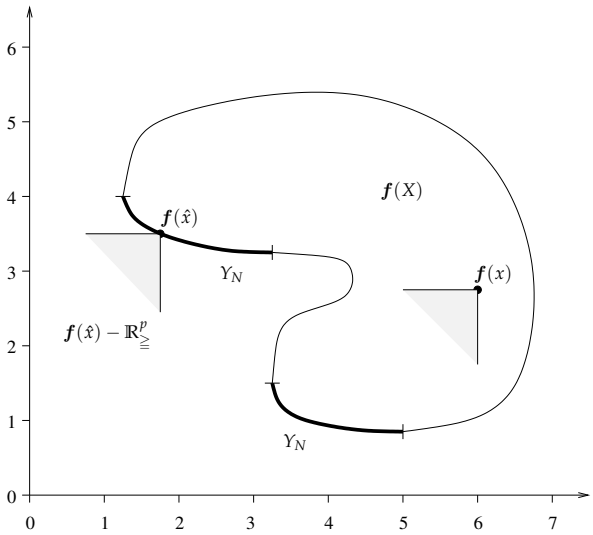
and

$$X_{sE} \subset X_E \subset X_{wE} \tag{2.25}$$

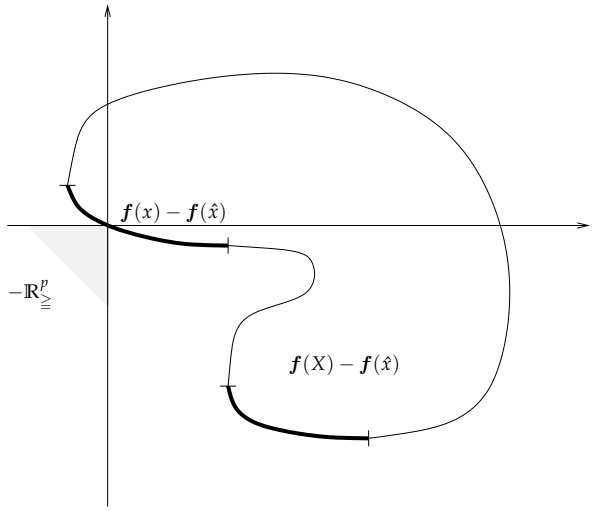
In Figure 37 nondominated and weakly nondominated points are compared.

As in the case of efficiency, weak efficiency has several equivalent definitions. We mention only two. A feasible solution $\hat{x} \in X$ is weakly efficient if and only if

1. there is no $x \in X$ such that $\boldsymbol{f}(\hat{x}) - \boldsymbol{f}(x) \in \operatorname{int} \mathbb{R}^p_{\leqq} = \mathbb{R}^p_{>}$;
2. $(\boldsymbol{f}(\hat{x}) - \mathbb{R}^p_{>}) \cap Y = \emptyset$.



(a) Definitions 1., 4., and 5.



(b) Definitions 2. and 3.

Figure 36: Illustration of definitions of efficient solutions.

There is no such concept as strict nondominance for sets $Y \subset \mathbb{R}^p$. By definition, strict efficiency prohibits solutions x^1, x^2 with $\boldsymbol{f}(x^1) = \boldsymbol{f}(x^2)$, i.e. strict efficiency resembles unique optimal solutions in scalar optimization

$$\hat{x} \in X_{sE} \iff \hat{x} \in X_E \wedge |\{ x \mid \boldsymbol{f}(x) = \boldsymbol{f}(\hat{x}) \}| = 1. \tag{2.26}$$

According to Definition 2.22, an efficient solution does not allow improvement of one objective function while retaining the same values on the others. Improvement of some criterion can only be obtained at the expense of the deterioration of at least one other criterion. These trade-offs among criteria can be measured by computing the increase in objective f_i, say, per unit decrease in objective f_j. In some situations such trade-offs can be unbounded. We give an example below and introduce Geoffrion's definition of efficient solutions with bounded trade-offs, so called *properly efficient solutions*.

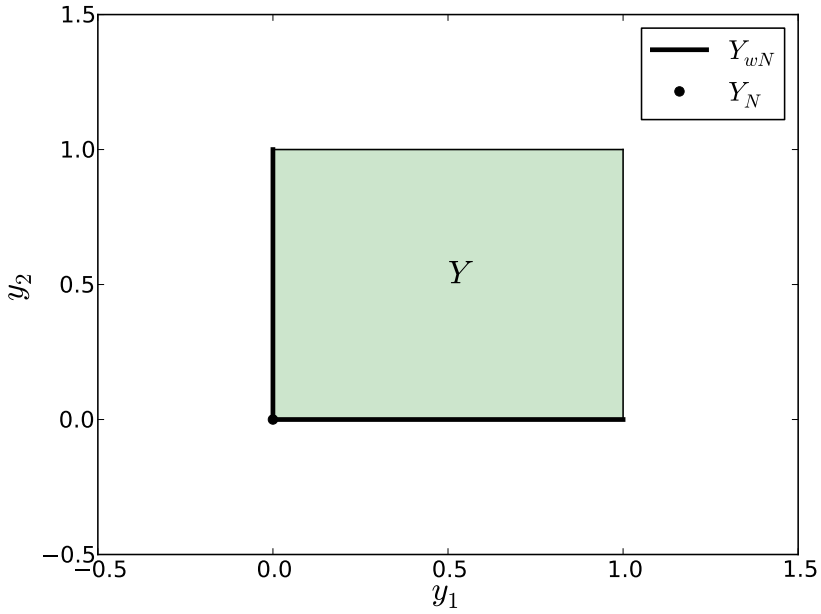


Figure 37: Nondominated and weakly nondominated points.

Example 2.24. Let the feasible set in decision and objective space be given by

$$X = \left\{ (x_1, x_2) \in \mathbb{R}^2 \;\middle|\; (x_1 - 1)^2 + (x_2 - 1)^2 \leq 1,\, 0 \leq x_1, x_2 \leq 1 \right\}, \tag{2.27}$$

and $Y = X$ as shown in Figure 38.

We have $Y_N = \left\{ (y_1, y_2) \in Y \;\middle|\; (y_1 - 1)^2 + (y_2 - 1)^2 = 1 \right\}$. The closer $\hat{\boldsymbol{y}}$ is moved towards $(1, 0)$, the larger an increase of y_1 is necessary to achieve a unit decrease in y_2. In the limit, an infinite increase of y_1 is needed to obtain a unit decrease in y_2.

Definition 2.25 (Geoffrion (1968))**.** A feasible solution $\hat{x} \in X$ is called *properly efficient*, if it is efficient and if there is a real number $M > 0$ such that $\forall\, i$ and $x \in X$ satisfying $f_i(x) < f_i(\hat{x})$ there exists an index j such that $f_j(\hat{x}) < f_j(x)$ and

$$\frac{f_i(\hat{x}) - f_i(x)}{f_j(x) - f_j(\hat{x})} \leq M. \tag{2.28}$$

The corresponding point $\hat{\boldsymbol{y}} = \boldsymbol{f}(\hat{x})$ is called *properly nondominated*.

According to Definition 2.25 properly efficient solutions therefore are those efficient solutions that have bounded trade-offs between the objectives.

Example 2.26. In Example 2.24 consider the solution $\hat{x} = (1, 0)$. We show that $\hat{x}$ is not properly efficient. We have to prove that for all $M > 0$ there is an index $i \in \{1, 2\}$ and some $x \in X$ with $f_i(x) < f_i(\hat{x})$ such that

$$\frac{f_i(\hat{x}) - f_i(x)}{f_j(x) - f_j(\hat{x})} > M \tag{2.29}$$

$\forall\, j \in \{1, 2\}$ with $f_j(x) > f_j(\hat{x})$.

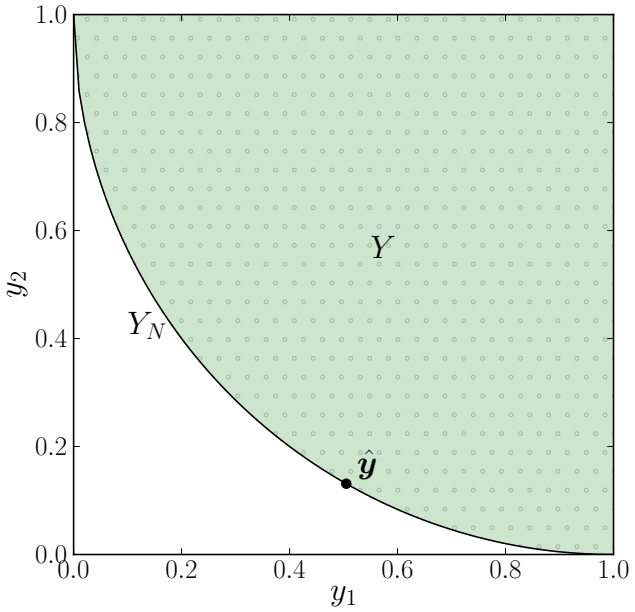


Figure 38: Properly nondominated point $\hat{\boldsymbol{y}}$.

Let $i = 1$ and choose $\boldsymbol{x}^\varepsilon$ with $x_1^\varepsilon = 1 - \varepsilon, 0 < \varepsilon < 1$ and $x_2^\varepsilon = 1 - \sqrt{1 - \varepsilon^2}$, i.e. $\boldsymbol{x}^\varepsilon$ is efficient because $(x_1^\varepsilon - 1)^2 + (x_2^\varepsilon - 1)^2 = 1$. Since $\boldsymbol{x}^\varepsilon \in X, x_1^\varepsilon < \hat{x}_1$ and $x_2^\varepsilon > \hat{x}_2$ we have $i = 1, j = 2$. Thus

$$\frac{f_i(\hat{\boldsymbol{x}}) - f_i(\boldsymbol{x}^\varepsilon)}{f_j(\boldsymbol{x}^\varepsilon) - f_j(\hat{\boldsymbol{x}})} = \frac{1 - (1 - \varepsilon)}{1 - \sqrt{1 - \varepsilon^2}} = \frac{\varepsilon}{1 - \sqrt{1 - \varepsilon^2}} \xrightarrow{\varepsilon \to 0} \infty. \tag{2.30}$$

The set of all properly efficient solutions and properly nondominated outcomes (in the sense of Geoffrion) are denoted by X_{pE} and Y_{pN}, respectively.

Some further definitions of proper efficiency have been given by Borwein, Benson, and Kuhn and Tucker [30].

Let $y_k^I := \min \{ f_k(x) \mid x \in X \}$ be the (global) minimum of $f_k(x), k = 1, \ldots, p$. The point $\boldsymbol{y}^I \in \mathbb{R}^p, \boldsymbol{y}^I = (y_1^I, \ldots, y_p^I)$ is called the *ideal point* of the MOP.

The point $\boldsymbol{y}^N$ with $y_k^N := \max \{ f_k(x) \mid x \in X_E \}$ is called the *nadir point* of the MOP.

The ideal and nadir points for a nonconvex problem are shown in Figure 39. We have $y_k^I \leq y_k$ and $y_k \leq y_k^N$ for any $\boldsymbol{y} \in Y_N$. $\boldsymbol{y}^I$ and $\boldsymbol{y}^N$ are tight lower and upper bounds on the efficient set. These points give an indication of the range of the values which nondominated points can attain. Since the ideal point is found by solving p single objective optimization problems its computations can be considered easy. On the other hand, the computation of $\boldsymbol{y}^N$ involves optimization over the efficient set, a very difficult problem.

2.2 PROPERTIES OF THE SOLUTION SETS

In this section some properties of nondominated and efficient sets are presented in order to enhance an understanding of the concepts of nondominance. Proofs can be found in [30].

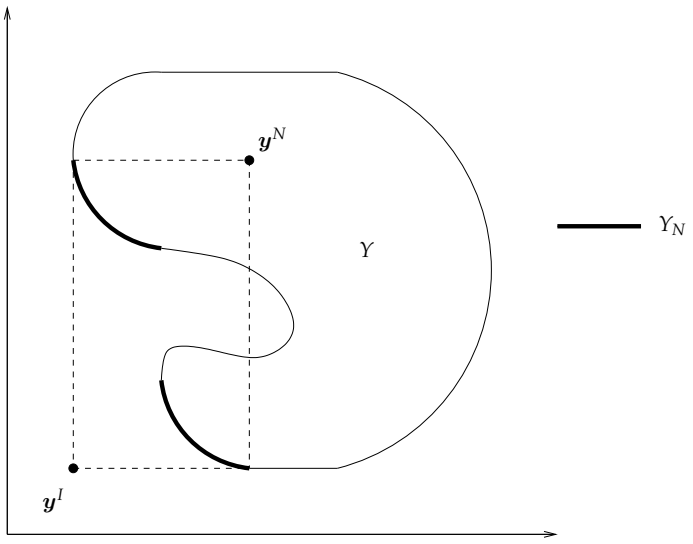


Figure 39: Efficient set, ideal, and nadir point.

So let $Y \subset \mathbb{R}^p$. Let $Y_N = \{ y \in Y \mid \nexists y^* \in Y \text{ such that } y^* \leq y \}$. In particular $Y_N \subset Y$.

Proposition 2.27. $Y_N = \left(Y + \mathbb{R}^p_{\leqq}\right)_N$.

According to Proposition 2.27, nondominated points are located in the "lower left" part of Y: adding $\mathbb{R}^p_{\leqq}$ to Y does not change the nondominated set. This proposition is shown in Figure 40.

A second result is that efficient points must belong to the boundary of Y, ∂Y.

Proposition 2.28. $Y_N \subset \partial Y$.

The next results concern the nondominated set of the (Minkowski) sum of two sets and a set multiplied by a positive scalar.

Proposition 2.29. $(Y_1 + Y_2)_N \subset (Y_1)_N + (Y_2)_N$.

Proposition 2.30. $(\alpha Y)_N = \alpha(Y_N)$, *for* $\alpha \in \mathbb{R}, \alpha > 0$.

If all the objective functions $f_k, k = 1, \ldots, p$ of the MOP are convex and the feasible set X is convex then the problem is called convex MOP. The outcome set Y of a convex MOP is $\mathbb{R}^p_{\leqq}$-convex, i.e., $Y + \mathbb{R}^p_{\leqq}$ is a convex set.

Another topological property of efficient and nondominated sets is connectedness. Connectedness is an important property when it comes to determine these sets. If Y_N or X_E is connected, the whole nondominated or efficient set can possibly be explored starting from a single nondominated/efficient point using local search ideas. Connectedness will also make the task of selecting a final compromise solution from among the set of efficient solutions X_E easier, as there are no "gaps" in the efficient set.

In Figure 41 two sets Y are shown, one of which has a connected nondominated set and the other has not. Apparently, connectedness cannot be expected, when Y is not $\mathbb{R}^p_{\leqq}$-convex.

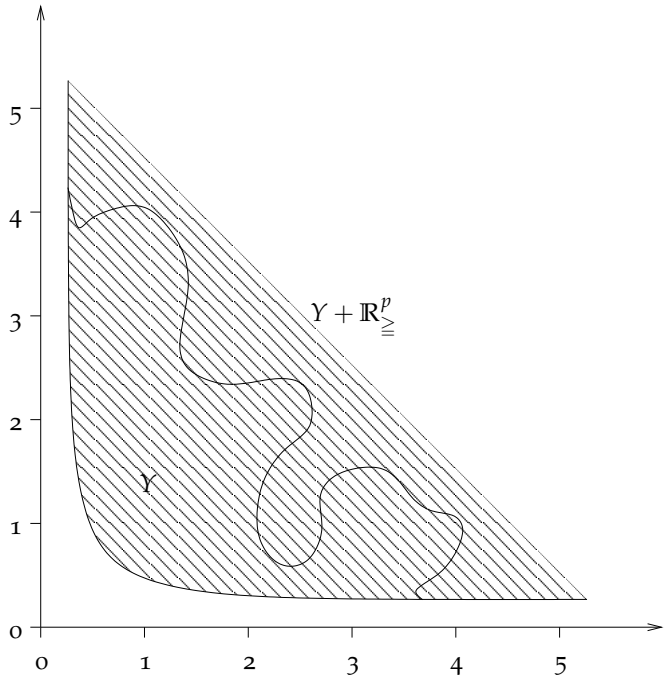


Figure 40: Nondominated points of Y and $Y + \mathbb{R}^p_{\leqq}$ are the same.

Definition 2.31. A function $f\colon S \to \mathbb{R}$ defined on a convex set $S \subseteq \mathbb{R}^n$ is called *quasiconvex* if $\forall\, x, y \in S$ and $\lambda \in [0,1]$ we have

$$f(\lambda x + (1-\lambda)y) \leq \max\{f(x), f(y)\}. \tag{2.31}$$

If furthermore

$$f(\lambda x + (1-\lambda)y) < \max\{f(x), f(y)\} \tag{2.32}$$

$\forall\, x \neq y$ and $\lambda \in (0,1)$, then f is called *strictly quasiconvex*.

Theorem 2.32. *Assume that $f_1, \ldots, f_p$ are continuous and that $X \subset \mathbb{R}^n$ satisfies one of the following conditions*

- *X is a compact, convex set;*
- *X is a closed, convex set and $\forall\, \boldsymbol{y} \in Y, X(\boldsymbol{y}) := \{\, \boldsymbol{x} \in X \mid \boldsymbol{f}(\boldsymbol{x}) \leqq \boldsymbol{y} \,\}$ is compact.*

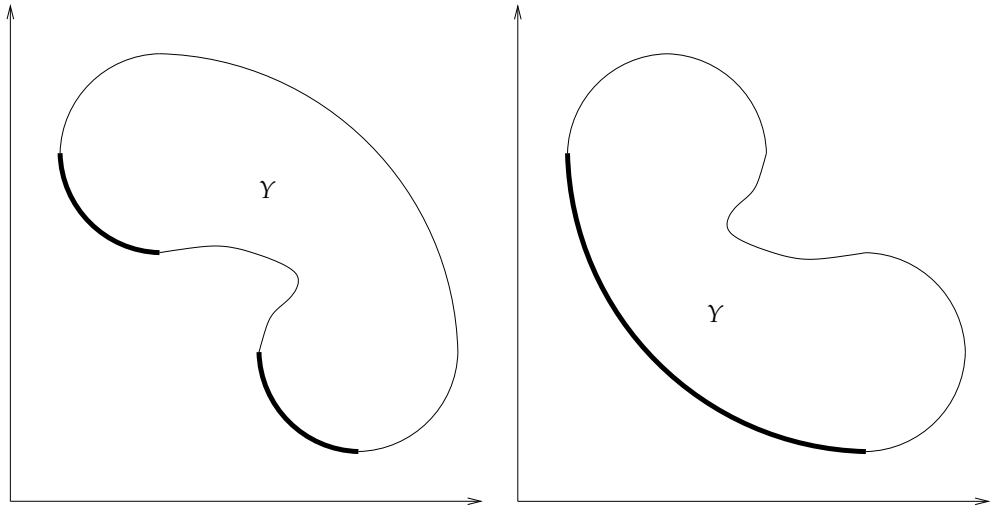
Then the following statements hold:

- *if $f_k\colon \mathbb{R}^n \to \mathbb{R}$ are quasiconvex on X for $k = 1, \ldots, p$ then X_{wE} is connected;*
- *if $f_k\colon \mathbb{R}^n \to \mathbb{R}$ are strictly quasiconvex on X for $k = 1, \ldots, p$ then X_E is connected.*

Since convex functions are continuous and the image of a connected set under a continuous function is connected, it follows that the sets Y_{wN} and Y_N are connected under the assumptions stated in Theorem 2.32, if the objective functions $f_k, k = 1, \ldots, p$ are continuous. However connectedness of Y_N can be proved under more general assumptions.

Definition 2.33. A set $Y \subset \mathbb{R}^p$ is called $\mathbb{R}^p_{\leqq}$*-compact*, if the set $(\boldsymbol{y} - \mathbb{R}^p_{\leqq}) \cap Y$ is compact $\forall\, \boldsymbol{y} \in Y$.

Theorem 2.34. *If Y is closed, convex, and $\mathbb{R}^p_{\leqq}$-compact then Y_N is connected.*



(a) Y_N is not connected. (b) Y_N is connected.

Figure 41: Connectedness of Y_N.

2.3 GENERATION OF THE SOLUTION SETS

There are three general approaches to multiobjective optimization, *scalarization methods, nonscalarizing methods*, and *approximating methods*. The first two techniques convert the MOP into a Single-objective Optimization Problem (SOP), a sequence of SOPs, or another MOP. Under some assumptions solution sets of these new programs yield solutions of the original problem. Solving the SOP typically yields one solution of the MOP so that repetitive solution scheme is needed to obtain a subset of solutions of the MOP.

A different approach is to determine an entire Pareto optimal solution set or a representative subset. Since the exact solution set is very often not attainable, approximating methods have been developed to find an approximated description of this set.

In this section we present some of the most common scalarization and nonscalarizing methods and we explain why they have not been used to deal with the parameter refinement problem, after it has been formulated as a MOP. A particular class of approximating methods have been chosen instead, known as *population-based metaheuristics*. They have been already applied to parameter extraction [60].

For a survey on scalarizing and nonscalarizing techniques, the reader is referred to [33]. Population-based metaheuristics that we implemented to perform parameter extraction are presented in detail in Chapter 3.

2.3.1 *Scalarization methods*

The traditional approach to solving MOPs is by scalarization which involves formulating a MOP-related SOP by means of a real-valued scalarizing function typically being a function of the objective functions of the MOP, auxiliary scalar or vector variables, and/or scalar or vector parameters. Sometimes the feasible set of the

MOP is additionally restricted by new constraint functions related to the objective functions of the MOP and/or the new variables introduced.

2.3.1.1 *The Weighted Sum Method*

In the weighted sum approach a MOP

$$\min_{x \in X}(f_1(x), \ldots, f_p(x)) \tag{2.33}$$

of the Pareto class

$$((X, \mathbb{R}^p, \boldsymbol{f}), \mathrm{id}, (\mathbb{R}^p, \leq)) \tag{2.34}$$

is solved (i.e. its efficient solutions be found) by minimizing a weighted sum of the objective functions

$$\min_{x \in X} \sum_{k=1}^{p} \lambda_k f_k(x), \tag{2.35}$$

which in terms of the classification of Section 2.1.5 is written as

$$((X, \mathbb{R}^p, \boldsymbol{f}), \langle \boldsymbol{\lambda}, \cdot \rangle, (\mathbb{R}, \leq)), \tag{2.36}$$

where $\langle \boldsymbol{\lambda}, \cdot \rangle$ denotes the scalar product in $\mathbb{R}^p$.

Let $Y \subset \mathbb{R}^p$. For a fixed $\boldsymbol{\lambda} \in \mathbb{R}^p_{\geq}$ we denote by

$$S(\boldsymbol{\lambda}, Y) := \left\{ \hat{\boldsymbol{y}} \in Y \;\middle|\; \langle \boldsymbol{\lambda}, \hat{\boldsymbol{y}} \rangle = \min_{\boldsymbol{y} \in Y} \langle \boldsymbol{\lambda}, \boldsymbol{y} \rangle \right\} \tag{2.37}$$

the set of optimal points of Y with respect to λ.

Figure 42 gives an example of a set $S(\boldsymbol{\lambda}, Y)$ consisting of two points $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$. These points are the intersection points of a line $\left\{ \boldsymbol{y} \in \mathbb{R}^2 \;\middle|\; \langle \boldsymbol{\lambda}, \boldsymbol{y} \rangle = \hat{c} \right\}$. Points $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ are nondominated. Considering c as parameter, and the family of lines

$$\left\{ \boldsymbol{y} \in \mathbb{R}^2 \;\middle|\; \langle \boldsymbol{\lambda}, \boldsymbol{y} \rangle = c \right\},$$

we see that in Figure 42 $\hat{c}$ is chosen as the smallest value of c such that the intersection of the line with Y is nonempty.

Graphically, to find $\hat{c}$ we can start with a large value of the parameter c and translate the line in parallel towards the origin as much as possible while keeping a nonempty intersection with Y. Analytically, this means finding elements of $S(\boldsymbol{\lambda}, Y)$.

Now the questions are:

- Does this process always yield nondominated points? (Is $S(\boldsymbol{\lambda}, Y) \subset Y_N$?)
- If so, can all nondominated points be detected this way?
  (Is $Y_N \subset \cup_{\boldsymbol{\lambda} \in \mathbb{R}^p_{\geq}} S(\boldsymbol{\lambda}, Y)$?)

The following Theorem gives the answers.

Theorem 2.35. *1. Let $\hat{x} \in X$ be an optimal solution of (2.35). The following statements hold:*

- *if $\boldsymbol{\lambda} > 0$ then $\hat{x} \in X_{pE}$;*
- *if $\boldsymbol{\lambda} \geq 0$ then $\hat{x} \in X_{wE}$;*

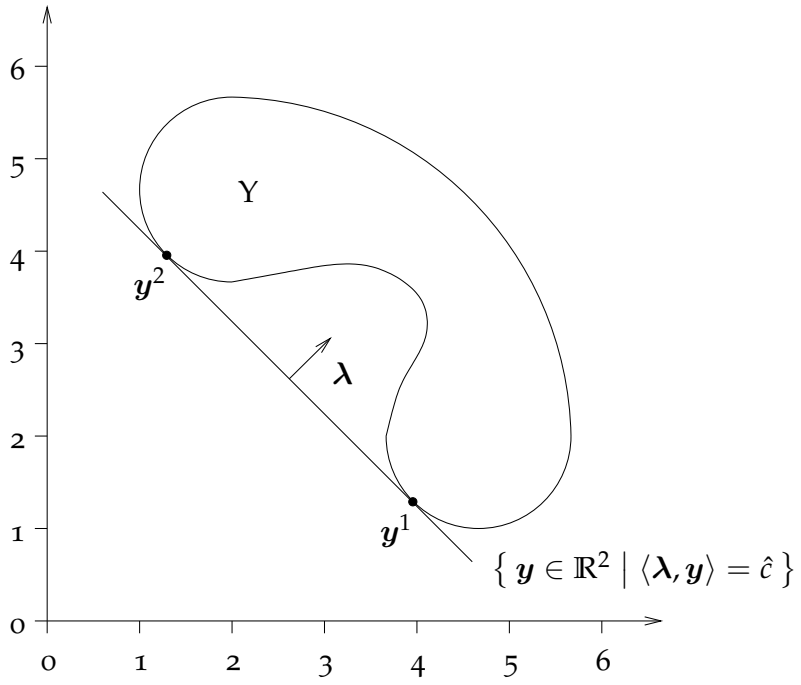


Figure 42: A set $S(\boldsymbol{\lambda}, Y)$.

- *if* $\boldsymbol{\lambda} \geq 0$ *and* $\hat{x}$ *is a unique optimal solution of* (2.35) *then* $\hat{x} \in X_{sE}$.

2. *Let* X *be a convex set and* $f_k, k = 1, \ldots, p$ *be convex functions. Then the following statements hold:*
   - *if* $\hat{x} \in X_{pE}$ *then there exists some* $\boldsymbol{\lambda} > 0$ *such that* $\hat{x}$ *is an optimal solution of* (2.35)*;*
   - *if* $\hat{x} \in X_{wE}$ *then there exists some* $\boldsymbol{\lambda} \geq 0$ *such that* $\hat{x}$ *is an optimal solution of* (2.35).

Point 2 of the previous theorem states that the weighted sum method allows computation of all the properly efficient and weakly efficient solutions for convex problems by varying $\boldsymbol{\lambda}$. For nonconvex problems, however, it may work poorly. Consider the two following examples.

Example 2.36. In Figure 43, the feasible set in objective space for a nonconvex problem is shown (Y is not $\mathbb{R}^2_{\leqq}$-convex). Since all objective vectors $\boldsymbol{y} = (f_1(x), f_2(x))$, which attain the same value $c = \sum_{k=1}^{2} \lambda_k f_k(x)$ of the weighted sum objective, are located on a straight line, the minimization problem (2.37) amounts to pushing this line towards the origin, until it intersects the boundary of Y. In Figure 43 this is illustrated for two weighting vectors (λ_1, λ_2) and (λ_1', λ_2'), that lead to the nondominated points $\boldsymbol{y}$ and $\boldsymbol{y}'$. The third point $\hat{\boldsymbol{y}} \in Y_N$ is properly nondominated, but none of its preimages x under $\boldsymbol{f}$ can be an optimal solution of (2.35) for any choice of $(\lambda_1, \lambda_2) \in \mathbb{R}^p_{>}$.

Example 2.37. Let $X = \left\{ \boldsymbol{x} \in \mathbb{R}^2_{\leqq} \;\middle|\; x_1^2 + x_2^2 \geq 1 \right\}$ and $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x}$. In this case $X_E = \{ \boldsymbol{x} \in X \mid x_1^2 + x_2^2 = 1 \}$, yet $\hat{\boldsymbol{x}}^1 = (1, 0)$ and $\hat{\boldsymbol{x}}^2 = (0, 1)$ are the only feasible solutions that are optimal solutions of (2.35) for any $\boldsymbol{\lambda} \geq 0$ (see Figure 44).

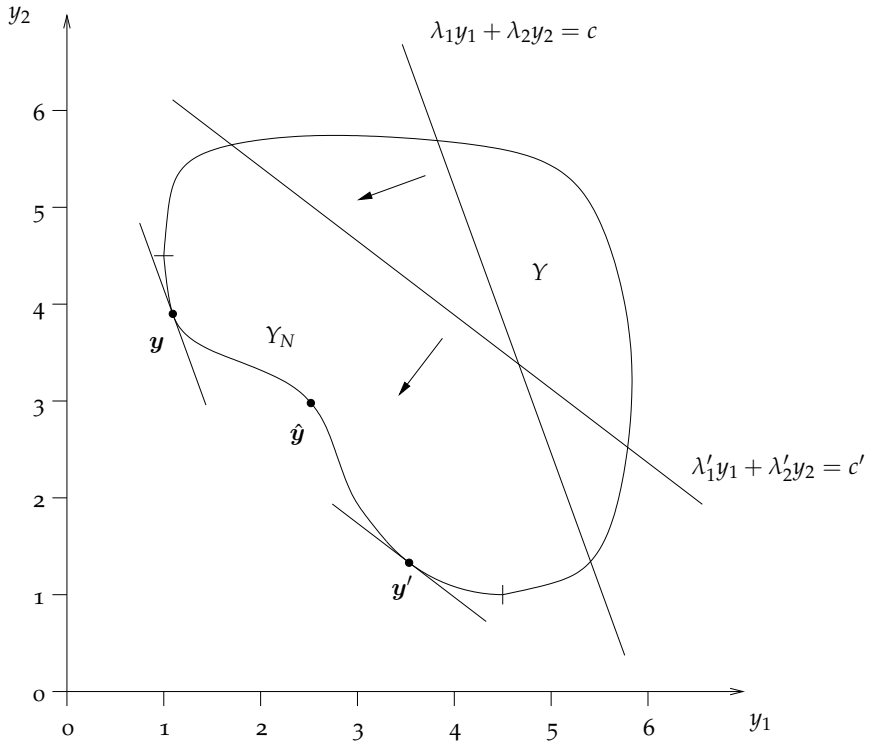


Figure 43: Properly nondominated $\hat{y} \in Y_N$.

Another problem concerning the weighted sum method lies in the proper selection of the weights to characterize the decision-maker's preferences. In practice, it can be very difficult to precisely and accurately select these weights, even for someone familiar with the problem domain. General applicability and ease of use are then sacrificed. Compounding this drawback is that scaling amongst objectives is needed and small perturbations in the weights can sometimes lead to quite different solutions.

2.3.1.2 *The ε-Constraint Method*

In the ε-constraint method there is no aggregation of criteria, instead only one of the original objectives is minimized, while the others are transformed to constraints.

The multicriteria optimization problem (2.33) is replaced by the ε-constraint problem

$$\begin{aligned} &\min_{x \in X} f_j(x) \\ &\text{subject to } f_k(x) \leq \varepsilon_k \quad k = 1, \ldots, p \quad k \neq j, \end{aligned} \tag{2.38}$$

where $\varepsilon_k \in \mathbb{R}^p, k = 1, \ldots, p \quad k \neq j$.

Figure 45 illustrates a biobjective problem, where an upper bound constraint is put on $f_1(x)$. The optimal values of (2.38) problem with $j = 2$ for two values of ε_1 are indicated. These show that the constraints $f_k(x) \leq \varepsilon_k$ might or might not hold with equality at an optimal solution of (2.38).

The following theorem justifies the approach.

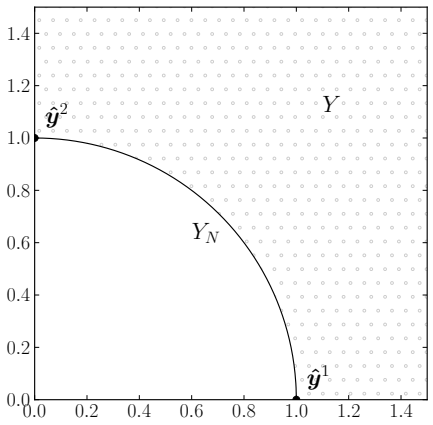


Figure 44: The weighted sum method fails for nonconvex problems.

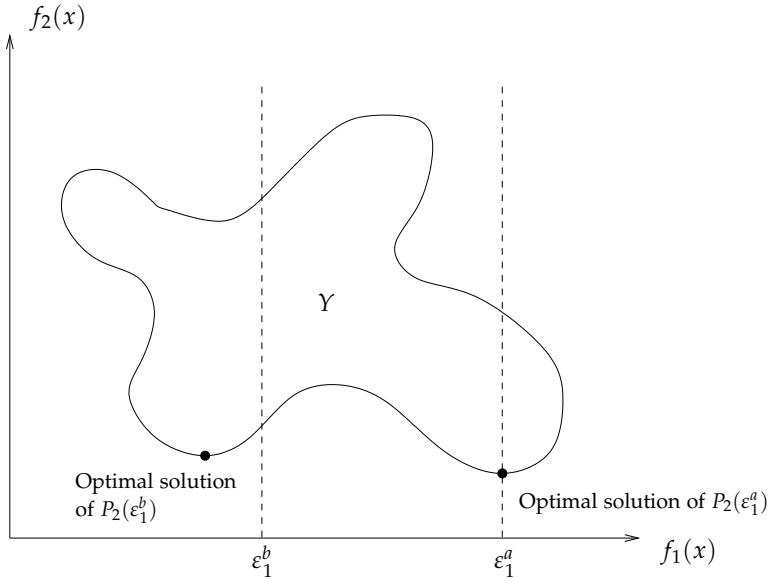


Figure 45: Optimal solutions of ε-constraint problems.

Theorem 2.38. 1. *Let $\hat{x}$ be an optimal solution of (2.38) for some j. Then $\hat{x}$ is weakly efficient.*

2. *Let $\hat{x}$ be a unique optimal solution of (2.38) for some j. Then $\hat{x} \in X_{sE}$ (and therefore $\hat{x} \in X_E$).*

3. *The feasible solution $\hat{x} \in X$ is efficient if and only if there exists an $\hat{\varepsilon} \in \mathbb{R}^p$ such that $\hat{x}$ is an optimal solution of (2.38) for all $j = 1, \ldots, p$.*

The proof can be found in [30]. Point 3 shows that with appropriate choices of ε all efficient solutions can be found. This result is independent of the structure of X, i.e., it is also true for nonconvex problems. However, as the proof shows, these ε_j values are equal to the actual objective values of the efficient solution one would like to find. A confirmation or check for efficiency is obtained rather than the discovery of efficient solutions.

This method has two main drawbacks.

- It is usually necessary to solve the problem (2.38) repeatedly to obtain several solutions of the original MOP with different trade-offs among objectives. It is therefore important to ask whether it can be solved with acceptable computational effort. Ehrgott and Ryan [32] report on a bicriteria set partitioning model in airline crew scheduling. They show that the problem (2.38) cannot be solved within acceptable time.
- Another issue is related to properly efficient solutions. In practical situations, decision makers are usually interested in properly efficient solutions rather than just efficient ones. The ε-constraint method as it has been presented so far does not provide results on proper efficiency of optimal solutions.

In [31] two modifications of the ε-constraint method are proposed to deal with these two issues.

2.3.2 *Nonscalarizing approaches*

In contrast to scalarizing approaches discussed in Section 2.3.1, nonscalarizing approaches do not explicitly use a scalarizing function but rather rely on other optimality concepts. In this section we summarize the most important approaches.

2.3.2.1 *The Lexicographic Approach*

Lexicographic optimization problems arise when conflicting objectives exist in a decision problem but the objectives have to be considered in a hierarchical manner. Section 2.1.3 presents a very simple lexicographic optimization problem. In [95] the problem of optimizing the water resources planning for Lake Verbano (Lago Maggiore) in northern Italy is described. The goal is to determine an optimal policy for the management of the water supply over some planning horizon. The objectives are to maximize flood protection, minimize supply shortage for irrigation, and maximization of electricity generation. This order of objectives is prescribed by law, so that the problem indeed has a lexicographic nature.

In lexicographic optimization first a ranking of the objectives is defined, then objective vectors are compared lexicographically. Let $\pi\colon \{1,\dots,p\} \to \{1,\dots,p\}$ be a permutation and consider the permutation $(f_{\pi(1)},\dots,f_{\pi(p)})$ of the objective functions. Let $\boldsymbol{f}^\pi\colon \mathbb{R}^n \to \mathbb{R}^p$ be $(f_{\pi(1)},\dots,f_{\pi(p)})$. The lexicographic problem is formulated as

$$\operatorname{lexmin}_{x\in X} \boldsymbol{f}^\pi(x) \tag{2.39}$$

Definition 2.39. A feasible solution $\hat{x} \in X$ is *lexicographically optimal* or a *lexicographic solution* if there is no $x \in X$ such that $\boldsymbol{f}^\pi(x) <_{\text{lex}} \boldsymbol{f}^\pi(\hat{x})$.

Recall that $\boldsymbol{y}^1 <_{\text{lex}} \boldsymbol{y}^2$ if $y^1_q < y^2_q$ where $q = \min\{\, k \mid y^1_k \neq y^2_k \,\}$ and that the lexicographic order is total. Therefore, in addition to Definition 2.39, we can state that $\hat{x} \in X$ is lexicographically optimal, if

$$\boldsymbol{f}^\pi(\hat{x}) \leq_{\text{lex}} \boldsymbol{f}^\pi(x) \quad \forall\, x \in X. \tag{2.40}$$

First, we establish the relationship between lexicographically optimal solutions and efficient solutions.

Proposition 2.40. *Let $\hat{x} \in X$ be such that $\boldsymbol{f}^\pi(\hat{x}) \leq_{\text{lex}} \boldsymbol{f}^\pi(x) \quad \forall\, x \in X$. Then $\hat{x} \in X_E$.*

While the essential feature of efficiency is the existence of tradeoff between objectives, lexicographic optimality implies a ranking of the objectives in the sense that optimization of $f_{\pi(k)}$ is only considered once optimality for objectives

$$\{\pi(1), \ldots, \pi(k-1)\}$$

has been established. That means objective $f_{\pi(1)}$ has the highest priority, and only in the case of multiple optimal solutions objectives $f_{\pi(2)}$ and further objectives are considered. This priority ranking implies the absence of tradeoffs between criteria. An improvement in an objective $f_{\pi(k)}$ can never compensate the deterioration of any $f_{\pi(i)}, i < k$.

The hierarchy among criteria allows us to solve lexicographic optimization problems sequentially, minimizing one objective $f_{\pi(k)}$ at a time and using optimal objective values of $f_{\pi(i)}, i < k$ as constraints, as shown in Algorithm 2.1.

Algorithm 2.1 Lexicographic Optimization

Input: feasible set X and permutation of the objective functions $\boldsymbol{f}^\pi$
Output: set of lexicographically optimal solutions
$X_1^\pi \leftarrow X$
$k \leftarrow 1$
while $k \leq p$ **do**
solve the single objective optimization problem

$$\min_{x \in X_k^\pi} f_{\pi(k)}(x) \tag{2.41}$$

if (2.41) has a unique optimal solution $\hat{x}_k$ **then**
$\hat{x}_k$ is the unique optimal solution of the lexicographic optimization problem **return** $\hat{x}_k$
else if (2.41) is unbounded **then**
the lexicographic optimization problem is unbounded, **stop**
else if $k = p$ **then return** $\left\{ x \in X_p^\pi \;\middle|\; f_{\pi(p)}(x) = \min_{x \in X_p^\pi} f_{\pi(p)}(x) \right\}$
end if
$X_{k+1}^\pi \leftarrow \left\{ x \in X_k^\pi \;\middle|\; f_{\pi(k)}(x) = \min_{x \in X_k^\pi} f_{\pi(k)}(x) \right\}$
$k \leftarrow k + 1$
end while

Algorithm 2.1 also gives a correct solution if $f_{\pi(k)}$ is unbounded over X, but bounded over X_k^π. Note that, if a problem

$$\min_{x \in X_k^\pi} f_{\pi(k)}(x)$$

is unbounded, it is not possible to define X_{k+1}^π.

Proposition 2.41. *If $\hat{x}$ is a unique optimal solution of (2.41) with $k < p$, or if $\hat{x}$ is an optimal solution of (2.41) with $k = p$ then $\boldsymbol{f}^\pi(\hat{x}) \leq_{\text{lex}} \boldsymbol{f}^\pi(x) \quad \forall\, x \in X$.*

Proposition 2.42. *If x is a unique optimal solution of (2.41) for some $k \in \{1, \ldots, p\}$, then $x \in X_{sE}$.*

Note that the inclusion $X_\pi = \cup_\pi X_p^\pi \subset X_E$ is usually strict, which means that not all the efficient points can be found with the lexicographic approach.

Example 2.43. Let $X = [0,1]$ and $f_1(x) = x, f_2(x) = 1 - x$. Clearly $X_E = X$. The optimal solution of

$$(([0,1], \mathbb{R}^2, \boldsymbol{f}),\ \mathrm{id}, (\mathbb{R}^2, <_{\mathrm{lex}}))$$

is $\hat{x} = 0$, the optimal solution of

$$(([0,1], \mathbb{R}^2, \boldsymbol{f}), \pi, (\mathbb{R}^2, <_{\mathrm{lex}})),$$

where $\pi(y_1, y_2) = (y_2, y_1)$ is $\hat{x} = 1$. Therefore $X_\pi = \{0,1\} \neq X_E$.

Moreover, because of the uniqueness of both lexicographically optimal solutions in this example $X_\pi \subset X_{sE}$, and again the inclusion is strict, as $X_E = X_{sE}$.

2.3.2.2 *The Max-ordering Approach*

The max-ordering approach does only consider the objective function f_k which has the highest (worst) value. The preference relation of the max-ordering approach is $\boldsymbol{y}^1 \leq_{\mathrm{MO}} \boldsymbol{y}^2$ if

$$\max_{k=1,\ldots,p} y_k^1 \leq \max_{k=1,\ldots,p} y_k^2.$$

The max-ordering problem is formulated as

$$\min_{x \in X} \max_{k=1,\ldots,p} f_k(x). \tag{2.42}$$

An optimal solution of (2.42) is weakly efficient. Furthermore, if $X_E \neq \emptyset$ and (2.42) has an optimal solution then there exists an optimal solution of (2.42) which is efficient, and consequently a unique optimal solution of (2.42) is efficient.

It is possible to include a weight vector $\boldsymbol{\lambda} \in \mathbb{R}^p_{\geq}$ so that the weighted max-ordering problem becomes

$$\min_{x \in X} \max_{k=1,\ldots,p} \lambda_k f_k(x). \tag{2.43}$$

Theorem 2.44. *Let $Y \subset \mathbb{R}^p_{>}$.*

1. *Let $\boldsymbol{\lambda} \in \mathbb{R}^p_{\geq}$. If $\hat{x} \in X$ is an optimal solution of (2.43) then $\hat{x} \in X_{wE}$.*
2. *If $\hat{x} \in X_{wE}$ there exists a $\boldsymbol{\lambda} \in \mathbb{R}^p_{>}$ such that $\hat{x}$ is an optimal solution of (2.43).*

The max-ordering problem (2.42) can be solved as a single objective optimization problem. Let z be $\max_{k=1,\ldots,p} f_k(x)$. Then (2.42) can be rewritten as

$$\begin{aligned} &\min z \\ &\text{subject to } f_k(x) \leq z \quad k = 1, \ldots, p \\ &x \in X. \end{aligned} \tag{2.44}$$

Section 2.1.3 presents a very simple max-ordering problem.

A common application of max-ordering problems is location planning. Ehrgott describes the problem of locating rescue helicopters in South Tyrol, Italy [29].

The objective in this problem is to minimize the distance between potential accident sites and the closest helicopter location. In order to minimize worst case response times in an emergency, the problem can be formulated as follows. Let $\boldsymbol{x}^h = (x_1^h, x_2^h), h \in H$ denote the variables that define helicopter locations and $(a_1^k, a_2^k), k \in 1, \ldots, p$ the potential emergency sites. Optimal helicopter locations are found by solving

$$\min_{\boldsymbol{x} \in \mathbb{R}^{2|H|}} \max_{k \in 1,\ldots,p} f_k(\boldsymbol{x}) \tag{2.45}$$

where $f_k(\boldsymbol{x})$ is defined as

$$f_k(\boldsymbol{x}) = \min_{h \in H} w_k \|\boldsymbol{x}^h - \boldsymbol{a}^k\|_2. \tag{2.46}$$

Operator $\|\cdot\|_2$ denotes the Euclidean norm.

2.3.2.3 *Lexicographic Max-Ordering Optimization*

An optimal solution of a max-ordering optimization problem is not necessarily efficient, because the max-ordering optimality concept considers only one of the p objective values at each $x \in X$, namely the worst. A straightforward idea is to extend this to consider the second worst objective, the third worst objective, etc., in the case that the max-ordering problem has several optimal solutions. This approach is similar to lexicographic optimization and considers a ranking of the objective values $f_1(x), \ldots, f_p(x)$. The difference is that the ranking is from worst to best value and thus depends on x. This result is called lexicographic max-ordering optimality, because it is a combination of max-ordering and lexicographic optimality, where the lexicographic order is applied to a nonincreasing ordered sequence of the objectives.

Definition 2.45. 1. For $\boldsymbol{y} \in \mathbb{R}^p$ let

$$\operatorname{sort}(\boldsymbol{y}) := (\operatorname{sort}_1(\boldsymbol{y}), \ldots, \operatorname{sort}_p(\boldsymbol{y}))$$

such that $\operatorname{sort}_1(\boldsymbol{y}) \geq \cdots \geq \operatorname{sort}_p(\boldsymbol{y})$ be a function that reorders the components of $\boldsymbol{y}$ in a nonincreasing way.

2. A feasible solution $\hat{x} \in X$ is called a *lexicographic max-ordering solution (lex-MO solution)* if

$$\operatorname{sort}(\boldsymbol{f}(\hat{\boldsymbol{x}})) \leq_{\text{lex}} \operatorname{sort}(\boldsymbol{f}(x)) \quad \forall\, x \in X. \tag{2.47}$$

A lexicographic max-ordering optimization problem can be written as

$$\min_{x \in X} \operatorname{sort}(\boldsymbol{f}(x)). \tag{2.48}$$

According to this definition, we apply a mapping $\operatorname{sort}\colon \mathbb{R}^p \to \mathbb{R}^p$ to the objective vectors $\boldsymbol{f}(x)$, which reorders the components, and apply the lexicographic order to compare reordered objective vectors.

This means that sort is used as model map and the lexicographic order for comparison. Thus a lexicographic max-ordering problem is denoted by

$$((X, \mathbb{R}^p, \boldsymbol{f}), \operatorname{sort}, (\mathbb{R}^p, <_{\text{lex}})). \tag{2.49}$$

The set of optimal solutions will be denoted by $X_{\text{lex-MO}}$ and its image in objective space by $Y_{\text{lex-MO}} = \boldsymbol{f}(X_{\text{lex-MO}})$.

Theorem 2.46. *The following relationship between lex-MO solutions, efficient solutions, and max-ordering solutions holds*

$$X_{\text{lex-MO}} \subset X_E \cap X_{\text{MO}} \tag{2.50}$$

and $X_{\text{lex-MO}} = X_E \cap X_{\text{MO}}$ *if* $\boldsymbol{f}(x)$ *is the same* $\forall\, x \in X_{\text{MO}}$.

Inclusion (2.50) in Theorem 2.46 indicates that the intersection of the efficient set and X_{MO} does not contain just the lex-MO solutions in general.

Example 2.47. Consider problem data with feasible set $X = \{a, b, c, d, e, f\}$, for which the objective function values are shown in Table 6.

x	$f(x)$	sort $f(x)$
a	$(1,3,8,2,4)$	$(8,4,3,2,1)$
b	$(4,3,8,1,1)$	$(8,4,3,1,1)$
c	$(7,5,4,6,1)$	$(7,6,5,4,1)$
d	$(3,7,4,6,5)$	$(7,6,5,4,3)$
e	$(4,7,5,6,5)$	$(7,6,5,5,4)$
f	$(5,6,7,3,8)$	$(8,7,6,5,3)$

Table 6: Feasible solutions and objective values in Example 2.47.

The sorted objective vectors are also shown for convenience. It is easily seen that $X_{\text{MO}} = \{c, d, e\}$, that $X_E = \{a, b, c, d, f\}$, and that $X_{\text{lex-MO}} = \{c\}$. Therefore $X_{\text{lex-MO}} \subset X_{\text{MO}} \cap X_E$, but the inclusion is strict. Note that lexicographically optimal solutions with respect to all permutations are a, b, c, d, so that $X_\pi \subset X_E$ and $X_{\text{lex-MO}} \subset X_\pi$ and both of these inclusions are strict.

Lex-MO solutions are not necessarily lexicographically optimal, because even if sort defines a permutation of $\boldsymbol{f}(x)$, it is one which depends on x, see Example 2.43, where $X_{\text{lex-MO}} = \{0.5\}$ and $X_\pi = \{0, 1\}$.

The following theorem states that X_E can be identified by solving lex-MO problems with positive weights $\boldsymbol{\lambda}$ for the objective functions.

Theorem 2.48. *A feasible solution* $\hat{x} \in X$ *is efficient if and only if there exists a* $\boldsymbol{\lambda} \in \mathbb{R}^p_>$ *such that* $\hat{x}$ *is an optimal solution of the lex-MO optimization problem*

$$\text{lexmin}_{x \in X}\, \text{sort}((\lambda_1 f_1(x), \ldots, \lambda_p f_p(x))). \tag{2.51}$$

Finding the solution of lex-MO problems is not always a straightforward task. If we apply a procedure like Algorithm 2.1, we would have to solve the max-ordering problem first. Then fix the value of the worst objective, solve the max-ordering problem for the remaining $p - 1$ objectives and so on. Unfortunately, we do not know which objective will be the worst, and there may be several max-ordering solutions x with the worst value obtained for different objectives. Taking into account all possible combinations would mean $p!$ sequences of the objectives, which would be computationally prohibitive in general.

Example 2.49. Consider the problem in Example 2.47. Let us apply a procedure like Algorithm 2.1, where the single objective optimization problem to solve at each iteration is similar to (2.44).

Define $X_1 = X$ and $k = 1$. Solve the problem

$$\begin{aligned} &\min z \\ &\text{subject to } f_j(x) \leq z \quad j = 1,2,3,4,5 \\ &x \in X_1. \end{aligned} \tag{2.52}$$

We obtain three max-ordering solutions $\{c, d, e\}$ with $f_1(c) = 7, f_2(d) = f_2(e) = 7$. No termination condition is met: (2.52) has not a unique optimal solution, it is not unbounded and $k \leq 5$. To define the next problem to solve we have to fix the value of the worst objective, namely f_1 or f_2. Let us fix f_2 first. Set $X_2 = \{c, d, e\}$ and $k = 2$. Solve the problem

$$\begin{aligned} &\min z \\ &\text{subject to } f_j(x) \leq z \quad j = 1,3,4,5 \\ &x \in X_2. \end{aligned}$$

We obtain two max-ordering solutions $\{d, e\}$ with $f_4(d) = f_4(e) = 6$. No termination condition is met. Fix f_4 and set $X_3 = \{d, e\}$ and $k = 3$. Solve the problem

$$\begin{aligned} &\min z \\ &\text{subject to } f_j(x) \leq z \quad j = 1,3,5 \\ &x \in X_3. \end{aligned}$$

We have $f_5(d) = f_3(e) = f_5(e) = 5$ for the two max-ordering solutions $\{d, e\}$. We have again obtained the worst value for two different objectives. No termination condition is met. So let us fix f_3 and set $X_4 = \{d, e\}$ and $k = 4$. Solve the problem

$$\begin{aligned} &\min z \\ &\text{subject to } f_j(x) \leq z \quad j = 1,5 \\ &x \in X_4. \end{aligned}$$

We obtain two max-ordering solutions $\{d, e\}$ with $f_5(d) = f_5(e) = 5$. No termination condition is met. Fix f_5 and set $X_5 = \{d, e\}$ and $k = 5$. Solve the problem

$$\begin{aligned} &\min z \\ &\text{subject to } f_j(x) \leq z \quad j = 1 \\ &x \in X_5. \end{aligned}$$

We have only one solution, $\{d\}$. We still do not know if it is lexicographic max-ordering optimal or just max-ordering optimal. We could have fixed f_5 instead of f_3 to define the single objective problem for $k = 4$. In this case the problem would have returned the optimal solution $\{d\}$ immediately. Moreover, we could have fixed f_1 instead of f_2 to define the problem for $k = 2$. In this case the optimal solution $\{c\}$ would have been returned. Now, if we compare c and d, we see that only c is a lex-MO solution.

We had to take into account several sequences of the objectives to find the lex-MO solution c. This can be computationally prohibitive in general.

There are exceptions, however. In [30] it is shown that lex-MO problems are easily solved when X is a finite set or the MOP is convex.

3

EVOLUTIONARY ALGORITHMS FOR SOLVING PARAMETER EXTRACTION PROBLEMS

Evolutionary Computation (EC) is a generic term for several stochastic search methods which computationally simulate the natural evolutionary process. EC embodies the techniques of Genetic Algorithm (GA)s, Evolution Strategy (ES)s and Evolutionary Programming (EP), collectively known as Evolutionary Algorithm (EA)s [11]. These techniques are loosely based on natural evolution and the Darwinian concept of *survival of the fittest* [43]. Common between them are the reproduction, random variation, competition and selection of contending individuals within some population [11].

In this chapter an overview is presented describing EAs we used to perform parameter extraction of a compact model of a generic device. Section 3.1 describes basic EA structural terms and concepts. Section 3.2 introduces the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). The CMA-ES is a stochastic method for real-parameter (continuous domain) optimization of non-linear, non-convex functions. Two variants of this method are presented: a non-elitist variant which we used to optimize one output characteristic at a time of an electronic device, and an elitist variant which is the starting point for developing a multi-objective version of the CMA-ES. Section 3.3 introduces some further terminology and notation used in successive sections. Section 3.4 presents the components of multi-objective EA. In Section 3.5 a description of the Multi Objective Evolutionary Algorithm (MOEA)s implemented in the Python library for parameter extraction is provided. In Section 3.6 the issue of MOEA's scalability to many-objective problems is addressed. These problems are encountered in parameter extraction of semiconductor devices when fitting a compact model against a large number of output characteristics. Section 3.7 reviews some methodologies for statistical performance assessment of stochastic multiobjective optimizers.

3.1 EA BASICS

The following presentation describes basic EA structural terms and concepts; the described terms' "meanings" are normally analogous to their genetic counterparts.

In EA terminology, a solution vector $\boldsymbol{x} \in X$, where X is the feasible space (see Section 2.1.2), is called an *individual* or a *chromosome*. Chromosomes are made of discrete units called *genes* which take on certain values (*alleles*) from some genetic alphabet. A *locus* identifies a gene's position within the chromosome. Each gene controls one or more features of the chromosome. In first EA implementations, genes were assumed to be binary digits. In later implementations, more varied gene types have been introduced, like real-valued gene types. Normally, a chromosome corresponds to a unique solution $\boldsymbol{x}$ in the feasible space. This requires a mapping between the solution space and the chromosomes. This mapping is called an *encoding*. In fact, EAs work on the encoding of a problem, not on the

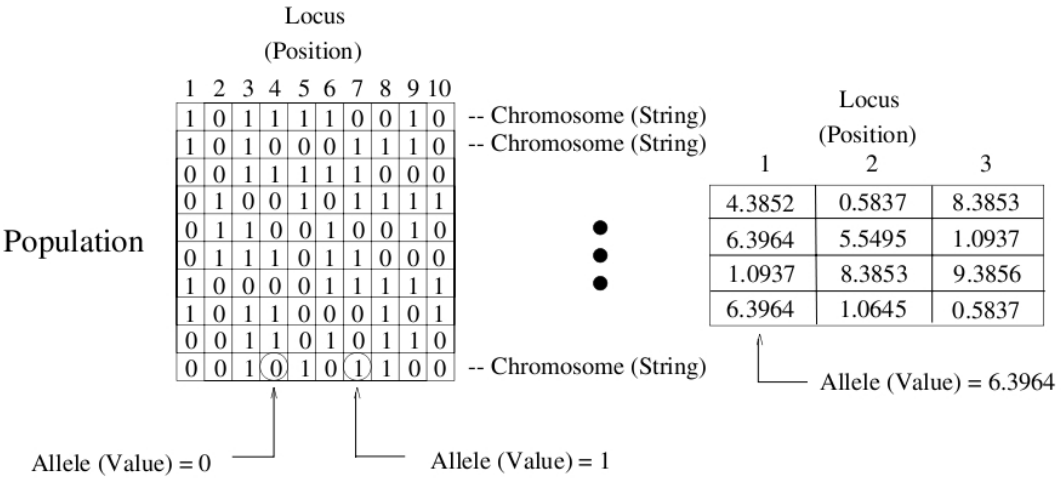


Figure 46: Generalized EA data structures and terminology [11].

problem itself. Finally, a given set a chromosomes is termed a *population*. These concepts are pictured in Figure 46 (for both binary and real-valued chromosomes) and in Figure 47.

In parameter extraction problems, an individual is a particular realization of the parameter set of a device model. Each parameter represents a specific gene and it is mapped to another real number by all the implemented population-based metaheuristic. We have then worked with real-valued chromosomes.

Just as in nature, Evolutionary Operator (EVOP)s operate on an EA's population attempting to generate solutions with higher and higher fitness. The three major EVOPs associated with EAs are *mutation*, *recombination* and *selection*. Illustrating this, Figure 48 shows *bitwise mutation* on an encoded string where a 1 is changed to a 0, or vice versa. Figure 49 shows *single-point crossover* (a form of recombination) operating on two parent binary strings; each parent is cut and recombined with a piece of the other. Above-average individuals in the population are *selected* to become members of the next generation more often than below-average individuals. The selection EVOP effectively gives strings with higher fitness a higher probability of contributing one or more children in the next generation. Figure 50 shows the operation of the common *roulette-wheel* selection (a *fitness proportional* selection operator) on two different populations of four strings each. Each string in the population is assigned a portion of the wheel proportional to the ratio of its fitness and the population's average fitness.

Real-valued chromosomes also undergo these same EVOPs even if implemented differently (see [21, 22, 26, 27]).

An EA requires both an *objective* and *fitness* function, which are fundamentally different. The objective function defines the EA's optimality condition (and is a feature of the problem domain) while the fitness function (in the algorithm domain) measures how "well" a particular solution satisfies that condition and assigns a corresponding real-value to that solution.

When describing EAs, *genotype* refers to decision variable space X, whereas *phenotype* refers to objective function space Y.

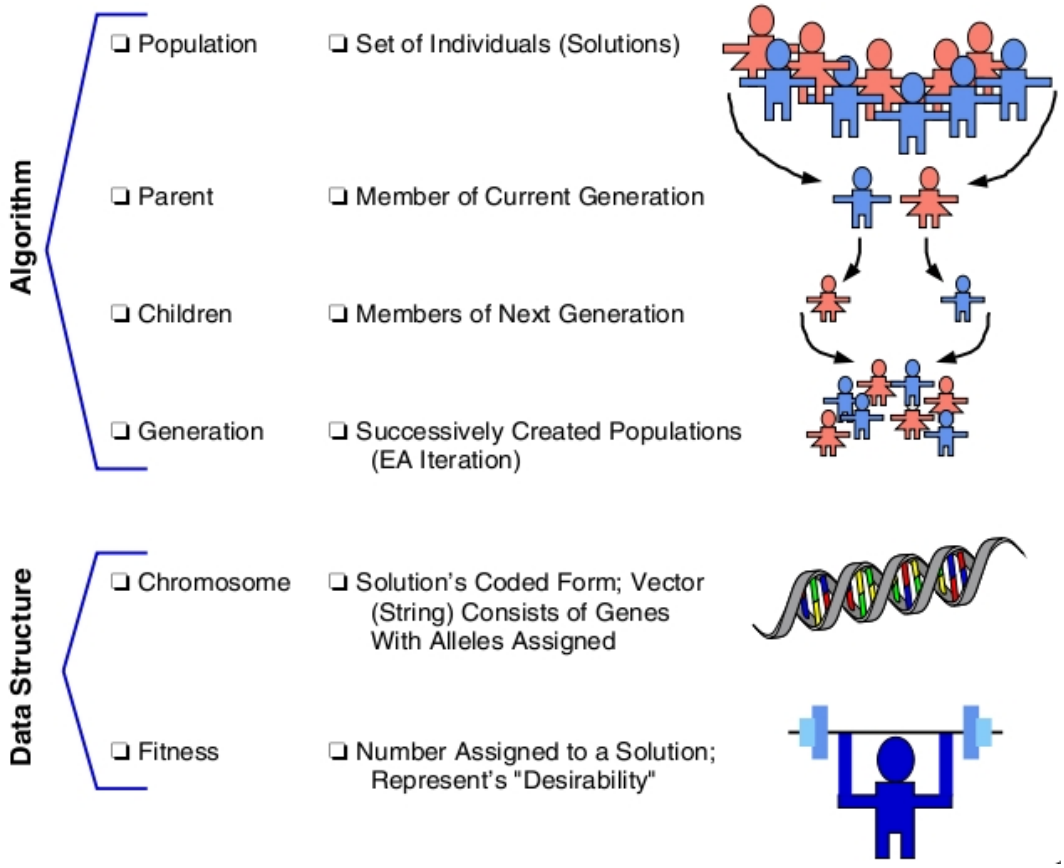


Figure 47: Key EA components [11].

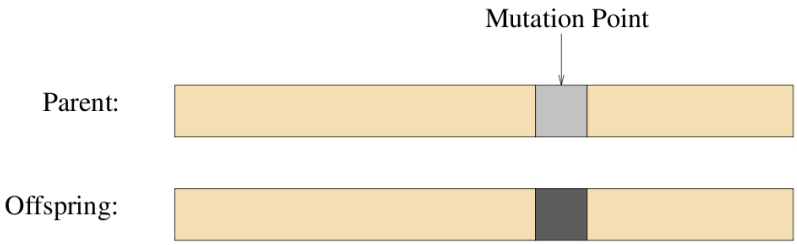


Figure 48: Bitwise mutation [11].

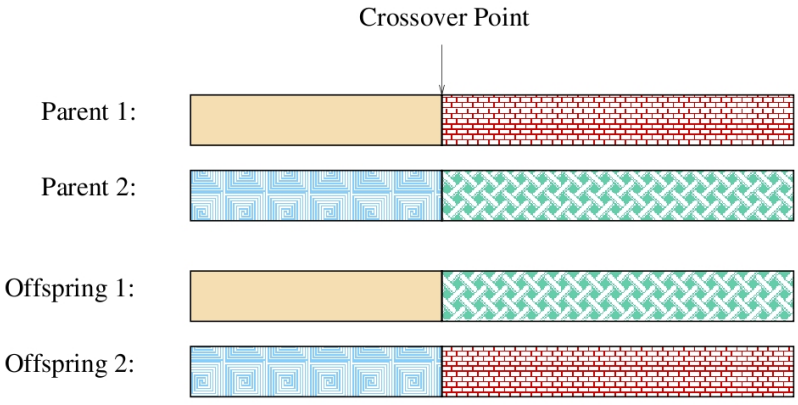


Figure 49: Single-point crossover [11].

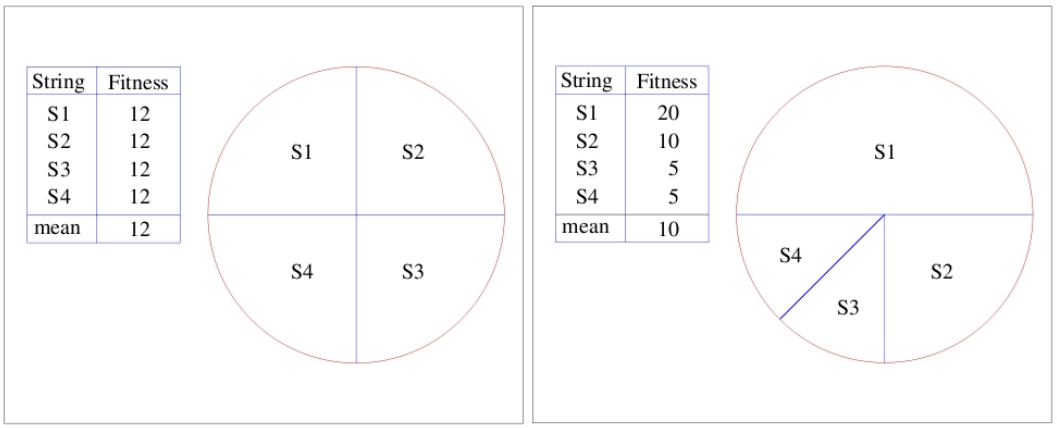


Figure 50: Roulette wheel selection [11].

Many other selection techniques are implemented by EAs, e.g., tournament and ranking [3]. Tournament selection operates by randomly choosing some number q individuals from the generational population and selecting the "best" to survive into the next generation. Binary tournaments ($q = 2$) are probably the most common. Ranking assigns selection probabilities solely on an individual's rank, ignoring absolute fitness value. Two other selection techniques are the $(\mu + \lambda)$ and (μ, λ) selection strategies, where μ represents the number of parent solutions and λ the number of children. The former selects the μ best individuals drawing from *both* the parents and children, the latter selects μ individuals from the child population only.

The choice of EA selection technique influences two conflicting goals that are common to all EA search: *exploration* and *exploitation* [3]. One goal is achieved at the expense of another. Before starting a discussion about EAs for multi-objective optimization, the CMA-ES will be introduced in the next section. The CMA-ES is an evolutionary algorithm for difficult non-linear non-convex single objective optimization problems in continuous domain.

3.2 THE CMA EVOLUTION STRATEGY FOR SINGLE OBJECTIVE OPTIMIZATION

The CMA-ES is an evolutionary algorithm for difficult *non-linear non-convex* optimization problems in continuous domain. The CMA-ES is typically applied to unconstrained or bounded constraint optimization problems, and search space dimension between three and a hundred. The method should be applied if derivative based methods, e.g. quasi-Newton BFGS or conjugate gradient [79], fail due to a rugged search landscape (e.g. discontinuities, sharp bends or ridges, noise, local optima, outliers). If second order derivative based methods are successful, they are usually faster than the CMA-ES [46].

Similar to quasi-Newton methods (but not inspired by them), the CMA-ES is a *second order* approach estimating a positive definite matrix within an iterative procedure (more precisely: a covariance matrix, that is, *on convex-quadratic functions*, closely related to the inverse Hessian [48]). This makes the method feasible on non-separable and/or badly conditioned problems. Unlike quasi-Newton methods, the

CMA-ES does not use or approximate gradients and does not even presume or require their existence. This makes the method feasible on non-smooth and even non-continuous problems, as well as on multimodal and/or noisy problems. Two studies showed that it was able to find the global optimum on some test functions [47], [45].

The CMA-ES has several *invariance properties*. Two of them are:

- Invariance to order preserving (i.e. strictly monotonic) transformations of the objective function value. For example, the performance of CMA-ES when optimizing $\|\boldsymbol{x}\|^2$ or $3\|\boldsymbol{x}\|^{0.2} - 100$ is identical.
- Invariance to angle preserving (rigid) transformations of the search space (including rotation, reflection, and translation), if the same transformation is applied to the initial search point.

Invariance properties are very important since they imply uniform behavior on classes of functions and therefore allow the generalization of empirical results.

The user is not required to tune strategy internal parameters. Setting of good (default) strategy parameters is part of the algorithm itself. The default population size λ is comparatively small to allow for fast convergence. Restarts with increasing population size improve the global search performance [2].

For a comprehensive introduction to the CMA-ES, we suggest to go through the tutorial slides presented by Anne Auger and Nikolaus Hansen at the Genetic and Evolutionary Computation Conference (GECCO) 2011, available at `http://www.lri.fr/~hansen/cmaesintro.html`.

Before introducing two variants of the CMA-ES in Section 3.2.4 and Section 3.2.5, a few required fundamentals are summed up.

3.2.1 *Eigenvalue Decomposition of a Positive Definite Matrix*

A symmetric, positive definite matrix, $\mathbf{C} \in \mathbb{R}^{n \times n}$, is characterized in that

$$\boldsymbol{x}^{\mathrm{T}}\mathbf{C}\boldsymbol{x} > 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$$

The matrix $\mathbf{C}$ has an orthonormal basis of eigenvectors, $\mathbf{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$, with corresponding eigenvalues $d_1^2, \ldots, d_n^2 > 0$. It can then be written as

$$\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^{\mathrm{T}}, \tag{3.1}$$

where

- $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$ is the diagonal matrix with square roots of eigenvalues of $\mathbf{C}$ as diagonal elements.
- $\mathbf{D}^2 = \mathbf{D}\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)^2 = \mathrm{diag}(d_1^2, \ldots, d_n^2)$ is a diagonal matrix with eigenvalues of $\mathbf{C}$ as diagonal elements.

The matrix decomposition (3.1) is unique, apart from signs of column of $\mathbf{B}$ and permutations of columns in $\mathbf{B}$ and $\mathbf{D}^2$ respectively, given all eigenvalues are different.

Given the decomposition (3.1), the inverse $\mathbf{C}^{-1}$ can be computed via

$$\begin{aligned}\mathbf{C}^{-1} &= (\mathbf{B}\mathbf{D}^2\mathbf{B}^\mathrm{T})^{-1} \\ &= (\mathbf{B}^\mathrm{T})^{-1}\mathbf{D}^{-2}\mathbf{B}^{-1} \\ &= \mathbf{B}\mathbf{D}^{-2}\mathbf{B}^\mathrm{T} \\ &= \mathbf{B}\,\mathrm{diag}\Big(\frac{1}{d_1^2},\ldots,\frac{1}{d_n^2}\Big).\end{aligned}$$

The square root of $\mathbf{C}$ is defined as

$$\mathbf{C}^{\frac{1}{2}} = \mathbf{B}\mathbf{D}\mathbf{B}^\mathrm{T}$$

and therefore

$$\begin{aligned}\mathbf{C}^{-\frac{1}{2}} &= \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^\mathrm{T} \\ &= \mathbf{B}\,\mathrm{diag}\Big(\frac{1}{d_1},\ldots,\frac{1}{d_n}\Big)\mathbf{B}^\mathrm{T}.\end{aligned}$$

3.2.2 *The Multivariate Normal Distribution*

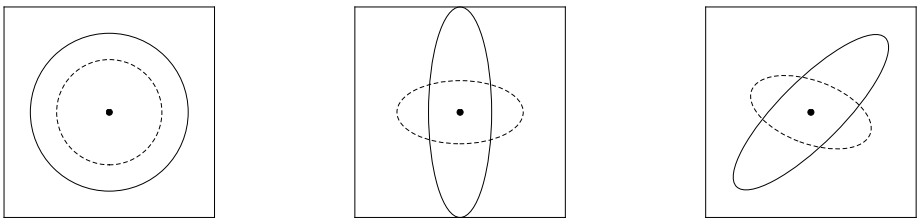
A multivariate normal distribution, $N(\boldsymbol{m},\mathbf{C})$, has a unimodal, "bell-shaped" density, where the top of the bell corresponds to the distribution mean, $\boldsymbol{m}$. The distribution $N(\boldsymbol{m},\mathbf{C})$ is uniquely determined by its mean $\boldsymbol{m}\in\mathbb{R}^n$ and its symmetric and positive definite covariance matrix $\mathbf{C}\in\mathbb{R}^{n\times n}$. Covariance (positive definite) matrices can be identified with the (hyper-)ellipsoid $\{\,\boldsymbol{x}\in\mathbb{R}^n \mid \boldsymbol{x}^\mathrm{T}\mathbf{C}^{-1}\boldsymbol{x}=1\,\}$, as shown in Figure 51. The ellipsoid is a surface of equal density of the distribution. The principal axes of the ellipsoid correspond to the eigenvectors of $\mathbf{C}$, the squared axes lengths correspond to the eigenvalues. The eigenvalue decomposition is given by (3.1). If $\mathbf{D}=\sigma\mathbf{I}$, where $\sigma\in\mathbb{R}^+$ and $\mathbf{I}$ denotes the identity matrix, $\mathbf{C}=\sigma^2\mathbf{I}$ and the ellipsoid is isotropic (see Figure 51a). If $\mathbf{B}=\mathbf{I}$, then $\mathbf{C}=\mathbf{D}^2$ is a diagonal matrix and the ellipsoid is axis parallel oriented (see Figure 51b). In the coordinate system given by the columns of $\mathbf{B}$, the distribution $N(\mathbf{0},\mathbf{C})$ is always uncorrelated.

The normal distribution $N(\boldsymbol{m},\mathbf{C})$ can be written in different ways

$$\begin{aligned}N(\boldsymbol{m},\mathbf{C}) &\sim \boldsymbol{m}+N(\mathbf{0},\mathbf{C}) \\ &\sim \boldsymbol{m}+\mathbf{C}^{\frac{1}{2}}N(\mathbf{0},\mathbf{I}) \\ &\sim \boldsymbol{m}+\mathbf{B}\mathbf{D}\underbrace{\mathbf{B}^\mathrm{T}N(\mathbf{0},\mathbf{I})}_{\sim N(\mathbf{0},\mathbf{I})} \\ &\sim \boldsymbol{m}+\mathbf{B}\underbrace{\mathbf{D}N(\mathbf{0},\mathbf{I})}_{\sim N(\mathbf{0},\mathbf{D}^2)},\end{aligned} \tag{3.2}$$

where "$\sim$" denotes equality in distribution. The last row can be well interpreted, from right to left

- $N(\mathbf{0},\mathbf{I})$ produces an isotropic distribution as in Figure 51a.
- $\mathbf{D}$ scales the isotropic distribution within the coordinate axes as in Figure 51b. $\mathbf{D}N(\mathbf{0},\mathbf{I}) \sim N(\mathbf{0},\mathbf{D}^2)$ has n independent components. The matrix $\mathbf{D}$ can be interpreted as (individual) step-size matrix and its diagonal entries are the standard deviations of the components.



(a) $N(\mathbf{0}, \sigma^2\mathbf{I})$ (b) $N(\mathbf{0}, \mathbf{D}^2)$ (c) $N(\mathbf{0}, \mathbf{C})$

Figure 51: Ellipsoids depicting one-σ lines of equal density of six different normal distributions, where $\sigma \in \mathbb{R}^+$, $\mathbf{D}$ is a diagonal matrix, and $\mathbf{C}$ is a positive definite full covariance matrix.

- $\mathbf{B}$ defines a new orientation for the ellipsoid, where the new principal axes of the ellipsoid correspond to the columns of $\mathbf{B}$.

Equation (3.2) is useful to compute $N(\boldsymbol{m}, \mathbf{C})$ distributed vectors, because $N(\mathbf{0}, \mathbf{I})$ is a vector of independent $(0, 1)$-normally distributed numbers that can easily be realized on a computer.

3.2.3 *Randomized Black Box Optimization*

Let us consider the problem of minimization of a single objective function

$$z\colon \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{x} \mapsto z(\boldsymbol{x}).$$

Black box optimization refers to the situation where function values of evaluated search points are the only accessible information on z. This situation may happen when the search landscape is rugged because it comprises discontinuities, sharp ridges, local optima, etc.

Stochastic black box search algorithms are regarded to be robust in such difficult situations. The CMA-ES belongs to this class of algorithms. In particular, it is designed to tackle, additionally, ill-conditioned and non-separable problems, i.e. problems which cannot be solved by solving 1-dimensional problems separately.

A stochastic black box search algorithm is outlined in Algorithm 3.1. In the CMA-ES the search distribution, P is a multivariate normal distribution. Given all variances and covariances, the normal distribution has the largest entropy of all distributions in $\mathbb{R}^n$. Furthermore, since it is isotropic, it does not favor any coordinate direction. Both makes the normal distribution a particularly attractive tool for randomized search.

According to the notation introduced in Section 3.1, let μ be the number of parents and λ the number of offspring of an evolution strategy. The (μ, λ)-CMA-ES is presented in Section 3.2.4. We used this variant of the CMA-ES to perform single objective optimizations. Section 3.2.5 presents the $(1 + \lambda)$-CMA-ES, which is an elitist algorithm used to develop an adaptation of the CMA-ES to multi-objective optimization.

Algorithm 3.1 Stochastic black box search.

Input: objective function z, search distribution P, function for updating strategy parameters F_θ
Output: local (possibly global) optimum

1: Initialize distribution parameters $\boldsymbol{\theta}^{(0)}$
2: Set population size $\lambda \in \mathbb{N}$
3: Initialize generation counter t, $t \leftarrow 0$
4: **repeat**
5: Sample λ independent points from distribution $P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)}) \rightarrow \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$
6: Evaluate the sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on z
7: Update parameters $\boldsymbol{\theta}^{(t+1)} \leftarrow F_\theta(\boldsymbol{\theta}^{(t)}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, z(\boldsymbol{x}_1), \ldots, z(\boldsymbol{x}_\lambda))$
8: $t \leftarrow t + 1$
9: **until** termination criterion is met

3.2.4 *The non-elitist CMA-ES with weighted recombination*

In this section, a concise description of the (μ, λ)-CMA-ES is given. For a full description of this algorithm, see [48]. The following nomenclature is used:

- $\mu \leq \lambda$ is the parent population size, i.e. the number of points selected to compute the new mean $\boldsymbol{m}^{(t+1)}$ of the search distribution.
- $\boldsymbol{y}_k \sim N(\mathbf{0}, \mathbf{C})$, for $k = 1, \ldots, \lambda$, are realizations from a multivariate normal distribution with zero mean and covariance matrix $\mathbf{C}$.
- $\mathbf{B}, \mathbf{D}$ result from an eigenvalue decomposition of the covariance matrix $\mathbf{C}$ with $\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^{\mathrm{T}}$ (see Section 3.2.1).
- $\boldsymbol{x} \in \mathbb{R}^n$, for $k = 1, \ldots, \lambda$. Sample of λ search points.
- $w_{i=1,\ldots,\mu} \in \mathbb{R}^+$ are positive weight coefficients for recombination.
- $\langle \boldsymbol{y} \rangle_{\mathrm{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}$ step of the distribution mean disregarding step-size σ.
- $\boldsymbol{y}_{i:\lambda} = (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})/\sigma$, see $\boldsymbol{x}_{i:\lambda}$ below.
- $\boldsymbol{x}_{i:\lambda} \in \mathbb{R}^n$, ith best point out of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ from Algorithm 3.2, line 8. The index $i : \lambda$ denotes the index of the ith ranked point, that is $z(\boldsymbol{x}_{1:\lambda}) \leq z(\boldsymbol{x}_{2:\lambda}) \leq \cdots \leq z(\boldsymbol{x}_{\lambda:\lambda})$.
- $\mu_{\mathrm{eff}} = \left(\sum_{i=1}^{\mu} w_i^2\right)^{-1}$ is the variance effective selection mass, see [48]. It holds $1 \leq \mu_{\mathrm{eff}} \leq \mu$.
- $\mathbf{C}^{-\frac{1}{2}} = \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^{\mathrm{T}}$, see $\mathbf{B}, \mathbf{D}$ above. From the definitions it follows

$$\mathbf{C}^{-\frac{1}{2}} \langle \boldsymbol{y} \rangle_{\mathrm{w}} = \mathbf{B} \langle \boldsymbol{g} \rangle_{\mathrm{w}}$$

  with $\langle \boldsymbol{g} \rangle_{\mathrm{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{g}_{i:\lambda}$.
- $\mathrm{E}(\cdot)$ is the expected value of a random variable. In particular

$$\mathrm{E}(\|N(\mathbf{0}, \mathbf{I})\|) = \sqrt{2}\Gamma\left(\frac{n+1}{2}\right)/\Gamma\left(\frac{n}{2}\right) \approx \sqrt{n}\left(1 - \frac{1}{4n} + \frac{1}{21n^2}\right).$$

- $h_\sigma = \begin{cases} 1 & \text{if } \frac{\|\boldsymbol{p}_\sigma\|}{\sqrt{1-(1-c_\sigma)^{2(t+1)}}} < \left(1.4 + \frac{2}{n+1}\right)\mathrm{E}(\|N(\mathbf{0},\mathbf{I})\|) \\ 0 & \text{otherwise} \end{cases}$, where t is the generation number. The Heaviside function h_σ stalls the update of $\boldsymbol{p}_c$ in Algorithm 3.2, line 14, if $\|\boldsymbol{p}_\sigma\|$ is large. This prevents a too fast increase of axes of $\mathbf{C}$ in a linear surrounding, i.e. when the step-size is far too small. This is useful when the initial step-size is chosen far too small or when the objective function changes in time.

- $\delta(h_\sigma) = (1 - h_\sigma)c_c(2 - c_c) \leq 1$ is of minor relevance. In the (unusual) case of $h_\sigma = 0$, it substitutes for the second summand from Algorithm 3.2, line 15.

Algorithm 3.2 $(\mu/\mu_\mathrm{w}, \lambda)$-CMA-ES

Input: Objective function z
Output: A local minimum of the objective function

1: Set parameters $\lambda, \mu, w_{i=1,\ldots,\mu}, c_\sigma, d_\sigma, c_c, c_1$, and c_μ to their default values according to Table 7
2: Initialize evolution paths $\boldsymbol{p}_\sigma \leftarrow \mathbf{0}, \boldsymbol{p}_c \leftarrow \mathbf{0}$, covariance matrix $\mathbf{C} \leftarrow \mathbf{I}$
3: Initialize generation counter $t \leftarrow 0$
4: **repeat**
5: **for** $k = 1, \ldots, \lambda$ **do** ▷ Sample new population of search points
6: $\boldsymbol{g}_k \sim N(\mathbf{0}, \mathbf{I})$
7: $\boldsymbol{y}_k \leftarrow \mathbf{B}\mathbf{D}\boldsymbol{g}_k \sim N(\mathbf{0}, \mathbf{C})$
8: $\boldsymbol{x}_k \leftarrow \boldsymbol{m} + \sigma\boldsymbol{y}_k \sim N(\boldsymbol{m}, \sigma^2\mathbf{C})$
9: **end for**
Selection and recombination
10: $\langle \boldsymbol{y} \rangle_\mathrm{w} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}$ where $\sum_{i=1}^{\mu} w_i = 1. w_i > 0$
11: $\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \langle \boldsymbol{y} \rangle_\mathrm{w} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$
Step-size control
12: $\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_\mathrm{eff}}\mathbf{C}^{-\frac{1}{2}} \langle \boldsymbol{y} \rangle_\mathrm{w}$
13: $\sigma \leftarrow \sigma \cdot \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathrm{E}(\|N(\mathbf{0},\mathbf{I})\|)} - 1\right)\right)$
Covariance matrix adaptation
14: $\boldsymbol{p}_c \leftarrow (1 - c_c)\boldsymbol{p}_c + h_\sigma\sqrt{c_c(2 - c_c)\mu_\mathrm{eff}} \langle \boldsymbol{y} \rangle_\mathrm{w}$
15: $\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\mathbf{C} + c_1(\boldsymbol{p}_c\boldsymbol{p}^\mathrm{T} + \delta(h_\sigma)\mathbf{C}) + c_\mu \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^\mathrm{T}$
16: $t \leftarrow t + 1$
17: **until** termination criterion is met

Selection and recombination:

$$\lambda = 4 + \lfloor 3 \ln n \rfloor, \quad \mu' = \frac{\lambda}{2}, \quad \mu = \lfloor \mu' \rfloor$$
$$w_i = \frac{w_i'}{\sum_{j=1}^{\mu} w_j'}, \quad w_i' = \ln(\mu' + 0.5) - \ln i \quad i = 1, \ldots, \mu$$

Step-size control:

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 5}, \quad d_\sigma = 1 + 2 \max\left(0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n+1}} - 1\right) + c_\sigma$$

Covariance matrix adaptation:

$$c_c = \frac{4 + \mu_{\text{eff}}/n}{n + 4 + 2\mu_{\text{eff}}/n}$$
$$c_1 = \frac{2}{(n+1.3)^2 + \mu_{\text{eff}}}$$
$$c_\mu = \min\left(1 - c_1, 2\frac{\mu_{\text{eff}} - 2 + 1/\mu_{\text{eff}}}{(n+2)^2 + \mu_{\text{eff}}}\right)$$

Table 7: Default strategy parameters of the non-elitist CMA, where $\mu_{\text{eff}} = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1$ and $\sum_{i=1}^{\mu} w_i = 1$, taken from [48].

The CMA-ES overcomes typical problems that are often associated with evolutionary algorithms.

1. Poor performance on badly scaled and/or highly non-separable objective functions. Line 15 adapts the search distribution to badly scaled and non-separable problems.

2. The need to use large population sizes. A strategy which is often used by evolutionary algorithms in order to prevent the degeneration of the population into a subspace is using large population sizes. In the CMA-ES, the population size can be freely chosen, because the learning rates c_1 and c_μ in line 15 prevent the degeneration even for small population sizes, e.g. $\lambda = 9$. Small population sizes usually allow to achieve faster convergence, large population sizes help to avoid local optima.

3. Premature convergence of the population. Step-size control in line 13 prevents the population to converge prematurely. It does not prevent the search to be trapped in a local optimum.

3.2.5 *A Single-Objective Elitist CMA Evolution Strategy*

In this section a single-objective, elitist CMA-ES with $(1 + \lambda)$-selection is presented. This algorithm is the starting point for developing a variant of the CMA-ES for multi-objective optimization. For a detailed study of this algorithm, see [53].

In the $(1 + \lambda)$-CMA-ES, the following nomenclature is used:

- Each individual a, is a 5-tuple $a = [\boldsymbol{x}, \bar{p}_{\text{succ}}, \sigma, \boldsymbol{p}_c, \mathbf{C}]$ comprising its candidate solution vector $\boldsymbol{x} \in \mathbb{R}^n$, an average success rate $\bar{p}_{\text{succ}} \in [0, 1]$, the global step size $\sigma \in \mathbb{R}^+$, an evolution path $\boldsymbol{p}_c \in \mathbb{R}^n$, and the covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$.
- $z \colon \mathbb{R}^n \to \mathbb{R}, \boldsymbol{x} \mapsto z(\boldsymbol{x})$ is the objective function to be minimized.
- $\lambda_{\text{succ}}^{(t+1)} = \left| \left\{ i = 1, \ldots, \lambda \;\middle|\; z(\boldsymbol{x}_i^{(t+1)}) \leq z(\boldsymbol{x}_{\text{parent}}^{(t)}) \right\} \right|$ is the number of successful offspring.
- $N(\boldsymbol{m}, \mathbf{C})$ is a multi-variate normal distribution with mean vector $\boldsymbol{m}$ and covariance matrix $\mathbf{C}$. The notation $\boldsymbol{x} \sim N(\boldsymbol{m}, \mathbf{C})$ denotes that random variable $\boldsymbol{x}$ is distributed according to the distribution $N(\boldsymbol{m}, \mathbf{C})$.
- $\boldsymbol{x}_{1:\lambda}^{(t)} \in \mathbb{R}^n$ is the best point from $\left\{ \boldsymbol{x}_1^{(t)}, \ldots, \boldsymbol{x}_\lambda^{(t)} \right\}$, that is, $z(\boldsymbol{x}_{1:\lambda}^{(t)}) \leq z(\boldsymbol{x}_i^{(t)})$ for all $i = 1, \ldots, \lambda$.

Algorithm 3.3 $(1 + \lambda)$-CMA-ES

Input: Number of offspring λ, damping parameter d, target success probability $p_{\text{succ}}^{\text{target}}$, success rate averaging parameter c_p, cumulation time horizon parameter c_c, covariance matrix learning rate c_{cov}, objective function z

Output: A minimum of the objective function

1: $t \leftarrow 0$
2: Initialize $a_{\text{parent}}^{(t)}$
3: **repeat**
4: $\quad a_{\text{parent}}^{(t+1)} \leftarrow a_{\text{parent}}^{(t)}$
5: $\quad$ **for** $k = 1, \ldots, \lambda$ **do**
6: $\quad\quad \boldsymbol{x}_k^{(t+1)} \sim N(\boldsymbol{x}_{\text{parent}}^{(t)}, {\sigma^{(t)}}^2 \mathbf{C}^{(t)})$
7: $\quad$ **end for**
8: $\quad$ `updateStepSize`$\left(a_{\text{parent}}^{(t+1)}, \frac{\lambda_{\text{succ}}^{(t+1)}}{\lambda}\right)$
9: $\quad$ **if** $z(\boldsymbol{x}_{1:\lambda}^{(t+1)}) \leq z(\boldsymbol{x}_{\text{parent}}^{(t)})$ **then**
10: $\quad\quad \boldsymbol{x}_{\text{parent}}^{(t+1)} \leftarrow \boldsymbol{x}_{1:\lambda}^{(t+1)}$
11: $\quad\quad$ `updateCovariance`$\left(a_{\text{parent}}^{(t+1)}, \frac{\boldsymbol{x}_{\text{parent}}^{(t+1)} - \boldsymbol{x}_{\text{parent}}^{(t)}}{\sigma_{\text{parent}}^{(t)}}\right)$
12: $\quad$ **end if**
13: $\quad t \leftarrow t + 1$
14: **until** termination criterion is met

The $(1 + \lambda)$-CMA-ES is given in Algorithm 3.3. It works as follows:

- λ new candidate solutions are sampled, lines 5–7.

Algorithm 3.4 updateStepSize$(a = [\boldsymbol{x}, \bar{p}_{\text{succ}}, \sigma, \boldsymbol{p}_c, \mathbf{C}], p_{\text{succ}})$

1: $\bar{p}_{\text{succ}} \leftarrow (1 - c_p)\bar{p}_{\text{succ}} + c_p p_{\text{succ}}$
2: $\sigma \leftarrow \sigma \cdot \exp\left(\frac{1}{d} \frac{\bar{p}_{\text{succ}} - p_{\text{succ}}^{\text{target}}}{1 - p_{\text{succ}}^{\text{target}}}\right)$

- The step size is updated based on the success rate $p_{\text{succ}} = \lambda_{\text{succ}}^{(t+1)}/\lambda$ with a learning rate $c_p (0 < c_p \leq 1)$, line 8. This update rule is shown in Algorithm 3.4. It implements the well-known heuristic that the step size should be increased if the success rate (i.e., the fraction of offspring better than the parent) is high, and the step size should be decreased if the success rate is low. The rule is reflected in the argument to the exponential function. For $\bar{p}_{\text{succ}} > p_{\text{succ}}^{\text{target}}$ the argument is greater than zero and the step size increases; for $\bar{p}_{\text{succ}} < p_{\text{succ}}^{\text{target}}$ the argument is smaller than zero and the step size decreases; for $\bar{p}_{\text{succ}} = p_{\text{succ}}^{\text{target}}$ the argument becomes zero and no change of σ takes place.

Algorithm 3.5 updateCovariance$(a = [\boldsymbol{x}, \bar{p}_{\text{succ}}, \sigma, \boldsymbol{p}_c, \mathbf{C}], \boldsymbol{x}_{\text{step}} \in \mathbb{R}^n)$

1: **if** $\bar{p}_{\text{succ}} < p_{\text{thresh}}$ **then**
2: $\quad \boldsymbol{p}_c \leftarrow (1 - c_c)\boldsymbol{p}_c + \sqrt{c_c(2 - c_c)}\boldsymbol{x}_{\text{step}}$
3: $\quad \mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \cdot \boldsymbol{p}_c \boldsymbol{p}_c^{\mathrm{T}}$
4: **else**
5: $\quad \boldsymbol{p}_c \leftarrow (1 - c_c)\boldsymbol{p}_c$
6: $\quad \mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \cdot (\boldsymbol{p}_c \boldsymbol{p}_c^{\mathrm{T}} + c_c(2 - c_c)\mathbf{C})$
7: **end if**

- If the *best* new candidate solution was better than the parent individual (see line 9), the covariance matrix is updated, line 11. The covariance update method is given in Algorithm 3.5. The update of the evolution path $\boldsymbol{p}_c$ depends on the value of $\bar{p}_{\text{succ}}$. If the smoothed success rate $\bar{p}_{\text{succ}}$ is above $p_{\text{thresh}} < 0.5$, the update of the evolution path is stalled. This prevents a too fast increase of axes of $\mathbf{C}$ when the step size is far too small. If the smoothed success rate $\bar{p}_{\text{succ}}$ is low, the update of $\boldsymbol{p}_c$ is given by an exponential smoothing. The constants c_c and c_{cov} $(0 \leq c_{\text{cov}} < c_c \leq 1)$ are learning rates for the evolution path and the covariance matrix, respectively. The evolution path is used to update the covariance matrix. The new covariance matrix is a weighted mean of the old covariance matrix and the outer product of $\boldsymbol{p}_c$. In the second case (Alg. 3.5, line 5), the second summand in the update of $\boldsymbol{p}_c$ is missing and the length of $\boldsymbol{p}_c$ shrinks. The term $c_c(2 - c_c)\mathbf{C}$ (Alg. 3.5, line 6) compensates for this shrinking in $\mathbf{C}$.

Simulation results of the $(1 + \lambda)$-CMA-ES are shown in [53].

3.3 USING MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

Evolutionary algorithms for dealing with multiobjective optimization problems, or MOEAs, are metaheuristics which deal simultaneously with a set of possible solutions (the population). This allows to find several members of the Pareto optimal set in a single "run" of the algorithm, instead of having to perform a series of separate runs. Evolutionary algorithms are less susceptible to the shape or continuity of the Pareto front (e.g. they can easily deal with discontinuous or concave Pareto fronts), whereas these two issues are a real concern for mathematical programming

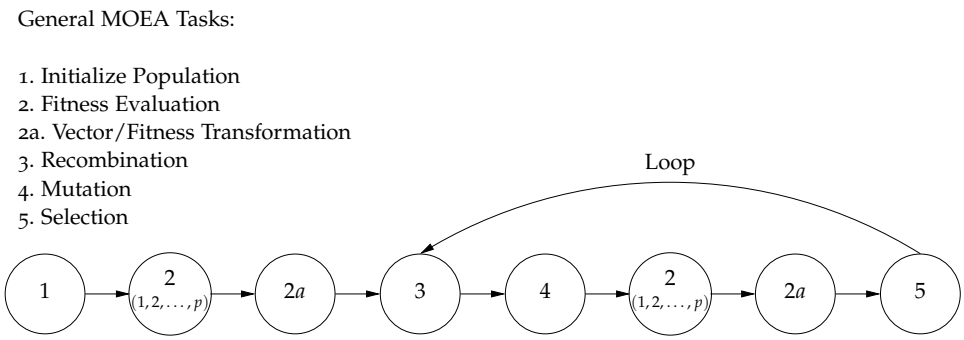


Figure 52: MOEA task decomposition.

techniques, see Section 2.3.1.1, for example. Additionally, they have the ability to find several alternative trade-offs.

Figure 52 shows a general MOEA's task decomposition. Task 2 computes p (where $p \geq 2$) fitness functions. In addition, because MOEAs expect a *single* fitness value with which to perform selection, processing is sometimes required to transform MOEA solutions' fitness *vectors* into a scalar (task 2*a*).

3.3.1 *Pareto Notation*

During MOEA execution, a "current" set of Pareto optimal solutions (with respect to the *current* MOEA generational population) is determined at each EA generation and termed $P_{\text{current}}(t)$, where t represents the generation number. Some MOEA implementations which will be presented also use a secondary population storing nondominated solutions found through the generations [11]. Corresponding vectors of this set must be periodically tested and solutions whose associated vectors are dominated removed.

This secondary population is named $P_{\text{known}}(t)$. This term is also annotated with t to reflect its possible changes in membership during MOEA execution. $P_{\text{known}}(0)$ is defined as the empty set ($\emptyset$) and P_{known} alone as the *final* set of solutions returned by the MOEA at termination. Different secondary population storage strategies exist; the simplest is when $P_{\text{current}}(t)$ is added at each generation (i.e., $P_{\text{current}}(t) \cup P_{\text{known}}(t-1)$). At any given time, $P_{\text{known}}(t)$ is thus the set of Pareto optimal solutions *yet found by the MOEA through generation* t. The *true* Pareto optimal set (termed P_{true}) is not explicitly known for problems of any difficulty.

$P_{\text{current}}(t)$, P_{known} and P_{true} are sets of MOEA genotypes; each set's corresponding phenotypes form a Pareto front. The associated Pareto front for each of these solution sets is called $PF_{\text{current}}(t)$, PF_{known} and PF_{true}.

3.4 DESIGN ISSUES AND COMPONENTS OF MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

In this section, we focus on important issues while designing a MOEA. We will make use of notation introduced in Section 3.3.1.

The ultimate goal of a multi-objective optimization algorithm is to identify solutions in the Pareto optimal set. Achieving the exact Pareto front of an arbitrary

problem is usually quite difficult. MOEAs try to approximate it as well as possible. With this concern in mind, a MOEA should achieve the following conflicting goals [108]:

GOAL 1. Progress towards points on PF_{true}. The best known Pareto front should be as close as possible to the true Pareto front.

GOAL 2. Maintain diversity of: points on PF_{known} and/or on P_{known}. Solutions in the best-known Pareto front, PF_{known} (phenotype), and/or in the best-known Pareto optimal set, P_{known} (genotype), should be uniformly distributed in order to provide the decision-maker a true picture of trade-offs.

GOAL 3. Provide the decision-maker with a limited number of PF_{known} points.

In Table 8, highlights of the MOEAs available in the Python library for parameter extraction with their advantages and disadvantages are given. In order to introduce these MOEAs, we describe the techniques they implement to attain the goals in multi-objective optimization. These techniques are listed below as follows according to their support to such goals:

GOAL 1. Progress towards points on PF_{true}.

- Dominance-based ranking.
- Contributing hypervolume.
- Elitism.

GOAL 2. Maintain diversity of: points on PF_{known} and/or on P_{known}.

- Fitness sharing/niching.
- Crowding distance.
- Contributing hypervolume.

Let us introduce some notations which will be used in the following sections:

- P_t is the population at generation t;
- N is the population size;
- z_q is the qth objective function;
- p is the number of objectives.

3.4.1 *Dominance-based ranking*

Dominance-based ranking approaches explicitly utilize the concept of Pareto dominance in evaluating fitness or assigning selection probability to solutions. The population is ranked according to a dominance rule, and then each solution is assigned a fitness value based on its rank in the population. A lower rank corresponds to a better solution in the following discussions.

The following procedure is an example of a Pareto ranking technique which sorts a population into different nondomination levels:

ALGORITHM	FITNESS ASSIGNMENT	DIVERSITY MECHANISM	ELITISM	EXTERNAL POPULATION	ADVANTAGES	DISADVANTAGES
NSGA-II [24]	Ranking based on non-domination sorting	Crowding distance	Yes	No	Single parameter (N), well tested, efficient	Crowding distance works in objective space only
SPEA2 [109]	Strength of dominators	Density based on the k-th nearest neighbor	Yes	Yes	Well tested, no parameter for clustering, extreme points are preserved	Computationally expensive fitness and density calculation
SPEA2+ [62]	Strength of dominators	Density based on the k-th nearest neighbor	Yes	Yes	Improved SPEA2, density calculation in both objective and decision space	Computationally expensive fitness and density calculation
PAES [63]	Pareto dominance is used to replace a parent if offspring dominates	Cell-based density as tie breaker between offspring and parent	Yes	Yes	Random mutation hill-climbing strategy, easy to implement, computationally efficient	Not a population based approach, performance depends on cell sizes
MO-CMA-ES [53]	Ranking based on non-domination sorting and contributing hypervolume	Contributing hypervolume	Yes	Yes	Invariance against rotation of the search space	Hypervolume computation exponential in the number of objectives

Table 8: List of MOEAs implemented in the Python library for parameter extraction.

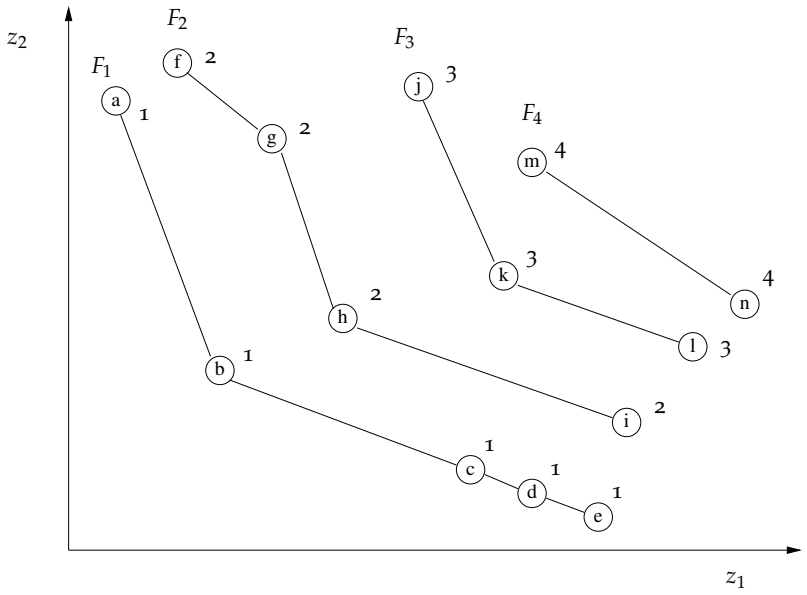


Figure 53: Example of NSGA-II ranking.

STEP 1. Set $i = 1$ and $TP = P_t$.

STEP 2. Identify nondominated solutions in TP and assign them to F_i.

STEP 3. For every solution $\boldsymbol{x} \in F_i$ assign rank $r_1(\boldsymbol{x}, t) = i$.

STEP 4. Set $TP = TP \setminus F_i$. If $TP = \emptyset$, stop, else set $i = i + 1$ and go to Step 2.

In the procedure above, $F_1, F_2, \ldots,$ are called *nondominated fronts*, and F_1 is the Pareto front of population P_t. If it is not carefully implemented, its worst-case complexity will be $O(Np^3)$ [24].

The Non Dominated Sorting Genetic Algorithm II (NSGA-II) [24] uses a more efficient nondominated sorting procedure with $O(Mp^2)$ computational complexity. Figure 53 illustrates an example of the NSGA-II ranking method. The same procedure is adopted in the Multi-Objective Covariance Matrix Adaption Evolution Strategy (MO-CMA-ES) [53].

In the Strength Pareto Evolutionary Algorithm 2 (SPEA2), an external archive A_t of a fixed size stores nondominated solutions that have been investigated up to generation t [109]. Each individual $\boldsymbol{x}$ in the archive A_t and the population P_t is assigned a strength value $s(\boldsymbol{x}, t)$, representing the number of solutions it dominates

$$s(\boldsymbol{x}, t) = |\{\, \boldsymbol{y} \mid \boldsymbol{y} \in P_t \cup A_t \wedge \boldsymbol{i} \preceq \boldsymbol{j} \,\}|$$

where $|\cdot|$ denotes the cardinality of a set and the symbol $\preceq$ corresponds to the Pareto dominance relation (see Section 2.1.4). On the basis of the s values, the *raw* fitness $r_2(\boldsymbol{x}, t)$ of an individual $\boldsymbol{x}$ is calculated as

$$r_2(\boldsymbol{x}, t) = \sum_{\boldsymbol{j} \in P_t \cup A_t, \boldsymbol{j} \preceq \boldsymbol{i}} s(\boldsymbol{j}, t).$$

Figure 54 shows an example of the computation of raw ranks used by the SPEA2. A similar ranking method has been used in the Improved Strength Pareto Evolutionary Algorithm 2 (SPEA2+) [62].

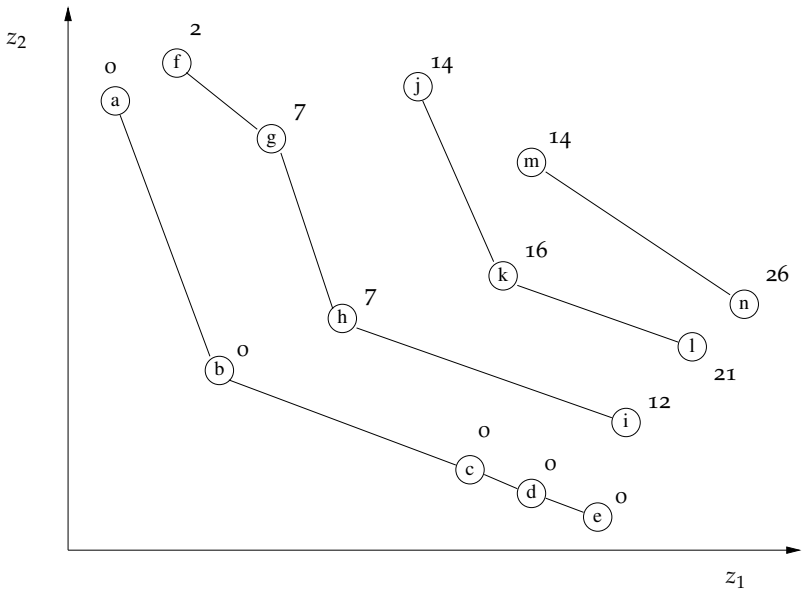


Figure 54: Example of SPEA2 raw rank computation.

Like the other MOEAs, the Pareto Archived Evolution Strategy (PAES) [63] compares solutions according to a dominance rule.

3.4.2 *Diversity*

Maintaining a diverse population is an important consideration in MOEAs to obtain solutions uniformly distributed over the Pareto front. Without taking preventive measures, the population tends to form relatively few clusters in MOEAs. This phenomenon is called *genetic drift* [11]. We will describe the approaches for preventing genetic drift implemented in the MOEAs considered in this work.

FITNESS SHARING/NICHING. In this case, one must count how many solutions are located within the same neighborhood (or *niche*), and the fitness is decreased proportionally to the number of individuals sharing the same neighborhood.

In the SPEA2 [109] the density estimation is given by the inverse of the distance to the kth nearest neighbor. For each individual $\boldsymbol{x}$ the distances (in objective space) to all individuals $\boldsymbol{y}$ in archive and population are calculated and stored in a list. After sorting the list in increasing order, the kth element gives the distance sought, denoted as σ_i^k. Parameter k can be specified by the user. Otherwise is taken as follows: $k = \sqrt{N + \bar{N}}$, being $\bar{N}$ the size of the archive. Afterwards, the density $d(\boldsymbol{x}, t)$ corresponding to $\boldsymbol{x}$ is defined by

$$d(\boldsymbol{x}, t) = \frac{1}{\sigma_i^k + 2}.$$

The final fitness of an individual $\boldsymbol{x}$ is then computed as $f(\boldsymbol{x}, t) = d(\boldsymbol{x}, t) + r_2(\boldsymbol{x}, t)$.

The SPEA2+ [62] tries to maintain diversity of the solution in the objective and the decision space. Nondominated solutions are stored in two external

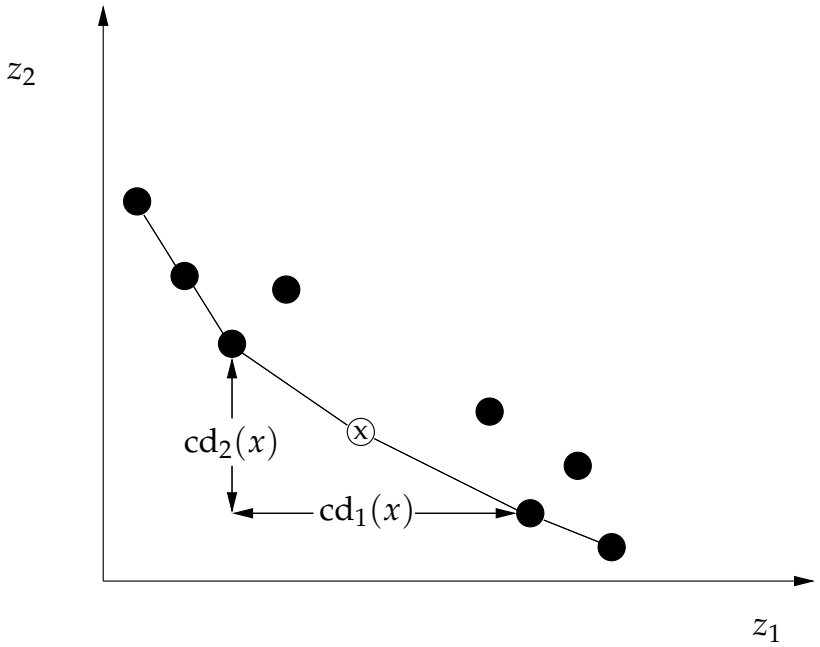


Figure 55: Definition of crowding distance.

archives A_t^O, A_t^V. Density estimation is performed following the same method used in SPEA2, but distances are calculated in the objective space for each individual $\boldsymbol{x} \in A_t^O \cup P_t$ and in the decision space for each individual $\boldsymbol{x} \in A_t^V \cup P_t$. Archive A_t^O is the same as A_t in SPEA2 implementation.

CROWDING DISTANCE. The aim of crowding distance methods is to obtain a uniform spread of solutions along the best-known Pareto front without using a fitness sharing parameter like parameter k used by SPEA2 and SPEA2+. NSGA-II [24] uses a crowding distance approach as follows (see Figure 55):

STEP 1. Rank the population and identify nondominated fronts $F_1, F_2, \ldots, F_R$. For each front $j = 1, \ldots, R$ repeat Steps 2 and 3.

STEP 2. For each objective function q, sort the solutions in F_j in ascending order. Let $l = |F_j|$ and $\boldsymbol{x}_{[1,q]}$ represent the ith solution in the sorted list with respect to the objective function q. Assign $\mathrm{cd}_q(\boldsymbol{x}_{[1,q]}) = \infty$ and $\mathrm{cd}_q(\boldsymbol{x}_{[l,q]}) = \infty$, and for $i = 2, \ldots, l-1$ assign

$$\mathrm{cd}_q(\boldsymbol{x}_{[i,q]}) = \frac{z_q(\boldsymbol{x}_{[\boldsymbol{i+1},\boldsymbol{q}]}) - z_q(\boldsymbol{x}_{[\boldsymbol{i-1},\boldsymbol{q}]})}{z_q^{\max} - z_q^{\min}}.$$

STEP 3. To find the total crowding distance $\mathrm{cd}(\boldsymbol{x})$ of a solution $\boldsymbol{x}$, sum the solution's crowding distances with respect to each objective

$$\mathrm{cd}(\boldsymbol{x}) = \sum_{q=1}^{p} \mathrm{cd}_q(\boldsymbol{x}).$$

The main advantage of this crowding approach is that a measure of population density around a solution is computed without requiring a user-defined parameter such as the kth closest neighbor.

CONTRIBUTING HYPERVOLUME. Let A' be a subset of the decision space and let the nondominated solutions in A' be denoted by A'_E. The hypervolume measure or S-metric can be defined as the Lebesgue measure Λ (i.e., the volume) of the union of hypercuboids in the objective space [11]

$$S_{\boldsymbol{a}_{\text{ref}}}(A') = \Lambda\left(\bigcup_{\boldsymbol{a}\in A'_E} \left\{ (z_1(\boldsymbol{a}'),\ldots,z_p(\boldsymbol{a}')) \;\middle|\; \boldsymbol{a} \preceq \boldsymbol{a}' \preceq \boldsymbol{a}_{\text{ref}} \right\}\right),$$

where $\boldsymbol{a}_{\text{ref}}$ is an appropriately chosen reference point. The contributing hypervolume of a point $\boldsymbol{a} \in A'_E$ is given by

$$\Delta_S(\boldsymbol{a}, A') := S_{\boldsymbol{a}_{\text{ref}}}(A') - S_{\boldsymbol{a}_{\text{ref}}}(A' \setminus \{\boldsymbol{a}\}).$$

The rank $s(\boldsymbol{a}, A')$ of an individual $\boldsymbol{a}$ can be defined recursively based on its contribution to the hypervolume. The individual contributing least to the hypervolume of A' gets the worst rank. The individual contributing least to the hypervolume of A' without the individual with the worst rank is assigned the second worst rank and so on.

For two objectives, this ranking can be calculated efficiently in log-linear time in the number of individuals using appropriate data structures and the equation for $\Delta_S(\boldsymbol{a}, A')$ given by [34]. The scaling behavior is exponential in the number of objectives in the worst case [97].

Let A' be now a set of nondominated solutions with the same level of nondominance. In the MO-CMA-ES, solutions in A' are ranked according to how much they contribute to the hypervolume of A'. This strategy has two effects:

- It promotes diversity of solutions. A uniformly distributed set of solutions would mean a larger hypervolume metric value (hence better) than a set of clustered solutions.
- It pushes the population towards PF_{true}. Consider two nondominated solutions with the same level of nondominance $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. If the contributing hypervolume of $\boldsymbol{x}_1$ is greater than that of $\boldsymbol{x}_2$, selecting $\boldsymbol{x}_1$ instead of $\boldsymbol{x}_2$ will yields a better approximation P_{known} of P_{true}.

CELL-BASED DENSITY. The objective space is divided in p-dimensional cells. The number of solutions in each cell is defined as the density of the cell, and the density of a solution is equal to the density of the cell in which the solution is located. This density information is used to achieve diversity similarly to the fitness sharing approach. This technique has been implemented in PAES [63]. Figure 56 shows a graphical illustration of PAES' grid subdivision.

3.4.3 *Elitism*

Elitism in the context of single-objective EAs means that the best solution found so far during the search always survives to the next generation. In this respect, all nondominated solutions discovered by a MOEA are considered elite solutions. However, implementation of elitism in multi-objective optimization is not as straightforward

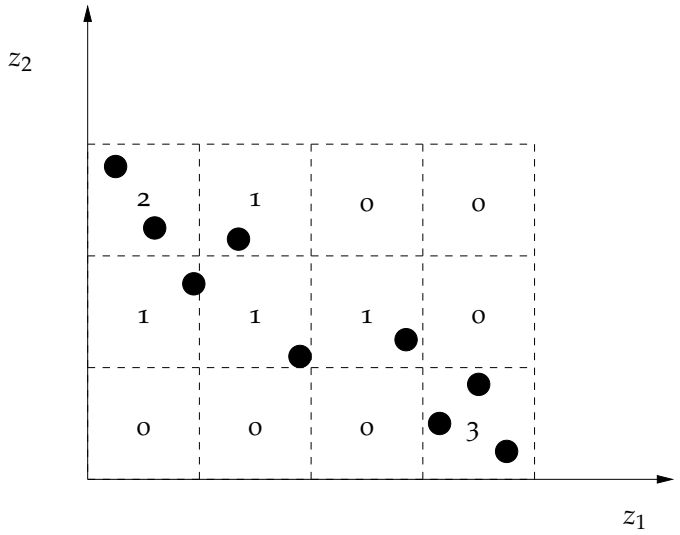


Figure 56: Graphical illustration of the grid subdivision approach used by PAES.

as in single objective optimization mainly due to the large number of possible elitist solutions. As discussed in [107] MOEAs using elitist strategies tend to outperform their non-elitist counterparts. Two strategies have been used to implement elitism:

- maintaining elitist solutions in the population;
- storing elitist solutions in an external secondary list and reintroducing them to the population.

3.4.3.1 *Strategies to maintain elitist solutions in the population*

Random selection does not ensure that a nondominated solution will survive to the next generation. A straightforward implementation of elitism in a MOEA is to copy all nondominated solutions in population P_t to population P_{t+1}, then fill the rest of P_{t+1} by selecting from the remaining dominated solutions in P_t. This will not work when the total number of nondominated parent and offspring solutions is larger than N. We will describe the approaches followed by NSGA-II and MO-CMA-ES to address this problem.

NSGA-II uses a fixed population size of N. In generation t, an offspring population Q_t of size N is created from parent population P_t and nondominated fronts $F_1, F_2, \ldots, F_R$ are identified in the combined population $P_t \cup Q_t$. The next population P_{t+1} is filled starting from solutions in F_1, then F_2, and so on as follows. Let k be the index of a nondominated front F_k such that $|F_1 \cup F_2 \cup \cdots \cup F_k| \leq N$ and $|F_1 \cup F_2 \cup \cdots \cup F_k \cup F_{k+1}| > N$. First, all solutions in fronts $F_1, F_2, \ldots, F_k$ are copied to P_{t+1}, and then the least crowded $(N - |P_{t+1}|)$ solutions in F_{k+1} are added to P_{t+1}. This approach makes sure that all nondominated solutions (F_1) are included in the next population if $|F_1| \leq N$, and the secondary selection based on crowding distance promotes diversity. The population size N is an important parameter.

MO-CMA-ES follows the same strategy but it uses the contributing hypervolume instead of the crowding distance as second sorting criterion.

3.4.3.2 *Elitism with external populations*

When an external list is used to store elitist solutions, several issues must be addressed. The first issue is which solutions are going to be stored in elitist list A. Some MOEAs store nondominated solutions identified so far during the search, and A is updated each time a new solution is created by removing elitist solutions dominated by a new solution or adding the new solution if it is not dominated by any existing elitist solution. This is a computationally expensive operation. Another issue is the size of the list A. Since there might possibly exist a very large number of Pareto optimal solutions for a problem, the elitist list can grow extremely large. Therefore, pruning techniques have been proposed to control the size of A.

Both SPEA2 and SPEA2+ use external lists to store nondominated solutions discovered so far in the search.

Let $\bar{N}$ be the size of the external list A_t. The update operation of A_t in SPEA2 works as follows:

STEP 1. Copy all nondominated individuals, i.e., those which have a fitness lower than one, to the archive of the next generation

$$A_{t+1} = \{ \boldsymbol{x} \mid \boldsymbol{x} \in P_t \cup A_t \wedge f(\boldsymbol{x}, t) < 1 \}.$$

STEP 2. If the nondominated front fits exactly into the archive ($|A_{t+1}| = \bar{N}$), stop. Else, if the archive is too small ($|A_{t+1}| < \bar{N}$), go to Step 3; else, the archive is too large ($|A_{t+1}| > \bar{N}$), then go to Step 4.

STEP 3. Sort $P_t \cup A_t$ according to the fitness value and copy the first $\bar{N} - |A_{t+1}|$ individuals $\boldsymbol{x}$ with $f(\boldsymbol{x}, t) \geq 1$ from the resulting ordered list to A_{t+1}.

STEP 4. Perform archive truncation by iteratively removing individuals from A_{t+1} until $|A_{t+1}| = \bar{N}$. At each stage, the individual which has the minimum distance (in objective space) to another individual is chosen. If there are several individuals with minimum distance the tie is broken by considering the second smallest distances and so forth.

The update operation of archives A_t^O and A_t^V in SPEA2+ is very similar. Let $\bar{N}$ be the size of both A_t^O and A_t^V. Then:

STEP 1. Copy all nondominated individuals in P_t, A_t^O and A_t^V to A_{t+1}^O, A_{t+1}^V.

STEP 2. If the nondominated front fits exactly into the archives ($|A_{t+1}^O|, |A_{t+1}^V| = \bar{N}$), stop. Else, if the archives are too small ($|A_{t+1}^O|, |A_{t+1}^V| < \bar{N}$), go to Step 3; else, the archive are too large ($|A_{t+1}^O|, |A_{t+1}^V| > \bar{N}$), then go to Step 4.

STEP 3. Sort $P_t \cup A_t^O \cup A_t^V$ according to the fitness value and copy the first $\bar{N} - |A_{t+1}^O|$ individuals $\boldsymbol{x}$ with $f(\boldsymbol{x}, t) \geq 1$ from the resulting ordered list to A_{t+1}^O, A_{t+1}^V.

STEP 4. Perform archive truncation in objective space to the individuals in A_{t+1}^O and archive truncation in decision space in A_{t+1}^V until $|A_{t+1}^O|, |A_{t+1}^V| = \bar{N}$.

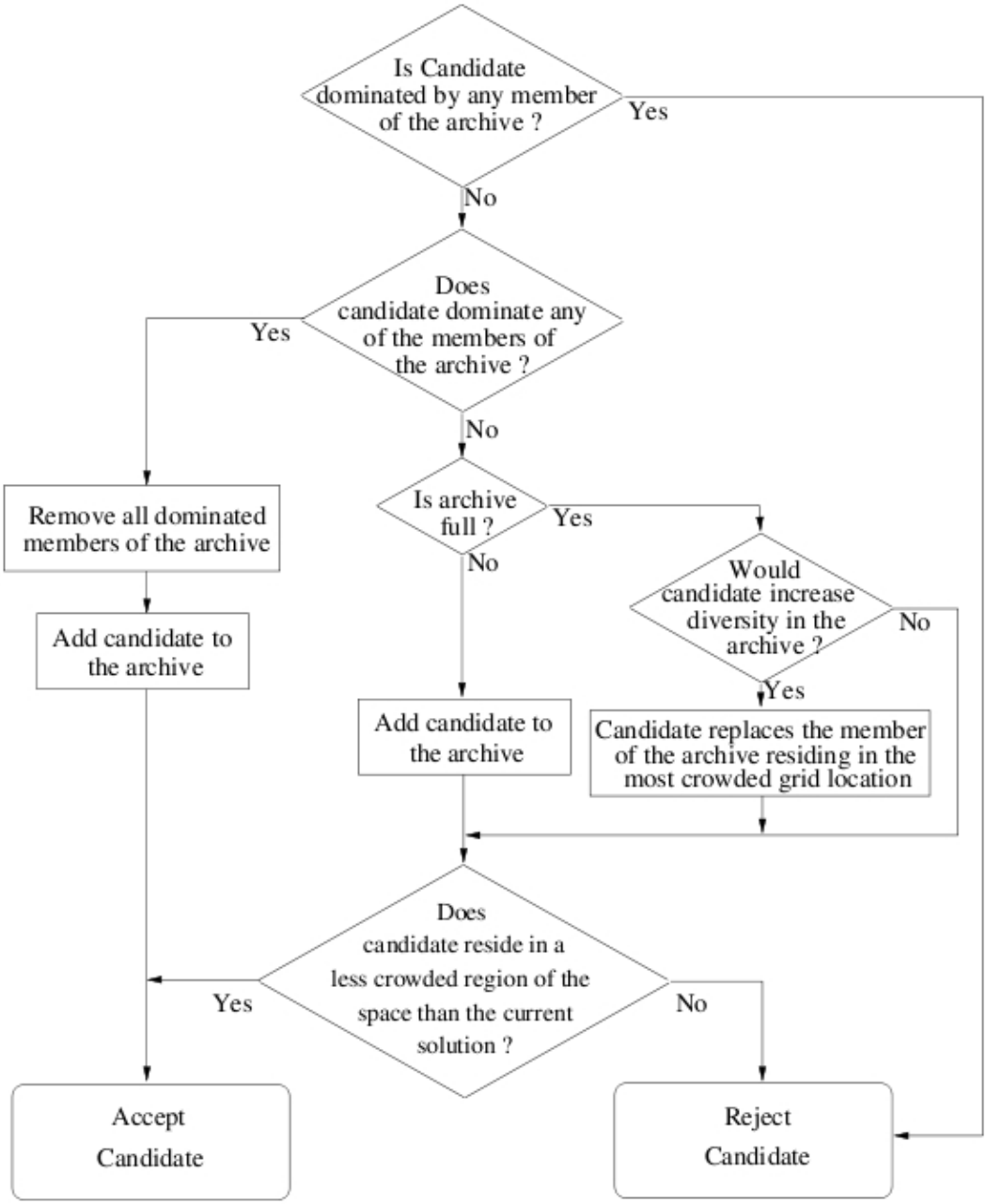


Figure 57: PAES archiving and acceptance logic [63].

Also the PAES uses an external archive to store elitist solutions. The base version of the algorithm begins with the initialization of a single chromosome (the *current solution*) which is then evaluated using the objective functions. A copy is made and a mutation operator is applied to the copy. This mutated copy is evaluated and forms the new *candidate solution*. The current and candidate solutions are then compared according to Pareto dominance. If neither solution dominates the other, the new candidate solution is compared with solutions saved in the archive. Candidates which dominate some individuals in the archive are always accepted and archived. Candidates which are dominated by the archive are always rejected, while those which are nondominated are accepted and/or archived based on the degree of crowding in their grid location. This archiving and acceptance logic is made explicit in Figure 57.

3.4.4 *Constraint handling*

We considered parameter extraction problems with only two types of constraints:

- *box constraints* on the parameters of a model of an electronic device;
- promoting model parameter values which do not cause a circuit simulator to abort.

The first constraint type is crucial when recombination and mutation operators generate chromosomes whose alleles lie outside the ranges defined at step 2 of the parameter extraction procedure (see Section 1.6.2). In this case, we used a constraint handling technique based on repairing infeasible solutions in order to make them feasible.

The second constraint type has been considered because particular combinations of model parameters can cause simulation failures. The reasons why a circuit simulation aborts are explained in Section 1.6.3 and in Appendix A. In this case, constraints are not given in algebraic form, so the exact location of the boundary between the feasible and infeasible regions is unknown. There is no way to quantify the amount by which a constraint is violated. Another important detail is that, for a given chromosome (hence a given set of parameter values), if a simulation of a specific output characteristic fails, simulations of other characteristics may end successfully. For all these reasons we implemented a *penalty method* where the fitting error associated to each failed simulation is set to a high value (the *penalty factor*). In this way, an individual corresponding to an aborted simulation will not be able to dominate any other solution with respect to the penalized objective.

The definition of the penalty factors in a penalty function is not straightforward. Ideally, the penalty should be kept as low as possible, just above the limit below which infeasible solutions are optimal [11]. This is due to the fact that if the penalty is too high or too low, then the problem might become very difficult for an evolutionary algorithm to solve [20]. If the penalty is too high and the optimum lies at the boundary of the feasible region, the EA will be pushed inside the feasible region very quickly, and will not be able to move back towards the boundary with the infeasible region. On the other hand, if the penalty is too low, a lot of the search time will be spent exploring the infeasible region because the penalty will be negligible with respect to the objective function [11].

3.5 STRUCTURE OF SELECTED MOEAS

Although the NFL theorems [103] indicate that there is no "best" MOEA, certain MOEAs have been experimentally shown to be more likely effective (robust) than others for specific MOP benchmarks and certain classes of real-world problems. Among these MOEAs we selected and implemented:

- NSGA-II [24];
- SPEA2 [109];
- SPEA2+ [62];
- PAES [63];
- MO-CMA-ES [53].

In the following sections, we present the pseudocode of these MOEAs.

3.5.1 *Nondominated Sorting Genetic Algorithm-II (NSGA-II)*

The pseudocode of the NSGA-II [24] in shown in Algorithm 3.6. This algorithm is currently used in most MOEA comparisons.

3.5.2 *Strength Pareto Evolutionary Algorithm 2 (SPEA2)*

The procedure of the SPEA2 [109] is given in Algorithm 3.7. The SPEA2 and NSGA-II are two of the most prominent MOEAs used when comparing a newly designed MOEA. They rely heavily on their density estimator mechanisms.

3.5.3 *Improved Strength Pareto Evolutionary Algorithm 2 (SPEA2+)*

SPEA2+ [62] adds three mechanisms to SPEA2 in order to improve its searching ability:

- an archive mechanism to maintain diversity of the solutions in the objective and decision spaces;
- a crossover mechanism named *neighborhood crossover*;
- a different mating selection method.

The first strategy has already been described in Section 3.4.2 and in Section 3.4.3.

Let us consider the crossover operator. In [62], Kim et al. state that effective crossover often cannot be performed in MOEAs, as the searching directions of each individual are very different from one another. Therefore, they propose neighborhood crossover, which performs crossover with individuals neighboring each other in objective space. In neighborhood crossover, individuals that match in the search direction are crossed over to generate offspring that are similar to the parents.

Binary tournament selection is used in SPEA2 as a method for mating selection. Using this selection technique results in an increase in nondominated individuals

Algorithm 3.6 NSGA-II algorithm

Input: Vector of objective functions $\boldsymbol{z}$, population size N, maximum number of generations g

Output: Approximate Pareto optimal solutions $P_{\text{known}}, PF_{\text{known}}$

1: Create a random parent population P_0 of size N
2: Use the fast nondominated sorting algorithm to identify the nondominated fronts $F_1, F_2, \ldots, F_R$ in P_0
3: Assign to each solution a rank equal to its nondomination level
4: Create a child population Q_0 of size N
5: Binary tournament selection on P_0 based on nondomination rank
6: Recombination and mutation
7: **for** $t = 0$ to $g - 1$ **do**
8: Set $R_t = P_t \cup Q_t$
9: Use the fast nondominated sorting algorithm to identify the nondominated fronts $F_1, F_2, \ldots, F_R$ in R_t
10: Set $P_{t+1} \leftarrow \emptyset, i \leftarrow 1$
11: **while** $|P_{t+1}| + |F_i| \leq N$ **do**
12: Calculate crowding distance in F_i
13: Include ith nondominated front in the parent pop, $P_{t+1} \leftarrow P_{t+1} \cup F_i$
14: Check the next front for inclusion, $i \leftarrow i + 1$
15: **end while**
16: Sort F_i in descending order with respect to crowding distances
17: Choose the first $(N - |P_{t+1}|)$ elements of F_i, $P_{t+1} \leftarrow P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$
18: Create a new population Q_{t+1}
19: Binary tournament selection on P_{t+1} based on nondomination rank and the crowding distance as second level sorting criterion
20: Recombination and mutation
21: **end for**

Algorithm 3.7 SPEA2 algorithm

Input: Vector of objective functions $\boldsymbol{z}$, population size N, archive size $\bar{N}$, maximum number of generations g
Output: Approximate Pareto optimal solutions $P_{\text{known}}, PF_{\text{known}}$

1: Create a random parent population P_0 of size N
2: Create empty archive $A_0 \leftarrow \emptyset$
3: **for** $t = 0$ to $g - 1$ **do**
4: For each individual $\boldsymbol{x} \in P_t \cup A_t$ calculate fitness value $f(\boldsymbol{x}, t)$
5: Copy all nondominated individuals in $P_t \cup A_t$ to A_{t+1}
6: **if** $|A_{t+1}| > \bar{N}$ **then**
7: Perform archive truncation on A_{t+1} in objective space until $|A_{t+1}| = \bar{N}$
8: **else if** $|A_{t+1}| < \bar{N}$ **then**
9: Sort $P_t \cup A_t$ in ascending order with respect to fitness values
10: Fill A_{t+1} with dominated individuals in $P_t \cup A_t$
11: **end if**
12: Create a new population P_{t+1}
13: Binary tournament selection on A_{t+1} based on fitness values $f(\cdot, t)$
14: Recombination and mutation
15: **end for**

within the external elitist list, and in most cases all individuals become nondominated in the later stages of the search [62]. Thus, use of binary tournament selection to generate the population sacrifices diversity of nondominated individuals. Therefore, in the proposed method, as the mating selection method, all of the archive A^O_{t+1} is copied to the population used in the search. This copy operation maintain the diversity of the population to allow for a more global search.

3.5.4 *Pareto Archived Evolution Strategy (PAES)*

PAES [63] consists of a $(1+1)$ evolution strategy (i.e., a single parent that generates a single offspring) in combination with a historical archive that records some of the nondominated solutions previously found. As the termination criterion, we considered the maximum number of objective function evaluations.

The archiving and acceptance logic used in lines 9–11 is made explicit in Figure 57. PAES is intended as a good baseline approach, against which more involved methods may be compared, and may also serve well in some real-world applications when local search seems superior to or competitive with population-based methods [63].

Other implementations of PAES have been proposed, namely $(1+\lambda)$-PAES and $(\mu+\lambda)$-PAES [63].

3.5.5 *Covariance Matrix Adaptation for Multi-objective Optimization (MO-CMA-ES)*

In this section, some variants of the CMA-ES for multi-objective optimization are presented noting that they continue to be modified and improved in newer versions.

Algorithm 3.8 SPEA2+ algorithm

Input: Vector of objective functions $\boldsymbol{z}$, population size N, archive size $\bar{N}$, maximum number of generations g
Output: Approximate Pareto optimal solutions $P_{\text{known}}, PF_{\text{known}}$

Create a random parent population P_0 of size N
Create empty archives $A_0^O \leftarrow \emptyset, A_0^V \leftarrow \emptyset$
for $t = 0$ to $g - 1$ **do**
    For each individual $\boldsymbol{x} \in P_t \cup A_t^O \cup A_t^V$ calculate fitness value $f(\boldsymbol{x}, t)$
    Copy all nondominated individuals in $P_t \cup A_t^O \cup A_t^V$ to A_{t+1}^O and A_{t+1}^V
    if $|A_{t+1}^O| > \bar{N}$ **then**
        Perform archive truncation on A_{t+1}^O in objective space until $|A_{t+1}^O| = \bar{N}$
        Perform archive truncation on A_{t+1}^V in variable space until $|A_{t+1}^V| = \bar{N}$
    else if $|A_{t+1}^O| < \bar{N}$ **then**
        Sort $P_t \cup A_t^O \cup A_t^V$ in ascending order with respect to fitness values
        Fill A_{t+1}^O and A_{t+1}^V with dominated individuals in $P_t \cup A_t^O \cup A_t^V$
    end if
    Create a new population P_{t+1}
      Set $P_{t+1} \leftarrow A_{t+1}^O$
      Neighborhood crossover and mutation
end for

Algorithm 3.9 PAES algorithm

Input: Vector of objective functions $\boldsymbol{z}$, archive size $\bar{N}$, number of subdivisions of the objective space in the grid used for encouraging diversity *ndiv*
Output: Approximate Pareto optimal solutions $P_{\text{known}}, PF_{\text{known}}$

Generate random current solution C
Evaluate and add to archive A
repeat
    Mutate C to generate new candidate solution C'
    Evaluate C'
    if $C \preceq C'$ **then**
        Discard C'
    else
        Compare candidate solution C' with archive members
        Update archive A
        Select new current solution from candidate and current solution
    end if
until termination criterion is met

In the CMA-ES Section 3.2 a small population size is usually sufficient and only one set of strategy parameters is maintained. In multi-objective optimization problems a large population is needed to evolve a diverse set of solutions, each ideally representing a Pareto optimal solution. The optimal strategy parameters for the members of this population may differ considerably and should therefore be adapted individually. This suggests that it is reasonable to apply a multi-objective selection mechanism to a population of individuals each of which uses the strategy adaptation of the CMA-ES (for details about the covariance matrix adaptation see Section 3.2 and [48]). For this reason, Igel et al. [53] first develop a single objective, elitist CMA-ES with $(1+\lambda)$-selection. In this elitist $(1+\lambda)$-CMA-ES the parent population consists of a single individual generating λ offspring and the best individual out of parent and offspring becomes the parent of the next population. A multi-objective version of CMA-ES, named MO-CMA-ES, is then implemented by considering a population of $(1+\lambda)$ evolution strategies, which are subject to multi-objective selection using nondominated sorting and the contributing hypervolume.

The $(1+\lambda)$-CMA-ES is presented in Section 3.2.5.

The multi-objective selection criterion can be defined as follows. Let A be a set of individuals and a, a' two individuals in A. Additionally, the following notation is used:

- $r(a', A)$ is the level of nondominance of a', as it is computed by the NSGA-II (see Figure 53);
- $A_{E,r(a',A)}$ is the nondominated front (better, the Pareto set) whose rank is given by $r(a', A)$;
- $s(a', A_{E,r(a',A)})$ is the rank of a' computed considering its contribution to the hypervolume of $A_{E,r(a',A)}$.

We can now introduce the relation $\prec_{s,A}$

$$a \prec_{s,A} a' \iff r(a,A) < r(a',A) \quad \text{or} \quad [(r(a,A) = r(a',A)) \wedge (s(a, A_{E,r(a',A)}) > s(a', A_{E,r(a',A)}))]. \tag{3.3}$$

That is, a is better than a' when compared using $\prec_{s,A}$ if either a has a better level of nondominance or a and a' are on the same level but a contributes more to the hypervolume when considering the points at that level of nondominance.

Now we have all the ingredients for a multi-objective CMA-ES. The $\mu \times (1+\lambda)$-MO-CMA-ES is presented in Algorithm 3.10. In this algorithm a population of μ elitist $(1+\lambda)$-CMA-ES is maintained. The kth individual in generation t is denoted by $a_k^{(t)} = [\boldsymbol{x}_k^{(t)}, \bar{p}_{\text{succ},k}^{(t)}, \sigma_k^{(t)}, \boldsymbol{p}_{c,k}^{(t)}, \mathbf{C}_k^{(t)}]$.

Some comments:

- $\lambda_{\text{succ},R^{(t)},i}^{(t+1)} = \left| \left\{ l = 1, \ldots, \lambda \;\middle|\; a_{i,l}^{\prime(t+1)} \prec_{s,R^{(t)}} a_i^{(t)} \right\} \right|$ is the number of successful offspring from parent $a_i^{(t)}$. This is a *parent-based notion of success*.
- $R_{\prec:i}^{(t)}$ is the best individual in $R^{(t)}$ with respect to $\prec_{s,R^{(t)}}$.
- In every generation t each of the μ parents generates λ offspring, lines 5–10.

Algorithm 3.10 $\mu \times (1+\lambda)$-MO-CMA-ES

$t \leftarrow 0$
Initialize $a_k^{(t)}$ for $k = 1, \ldots, \mu$
Set $P_0 = \left\{ a_k^{(t)} \;\middle|\; 1 \leq k \leq \mu \right\}$
repeat
  for $k = 1, \ldots, \mu$ **do**
    for $h = 1, \ldots, \lambda$ **do**
      $a_{k,h}^{\prime(t+1)} \leftarrow a_k^{(t)}$
      $\boldsymbol{x}_{k,h}^{\prime(t+1)} \sim N(\boldsymbol{x}_k^{(t)}, {\sigma_k^{(t)}}^2 \mathbf{C}_k^{(t)})$
    end for
  end for
  $R^{(t)} = \left\{ a_{k,h}^{\prime t+1}, a_k^{(t)} \;\middle|\; 1 \leq k \leq \mu \wedge 1 \leq h \leq \lambda \right\}$
  $P_{(t+1)} \leftarrow \emptyset$
  for $i = 1, \ldots, \mu$ **do**
    $a_i^{(t+1)} \leftarrow R^{(t)}_{\prec:i}$
    $P_{(t+1)} \leftarrow P_{(t+1)} \cup a_i^{(t+1)}$
  end for
  for $i = 1, \ldots, \mu$ **do**
    `updateStepSize`$\left(a_i^{(t+1)}, \lambda^{(t+1)}_{\text{succ},R^{(t)},i}\right)$
    if $a_i^{(t+1)}$ was an offspring **then**
      `updateCovariance`$\left(a_i^{(t+1)}, \frac{\boldsymbol{x}_i^{(t+1)} - \boldsymbol{x}_i^{(t)}}{\sigma_i^{(t)}}\right)$
    end if
  end for
  $t \leftarrow t + 1$
until stopping criterion is met

- The step sizes of a parent and its offspring are updated (line 18) depending on whether the mutations were successful, that is, whether the offspring are better than the parent according to the relation $\prec_{s,R^{(t)}}$ (see eq. (3.3)). We stress again that this is a parent-based notion of success. Another notion of success will be introduced shortly.
- The covariance matrix of the offspring is updated taking into account the mutation that has led to its genotype (line 20).
- Both the step size and the covariance matrix update are the same as in the single-objective $(1+\lambda)$-CMA-ES.
- The best μ individuals in $R^{(t)}$ sorted by $\prec_{s,R^{(t)}}$ form the next parent generation, lines 12–16.

Empirical evaluations of the MO-CMA-ES are reported in [53].

We implemented the $\mu \times (1+\lambda)$-MO-CMA-ES and two other variants which are supposed to improve its performance.

In the first variant a different strategy is proposed for the covariance matrix adaptation [93]. In the $\mu \times (1+\lambda)$-MO-CMA-ES described in [53], each individual learns its own covariance matrix for the mutation distribution considering only its parent and offspring. However, the optimal mutation distribution of individuals that are close in decision space are likely to be similar if we presume some notion of continuity of the optimization problem. Therefore, Voß et al. [93] proposed an inter-individual transfer of information in the MO-CMA-ES considering also successful mutations of *neighboring individuals* for the covariance matrix adaptation. A full description of this new updating strategy and its empirical evaluation may be found in [93]. We applied it to parameter extraction problems. Results are shown in Chapter 4.

The second variant is based on the improved step size adaptation scheme described in [94]. In Algorithm 3.10, a mutation is regarded as successful if the offspring ranks better that its parent in the elitist, rank-based selection procedure. In contrast, in [94] a mutation is regarded as successful if the offspring is selected into the next parental population. This criterion reduces the computational complexity of the MO-CMA-ES. Moreover, it leads to larger step sizes and thereby counteracts premature convergence. We refer to [94] for a description of this updating method and its empirical results. We applied it to parameter extraction problems with and without recombination of strategy parameter presented in [93]. Results are shown in Chapter 4.

3.6 MANY-OBJECTIVE OPTIMIZATION BASICS

MOEAs usually work very well on two-objective problems. Their search ability is, however, severely deteriorated by the increase in the number of objectives. Multi-objective problems with four or more objectives are often referred to as *many-objective problems*.

This situation easily happens in parameter extraction problems, since there can be several several output characteristics that have to be fitted with a device compact model. For example, as it is shown in Chapter 1, the behavior of a power diode

can be described through at least four output characteristics: junction capacitance vs reverse bias voltage, DC current–voltage characteristic, reverse recovery current and voltage under a single turn-off condition. In Chapter 1, we showed that fitting of the capacitance characteristic can be carried out independently from the other characteristics. In Chapter 4, we will show that the problem can be further simplified to the simultaneous optimization of the DC i–v and reverse recovery current curves only.

However, there can be more challenging situations. For example, reverse recovery current and voltage measurements could be collected for different values of the supply voltage and/or the load current. It could also be necessary to fit turn-on characteristics. The situation is even worse if a more sophisticated devices than a diode is analyzed, like an IGBT, which is characterized by a higher number of output characteristics, or several devices and their corresponding models are studied all together. The higher the number of characteristics to be fitted, the more difficult a multi-objective parameter extraction problem. We are currently working on an extraction problem for Extended Lauritzen diode model with seven objectives: DC i–v, reverse recovery current and voltage under three different turn-off conditions.

It is also important to note that, when optimizing several models at the same time, a multi-objective optimization has to be carefully monitored in order not to create models which are more behavioral and less physical [7].

For these reasons, we studied how to deal with many-objective problems.

When applying a MOEA to a many objective problem, we may encounter a number of serious difficulties such as:

1. Deterioration of the search ability of Pareto dominance-based algorithms, such as NSGA-II [24] and SPEA2 [109]. When the number of objectives increases, almost all solutions in each population become nondominated. This severely weakens the Pareto dominance-based selection pressure toward the Pareto front. That is the convergence property of MOEAs is deteriorated.

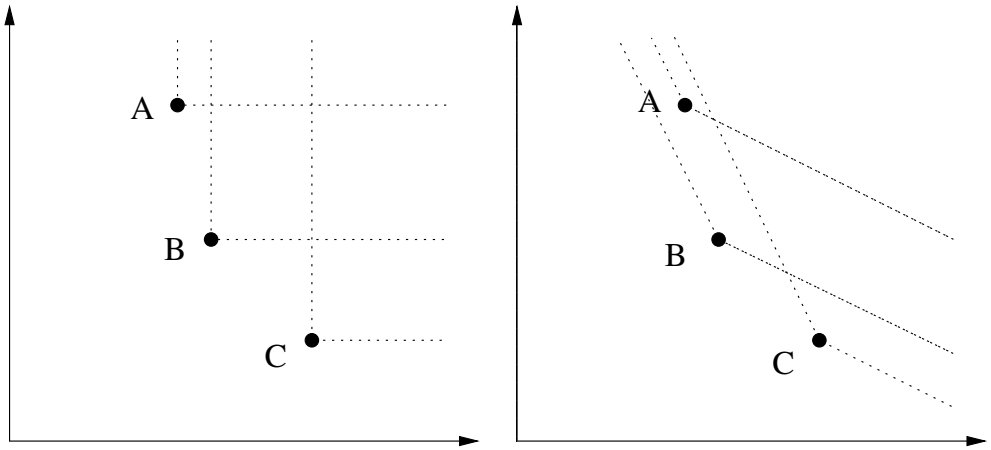
2. Exponential increase in the number of solutions required for approximating the entire Pareto front. The goal of a population-based MOEA is to find a set of nondominated solutions that well approximates the entire Pareto front. Since the Pareto front is a hyper-surface in objective space, the number of solutions required for its approximation exponentially increases with the dimensionality of the objective space (i.e., with the number of objectives). That is, we may need thousands of nondominated solutions to approximate the entire Pareto front of a many objective problem.

3. Difficulty of the visualization of solutions. This can make the choice of a final solution very hard in many-objective optimization.

The first difficulty has been pointed out in some studies [61], [83], [58].

A scalability improvement of MOEAs to many-objective problems can be achieved by increasing the selection pressure toward the Pareto front.

One approach based on this idea is to modify Pareto dominance in order to decrease the number of nondominated solutions in each population [85]. In order to increase the selection pressure toward the Pareto front by the Pareto sorting in NSGA-II, Sato et al. modified Pareto dominance as shown in Figure 58 where the



(a) Original Pareto dominance. (b) Modified Pareto dominance.

Figure 58: Illustration of the modification of Pareto dominance.

dominated region by each solution is delimited by a dotted line. When we use the standard Pareto dominance, in Figure 58a, all three solutions are nondominated with each other. On the other hand, solution A is dominated by solution B if we use the modified dominance in Figure 58b. In this manner the selection pressure toward the Pareto front can be strengthened because the number of nondominated solutions in each population is decreased by the use of the modified Pareto dominance. The extend of the modification of Pareto dominance (i.e., the angle of the dominated region in Figure 58b) should be adjusted to the number of objectives. Roughly speaking, the increase in the number of objectives requires a wider angle of the dominated region in Figure 58b as suggested in [85]. The modification of Pareto dominance leads to the decrease in the diversity of solutions [85].

An alternative approach to the modification of Pareto dominance is to assign different ranks to nondominated solutions [88, 65, 18]. As in the case of the modification of Pareto dominance, the introduction of different ranks to nondominated solutions leads to the increase in the selection pressure toward the Pareto front and the decrease in the diversity of solutions.

Another idea for the scalability improvement of MOEAs to many-objective problems is to use different scalarizing functions for fitness evaluations [57, 56].

These three classes of methods (modification of the Pareto dominance, assignment of different ranks to nondominated solutions, use of scalarizing functions) are not well suited to solve parameter extraction problems in our opinion. Their weaknesses are similar to those that have been discussed in Section 1.6:

- Defining the extend of the modified Pareto dominance or selecting proper weights to characterize the decision-maker's preferences can be quite difficult and very much problem dependent.
- Scalarizing methods are sensitive to the shape of the Pareto front: a weighted-sum approach cannot return any solution on the concave portion of the Pareto front.

For these reasons, we followed different strategies to deal with three or more objectives.

The second difficulty of many-objective optimization (i.e., the exponential increase in the number of nondominated solutions that are necessary for the approximation of the Pareto front) has often been tackled by incorporating preference information in MOEAs [35], [25]. Preference information is used to concentrate on a small region of the Pareto front while MOEAs are used to find multiple nondominated solutions in such a small region of the Pareto front.

We implemented the procedure described in [25]. First, the decision-maker is asked to specify reference points (in objective space). The procedure will then look for points on the Pareto frontier which are close to the reference points provided. It is therefore able to handle multiple preference conditions simultaneously and for each preference condition, a set of Pareto-optimal solutions is the target set of solutions, instead of one solution. Another capability of this method is that it is indifferent to the shape of the Pareto frontier (such as convex or non-convex, continuous or discrete, connected or disconnected and others). In [25], this approach has been embedded in the NSGA-II. The resulting MOEA was called Reference point based Non Dominated Sorting Genetic Algorithm II (R-NSGA-II). However, a similar strategy can also be adopted with other MOEAs. The main ideas behind choosing the preferred set of solutions are:

1. Solutions closer to the reference points (in the objective space) are to be emphasized more.

2. Solutions within a ε-neighborhood to a near-reference-point solution are deemphasized in order to maintain a diverse set of solutions near each reference point.

As usual, both parent and offspring populations are combined together and a nondominated sorting is performed. In the following, we describe an iteration of the R-NSGA-II, which incorporates the above two ideas:

STEP 1. An ε-clearing idea is applied to nondominated fronts. For a given nondominated front, a random solution is picked from the nondominated set first. Thereafter, all solutions having a sum of normalized differences in objective values of ε or less from the chosen solution are discouraged to remain in the race. This way, only one solution within a ε-neighborhood is emphasized. Then, another solution from the nondominated set (and is not already considered earlier) is picked and the above procedure is performed, until all the individuals in the front have been considered. The algorithm is then repeated for the next nondominated front, until the total number of individuals that survive the *varepsilon*-procedure is enough to fill the next parental population.

STEP 2. For each reference point and for each "survived" nondominated front, the normalized Euclidean distance (see equation (3.4)) of each solution of the front is calculated and the solutions are sorted in ascending order of distance. This way, the solution closest to the reference point is assigned a rank of one.

STEP 3. After such computations are performed for all reference points, the minimum of the assigned ranks is assigned as the *preference distance* to a solution. This way, solutions closest to all reference points are assigned the smallest

preference distance of one. The solutions having next-to-smallest Euclidean distance to all reference points are assigned the next-to-smallest preference distance of two, and so on.

STEP 4. Solutions with a smaller preference distance are preferred in the tournament selection and in forming the new parent population. Solutions from the cleared nondominated fronts are chosen front-wise up and the preference distance is used to choose a subset of solutions from the last front which cannot be entirely chosen to maintain the population size of the next population.

The above procedure emphasizes each objective function equally. If the decision-maker is interested in biasing some objectives more than others, a suitable weight vector can be used with each reference point and instead of emphasizing solutions with the shortest weighted Euclidean distance from a reference point, solutions with a shortest weighted Euclidean distance from the reference point can be emphasized

$$d_{ij} = \sqrt{\sum_{i=1}^{p} w_i \left(\frac{z_i(\boldsymbol{x}) - \bar{r}_i}{z_i^{\max} - z_i^{\min}} \right)^2} \tag{3.4}$$

where $z_i^{\max}$ and $z_i^{\min}$ are the population maximum and minimum function values of the ith objective and $\bar{r}_i$ is the ith component of the reference point. Note that this weighted distance can also be used to find a set of preferred solutions in the case of problems having non-convex Pareto front.

Simulation results of the R-NSGA-II are shown in [25] and in Chapter 4. In [25] the effects of different choices for the parameter ε are also shown.

A direct approach to the handling of the third difficulty associated to a many-objective problem (the difficulty of the visualization of solutions) is to decrease the number of objectives. Dimensionality reduction (i.e. objective selection) can remedy not only the third difficulty but also the other difficulties. We implemented the objective reduction method proposed in [23], which is based on Principal Component Analysis (PCA). It should be used to solve those large objective (p) problems which degenerate to possess a lower-dimensional Pareto optimal front (lower than p).

In some problems, even though, apparently there may exist a conflicting scenario between objectives, for two randomly picked feasible solutions, there may not exist any conflict between the same objectives, for two solutions picked from near the Pareto optimal front. That is, although a conflict exists elsewhere, some objectives may behave in a non-conflicting manner, near the Pareto optimal region. In such cases, the Pareto front will be of dimension lower than the number of objectives. Then some of the objectives are *redundant*. This may happen in parameter extraction problems, where there are strong couplings between different characteristics. For example, in Chapter 4 optimization of reverse recovery voltage is dropped from the simultaneous optimization of DC i–v, reverse recovery current and reverse recovery voltage characteristics, since current and voltage look quite interrelated in the proximity of the Pareto front. This provides computational evidence of the physical coupling between current and voltage in the phenomena of turn-off.

A principal component analysis is concerned with explaining the covariance structure of a set of variables through a few *linear* combinations of these variables. Although p components are required to reproduce the total system variability, often much of this variability ca be accounted for by a small number k of the principal components. If so, there is (almost) as much information in the k components as there is in the original p variables. The k principal components can then replace the initial p variables, and the original data set, consisting of N measurements on p variables, is reduced to a data set consisting of N measurements on k principal components.

Algebraically, principal components are particular linear combinations of the p original variables $z_1, z_2, \ldots, z_p$ (the objective functions, in our case). Geometrically, these linear combinations represent the selection of a new coordinate system obtained by rotating the original system with $z_1, z_2, \ldots, z_p$ as the coordinate axes. The new axes represent the directions with maximum variability.

We refer to [59] for a comprehensive introduction to PCA. In the following, we will present how this technique can be applied to multi-objective optimization.

PCA FOR MULTIOBJECTIVE OPTIMIZATION. In the context of a p-objective optimization problem being solved using N population members, the initial data matrix will be of size $p \times N$. A PCA is then carried out on the correlation matrix $\mathbf{R}$, since variables can be measured on different scales, or on a common scale with widely differing ranges. Principal components are nothing but the eigenvectors of the correlation matrix. The eigenvector corresponding to the largest eigenvalue is referred as the first principal component, the one corresponding to the second largest eigenvalue is called the second principal component and so on. The elements of a principal component denote the relative contribution of each objective. A positive value denotes an increase in objective value moving along this principal component (axes) and a negative value denotes a decrease.

The algorithm proposed in [23] starts with analyzing the first principal component and then proceeds to analyzing the second principal component and so on, till all the significant components are considered. The decision-maker has to provide a threshold cut, and when the cumulative contribution of all previously principal components exceeds the threshold cut, no more principal components are analyzed. The choice of the threshold cut is very important. If it is too high, many redundant objectives may be chosen by the PCA; if it is too small, important objectives may be ignored. To emphasize the contributions of the eigenvectors, the matrix $\mathbf{R}\mathbf{R}^{\mathrm{T}}$ is used, instead of $\mathbf{R}$, to find eigenvalues and eigenvectors. This does not change the eigenvectors of the original correlation matrix $\mathbf{R}$, but the eigenvalues get squared and more emphasized.

The complete PCA procedure is given in Algorithm 3.11. The final reduction (line 33) is implemented as follows. We return to a *reduced* correlation matrix (only columns and rows corresponding to non-redundant objectives) and investigate if there still exists a set of objectives having identical positive or negative correlation coefficients with other objectives and having a positive correlation among themselves. This will suggest that any one member from

Algorithm 3.11 Principal Component Analysis for Multi-Objective Optimization

Input: Population of individuals P, threshold cut TC
Output: List L of non-redundant objectives

1: Initialize list L, $L \leftarrow \emptyset$
2: Compute the correlation matrix $\mathbf{R}$
3: Compute eigenvalues $\hat{\lambda}_i$ and eigenvectors $\hat{\boldsymbol{e}}_i$ of $\mathbf{R}\mathbf{R}^{\mathrm{T}}$
4: Add to L both objectives contributing the most positive and most negative to the first principal component
5: $i \leftarrow 2$
6: **while** Cumulative proportion of variance up to the $(i-1)$th component is $\leq TC$ **do**
7: **if** $\hat{\lambda}_i \leq 0.1$ **then**
8: $L \leftarrow L \cup \left\{ j \;\middle|\; j = \operatorname{argmax}_{l=1,\ldots,p} |\hat{e}_i^l| \right\}$
9: **else if** $\hat{e}_i^l > 0$ for $l = 1, \ldots, p$ **then**
10: $L \leftarrow L \cup \left\{ j \;\middle|\; j = \operatorname{argmax}_{l=1,\ldots,p} \hat{e}_i^l \right\}$
11: **else if** $\hat{e}_i^l < 0$ for $l = 1, \ldots, p$ **then**
12: $L \leftarrow L \cup \{ 1, \ldots, p \}$
13: **break**
14: **else**
15: Find the highest positive element m_{pos} of $\hat{\boldsymbol{e}}_i$
16: Find the most negative element m_{neg} of $\hat{\boldsymbol{e}}_i$
17: **if** $m_{\mathrm{pos}} < |m_{\mathrm{neg}}|$ **then**
18: **if** $m_{\mathrm{pos}} \geq 0.9|m_{\mathrm{neg}}|$ **then**
19: Add to L both the objectives corresponding to m_{pos} and m_{neg}
20: **else**
21: Add to L only the objective corresponding to m_{neg}
22: **end if**
23: **else**
24: **if** $m_{\mathrm{pos}} \geq 0.8|m_{\mathrm{neg}}|$ **then**
25: Add to L both the objectives corresponding to m_{pos} and m_{neg}
26: **else**
27: Add to L only the objective corresponding to m_{pos}
28: **end if**
29: **end if**
30: **end if**
31: $i \leftarrow i + 1$
32: **end while**
33: Perform final reduction using the reduced correlation matrix

such group would be enough to establish the conflicting relationships with the remaining objectives. In such case, the one which was chosen the earliest (corresponding to the largest eigenvalue) by the PCA. Other objectives from the set are not considered further.

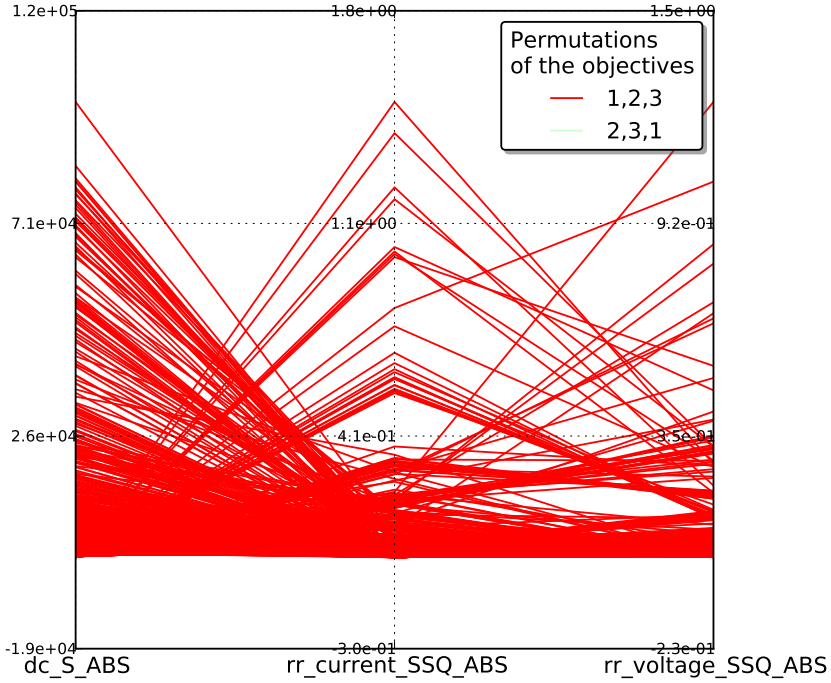
A systematic and rigorous way of representing the relationships between multiple variables – in our case, objective functions – is given by *parallel coordinates* [96].

The approach of parallel coordinates places all the axes parallel to each other thus allowing any number of axes to be shown in a flat representation. Figure 59 illustrates a representation that deals with three objectives associated to a parameter extraction problem. Each line in the graph connects the performance objectives achieved by an individual member of an approximated Pareto front and represents a potential solution to the extraction problem.

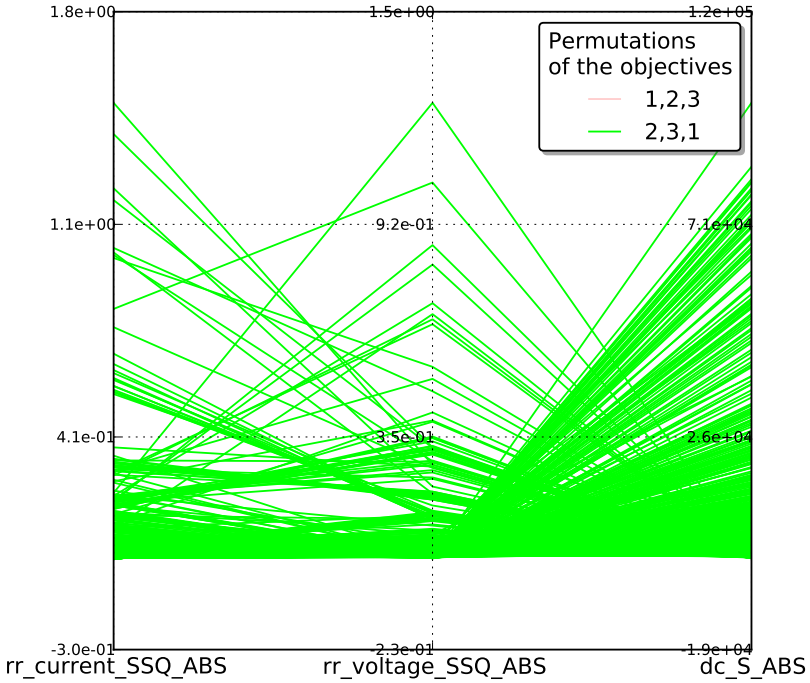
It can be shown that the geometrical features of a surface in p-dimensional space are preserved in the parallel coordinate system [96]. This is important because it allows these features to be easily identifiable when represented in parallel coordinates and therefore the relationship between the variables that give rise to these features can be visualized. For example, in Figure 59, "crossing lines" indicate conflict between the two adjacent objectives. The degree of conflict is demonstrated by the intensity, or degree to which, the lines cross. Conversely, lines that do not cross demonstrate objectives which are in relative harmony with one another. Objectives 1 and 2 are clearly conflicting, while objectives 2 and 3 seem to be less in conflict. As we have already said, it is not physically surprising that reverse recovery current and voltage are not strongly conflicting because of the coupling in the phenomena of turn-off. Therefore, parallel coordinate plots may help in reducing the dimensionality of the problems, like the PCA. We suggest to use both the approaches when performing dimensionality reduction. In [35] parallel coordinate plots have been applied to a real-world problem in order to focus on a specific region of interest on the Pareto front and to isolate a desired design solution.

Other properties of parallel coordinates are:

- Clustering is easily diagnosed [96].
- There is no loss of data in the representation, which in turn ensures that there is a unique representation for each unique set of data [96].
- Linear relationships can be diagnosed. A linear rescaling of one or more of the axes is sometimes helpful because it guides the eye in looking for approximately parallel line segments. Nonlinear relationships can also be highlighted by applying suitable nonlinear transformations to the axes.
- It requires multiple views (different orderings of objectives) to see different trade-offs. In [96], an algorithm is proposed to find a minimal set of permutations of the objective axes so that every possible adjacency is present and trade-offs can be seen more easily.
- It can be hard to see what is going on when many vectors are represented. In [96] some countermeasures to this problem are described.



(a) First permutation of objectives.



(b) Second permutation of objectives.

Figure 59: Parallel coordinate plots of three objectives: DC residual sum of magnitudes (dc_S_ABS), RR current residual sum of squares (rr_current_SSQ_ABS), RR voltage residual sum of squares (rr_voltage_SSQ_ABS). Two permutations of the objectives are needed to show every possible adjacency.

relation		interpretation in objective space
strictly dominates	$A < B$	every $z^2 \in B$ is strictly dominated by at least one $z^1 \in A$
dominates	$A \leq B$	every $z^2 \in B$ is dominated by at least one $z^1 \in A$
better	$A \lhd B$	every $z^2 \in B$ is weakly dominated by at least one $z^1 \in A$ and $A \nsim B$
weakly dominates	$A \leqq B$	every $z^2 \in B$ is weakly dominated by at least one $z^1 \in A$
incomparable	$A \parallel B$	neither $A \leqq B$ nor $B \leqq A$
indifferent	$A \sim B$	$A \leqq B$ and $B \leqq A$

Table 9: Selected preference relations on Pareto front approximations; the corresponding relations on Pareto set approximations are defined by considering the associated Pareto front approximations. Minimization of all objectives is assumed. The relations $>, \geq, \geqq$ and $\rhd$ are defined accordingly with reversed order of the arguments, e.g., $A > B$ is equivalent to $B < A$. Notice that $A < B \implies A \leq B \implies A \lhd B$ and two indifferent Pareto front approximations are identical, while this does not need to hold for two indifferent Pareto set approximations.

3.7 MOEA PERFORMANCE ASSESSMENT

This section introduces statistical methodologies for comparing the performance of two or more MOEAs. Much of the material covered in this section can be found in [64]. In the following, quality assessment in the objective space will be addressed. The term *approximation set* will be used as an alias for an approximated Pareto front returned by an optimizer, PF_{known}. The set of all approximation sets over the objective space is represented by Ω. The preference relations on $\mathbb{R}^p$ listed in Table 5 can be extended to Ω as defined in Table 9. Without loss of generality, minimization is assumed for all objectives.

3.7.1 *Outperformance*

Suppose we would like to assess the quality of the outcome of two multi-objective optimizers. Assume that the two algorithms to be compared are deterministic, i.e., with each optimizer exactly one approximation set is associated. The most natural way to compare two approximation sets A and B generated by two different multi-objective optimizers is to use weak Pareto dominance. There can be four situations (see Table 9):

1. A is better than B;
2. B is better than A;
3. A and B are incomparable;
4. A and B are indifferent.

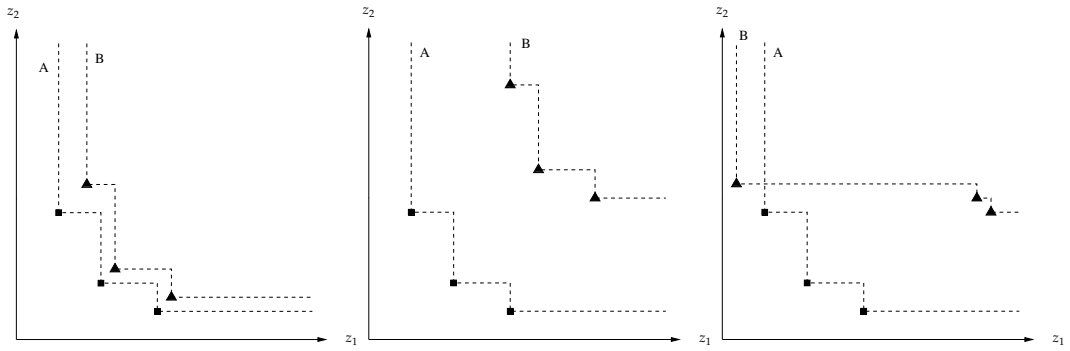


Figure 60: Three examples illustrating the limitations of statements purely based on weak Pareto dominance. In both figures on the left, the approximation set A dominates the approximation set B, but in one case the two sets are much closer together than in the other case. On the right, A and B are incomparable, but in most situations A will be more useful to the decision-maker than B.

Often, though, we are interested in more precise statements that quantify the difference in quality on a continuous scale. For instance, in cases 1 and 2 we may be interested in knowing how much better the preferable approximation set is, and in case 3 one may ask whether either set is better than the other in certain aspects not captured by the preference structure. This is illustrated in Figure 60.

For this reason, quantitative performance measures have been introduced. Since the term *performance* refers to both quality and time, while the measures we will present only capture the former aspect, we will use the term *quality indicator* instead.

Definition 3.1. A (unary) quality indicator is a function $I\colon \Omega \to \mathbb{R}$ that assigns each approximation set a real number.

In combination with the $\leq$ or $\geq$ relation on $\mathbb{R}$, a quality indicator I defines a total order of Ω and thereby induces a corresponding preference structure: A is preferable to B if and only if $I(A) > I(B)$, assuming that the indicator values are to be maximized. That means we can compare the outcomes of two multi-objective optimizers by comparing the corresponding indicator values. An example of indicator value is given by the hypervolume indicator, introduced in Section 3.4.2. It will be described in greater detail later.

Every quality indicator represents certain assumptions about the preferences of the decision-maker. As a consequence, every comparison of multi-objective optimizers is not only restricted to the selected benchmark problems and parameter settings, but also to the quality indicator(s) under consideration.

The total order of Ω imposed by the choice of I should not contradict the partial order of Ω that is imposed by the weak Pareto dominance relation. That is, whenever an approximation set A is preferable to B with respect to weak Pareto dominance, the indicator value for A should be at least as good as the indicator value for B; such indicators are called *Pareto compliant*. In this work, only Pareto compliant indicators have been used.

Many of the indicators that have been proposed in the literature are not Pareto compliant. These quality indicators are called *Pareto non-compliant*. Pareto non-compliant indicators are useful, for instance, to refine the preference structure of a Pareto compliant indicator for approximation sets having identical indicator values.

The above discussion was restricted to unary quality indicators only, although an indicator can take an arbitrary number of approximation sets as arguments.

3.7.2 *Stochasticity*

In Section 3.7.1 we assumed that each algorithm under consideration always generates the same approximation sets for a specific problem. However, MOEAs are stochastic. If a stochastic multi-objective optimizer is applied several times to the same problem, each time a different approximation set may be returned. By running a specific algorithm several times on the same problem instance, one obtains a sample of approximation sets. Now, comparing two stochastic optimizers means comparing the two corresponding approximation set samples. This leads to the issue of statistical hypothesis testing.

There exist two basic approaches in the literature to analyze two or several samples of Pareto set approximation statistically. The more popular approach first transforms the approximation set samples into samples of real values using quality indicators; then, the resulting samples of indicator values are compared based on standard statistical testing procedures.

The alternative approach, the *attainment function method*, summarizes a sample of approximation sets in terms of a so-called *empirical attainment function*. To explain the underlying idea, suppose that a certain stochastic multi-objective optimizer is run once on a specific problem. For each objective vector z in the objective space, there is a certain probability p (do not confuse it with the number of objectives) that the resulting approximation set contains an objective vector that weakly dominates z. We say p is the probability that z is attained in one optimization run of the considered algorithm. The true attainment function is usually unknown, but it can be estimated on the basis of the approximation set samples: one counts the number of approximation sets by which each objective vector is attained and normalizes the resulting number with the overall sample size.

Example 3.2. Consider Figure 61. For the scenario on the right, the three approximation sets cut the objective space into four regions: the upper right region is attained in all of the runs and therefore is assigned a relative frequency of 1, the lower left region is attained in none of the runs, and the remaining two regions are assigned relative frequencies of $1/3$ and $2/3$ because they are attained in one respectively two of the three runs. In the scenario on the left, the objective space is partitioned into six regions; the relative frequencies are determined analogously as shown in the figure.

A third approach proposed in [64] consists in ranking the obtained approximations by using the dominance relation, in the same way that dominance-based fitness assignment ranks objective vectors in MOEAs. First, all approximation sets generated by the different optimizers under consideration are pooled, and then

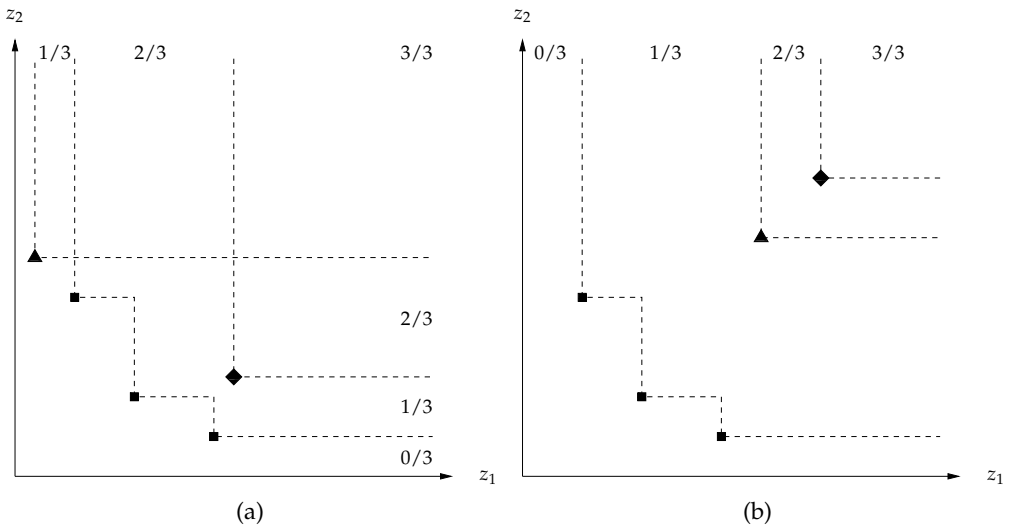


Figure 61: Hypothetical outcomes of three runs for two different stochastic optimizers (left and right). The numbers in the figures give the relative frequencies according to which the distinct regions in the objective space were attained.

each approximation set is assigned a rank reflecting the number of approximation sets in the pool that are better (see nomenclature introduced at the beginning of Section 3.7.1). At the end, for each algorithm, a set of ranks is obtained, We can then verify whether the rank distributions for the algorithms differ significantly or not.

In this work, we followed the quality indicator and empirical attainment function approaches to evaluate the quality of the outcome of MOEAs when applied to parameter extraction problems. These two approaches will be described in Section 3.7.3 in greater detail.

3.7.3 *Sample Transformations*

The three comparison methodologies outlined in the previous section have in common that the sample of approximation sets associated with an algorithm is first transformed into another representation–a sample of indicator values, an empirical attainment function, or a sample of ranks–before the statistical testing methods are applied. In the following, quality indicators and empirical attainment function methods are reviewed.

3.7.3.1 *Quality Indicators*

In this section, two unary quality indicators (see definition 3.1) are presented:

THE HYPERVOLUME INDICATOR I_H. It measures the hypervolume of that portion of the objective space that is weakly dominated by an approximation set A, and has to be maximized (see Figure 62). It has also been considered in Section 3.4.2. In order to measure this quantity, the objective space must be bounded–if it is not, then a bounding reference point that is (at least weakly) dominated by all points should be used, as shown in the figure.

One can also consider the hypervolume difference to a reference set R, and we will refer to this indicator as I_H^-. Given an approximation set A, the indicator value is defined as

$$I_H^-(A) = I_H(R) - I_H(A) \tag{3.5}$$

where smaller values correspond to higher quality–in contrast to the original hypervolume I_H.

The I_H indicator has a desirable property: given two approximation sets A and B such that $A \triangleleft B$, then $I_H(A) > I_H(B)$–provided that the bounding point is *strictly* dominated by all the points in A and B. Therefore, the hypervolume indicator is always able to detect if a set is not better than another. From $I_H(A) < I_H(B)$ one can infer that A cannot be better than B. As we already pointed out, computation of the hypervolume indicator is exponential in the number of objectives [97] and is polynomial in the number of points in the approximation set.

THE UNARY EPSILON INDICATORS I_ε^1 AND $I_{\varepsilon+}^1$. The epsilon indicator family has been introduced in [110] and comprises a multiplicative and an additive version–both exist in unary and binary form. The binary multiplicative epsilon indicator, $I_\varepsilon(A, B)$, gives the minimum factor ε by which each point in B can be multiplied such that the resulting transformed approximation set is weakly dominated by A

$$I_\varepsilon(A, B) = \inf_{\varepsilon \in \mathbb{R}} \left\{ \forall \boldsymbol{z}^2 \in B \quad \exists \boldsymbol{z}^1 \in A \;\middle|\; \boldsymbol{z}^1 \preceq_\varepsilon \boldsymbol{z}^2 \right\}.$$

This indicator relies on the ε-dominance relation, $\preceq_\varepsilon$, defined as

$$\boldsymbol{z}^1 \preceq_\varepsilon \boldsymbol{z}^2 \iff \forall i \in 1, \ldots, p : z_i^1 \leq \varepsilon \cdot z_i^2$$

for a minimization problem, and assuming that all points are positive in all objectives. On this basis, the unary multiplicative epsilon indicator, $I_\varepsilon^1(A)$ can then be defined as

$$I_\varepsilon^1(A) = I_\varepsilon(A, R), \tag{3.6}$$

where R is any reference set of points. An equivalent unary additive epsilon indicator $I_{\varepsilon+}^1$ is defined analogously, but is based on additive ε-dominance

$$\boldsymbol{z}^1 \preceq_{\varepsilon+} \boldsymbol{z}^2 \iff \forall i \in 1, \ldots, p : z_i^1 \leq \varepsilon + z_i^2.$$

Both unary indicators are to be minimized. An indicator value smaller than 1 (I_ε^1) respectively 0 ($I_{\varepsilon+}^1$) implies that A strictly dominates the reference set R.

For the unary epsilon indicators, it holds that whenever $A \triangleleft B$, then $I_\varepsilon^1(A) \leq I_\varepsilon^1(B)$ respectively $I_{\varepsilon+}^1(A) \leq I_{\varepsilon+}^1(B)$. Unlike the hypervolume indicator, it may happen that $I_\varepsilon^1(A) = I_\varepsilon^1(B)$ but A is better than B, or vice versa, actually. However, like the hypervolume indicator, from $I_\varepsilon^1(A) > I_\varepsilon^1(B)$ respectively $I_{\varepsilon+}^1(A) > I_{\varepsilon+}^1(B)$, one can deduce that A is not better than B. In some cases, the hypervolume and epsilon indicators may return opposite preference orderings for a pair of approximation sets A and B. An example of such case is

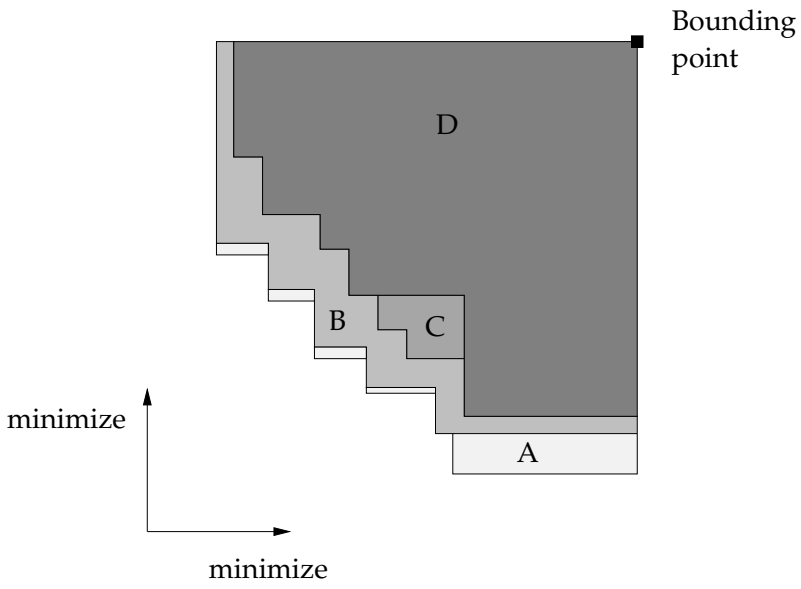


Figure 62: The hypervolume indicator measures the size of the dominated region, bounded by some reference point. Here four different sets A, B, C, D are shown by increasing order of darkness of the shaded region, with $A \lhd B \lhd C \lhd D$. A is different to B mainly in the extent, B is better than C in proximity to the Pareto front, and C is better than D mainly in evenness. Thus, the hypervolume indicator is capable of detecting differences in any of these different aspects.

given in Figure 63. From such a result, it logically follows that the two sets A and B are incomparable, because the epsilon and the hypervolume indicators are Pareto-compliant.

The situation is different if the indicators used are Pareto non-compliant. It is possible that all such indicators judge A being preferable to B, when in fact B is better than A according to Pareto dominance.

For any finite approximation set A, and any finite reference set R, the unary epsilon indicator is cheap to compute: the runtime complexity is of order $O(p \cdot |A| \cdot |R|)$, where p is the number of objectives.

Some important aspects need to be considered when quality indicators are to be used for sample transformation.

COMBINATION OF QUALITY INDICATORS. Different quality indicators normally reflect different decision-maker preferences. A combination of Pareto compliant indicators can yield interpretations that are more powerful than can be made by a single indicator alone. As we have already said, it two Pareto compliant indicators contradict one another on the preference ordering of two approximation sets, then this implies that the two sets are incomparable. This does not hold for Pareto non-compliant indicators.

SCALING AND NORMALIZATION. Scaling and normalization can be necessary in order to allow different objectives to contribute approximately equally to indicator values, for indicators such as I_H and $I^1_{\varepsilon+}$. We scaled each objective

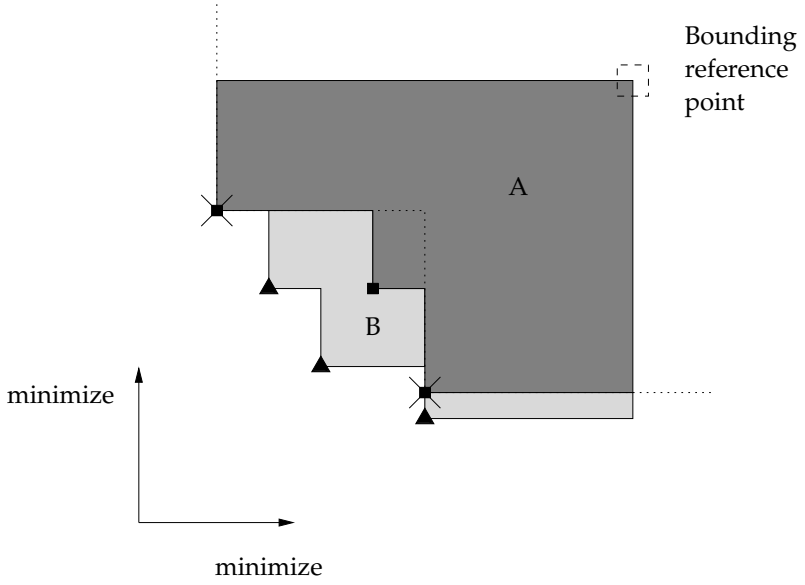


Figure 63: Two incomparable approximation sets A and B. Under the hypervolume indicator, B is the better set, but under the epsilon indicator A is better with respect to the reference set (the two X points connected by the dashed line), since $I_\varepsilon^1(A)$ is one, while $I_\varepsilon^1(B)$ is greater than one. This discrepancy indicates that the two sets are incomparable.

value between $[1, 2]$, because the epsilon indicator $I_{\varepsilon+}^1$ requires all objective values being strictly positive

$$z_i' = \frac{z_i - z^{\min}}{z_i^{\max} - z_i^{\min}} + 1 \tag{3.7}$$

where $z_i^{\min}$ and $z_i^{\max}$ are minimum and maximum objective values attained by the reference set under consideration.

REFERENCE POINTS AND SETS. The reference point $\boldsymbol{z}^+$ bounding the dominated region in the hypervolume indicator, see Figure 62, must be set in such a way that the objective vectors contained in the approximation sets $A_1, A_2, \ldots, A_r$ under consideration strictly dominate the reference point

$$\forall i = 1, \ldots, r, \quad \forall \boldsymbol{z} \in A_i \quad \forall j = 1, \ldots, p : z_j < z_j^+.$$

As to the reference approximation set in the context of the hypervolume and epsilon indicators, the Pareto front is the ideal reference because it can yield more power to the indicators. However, in parameter extraction problems the Pareto front is unknown. We then followed this approach:

- All approximation sets generated by the algorithms under consideration are combined, and then the dominated objective vectors are removed from this union. The remaining points, which are nondominated by any of the approximation sets, form the reference set.

The advantage of this method is that the reference set weakly dominates all approximation sets under consideration; however, whenever additional approximation sets are included in the comparison, the reference set needs to be re-computed.

3.7.3.2 *Empirical Attainment Function*

The attainment function of a multi-objective optimizer is a function $\alpha\colon \mathbb{R}^p \to [0,1]$ such that, for each objective vector $\boldsymbol{z} \in \mathbb{R}^p$, $\alpha(\boldsymbol{z})$ is the probability that the optimizer attains objective vector $\boldsymbol{z}$ in a single run. Each objective vector attained by an optimizer is also referred to as an attained *goal*. The attainment function is a first order moment measure. It describes the location of the approximation set distribution generated by the optimizer. Higher order moments are needed if the variability across runs is to be assessed, and to assess dependencies between the probabilities of attaining two or more goals *in the same run* [36].

The attainment function can be estimated from a sample of r independent runs of an optimizer via the Empirical Attainment Function (EAF) defined as

$$\alpha_r(\boldsymbol{z}) = \frac{1}{r}\sum_{i=1}^{r} \mathbf{I}(A^i \leqq \boldsymbol{z}) \tag{3.8}$$

where A^i is the ith approximation set of the optimizer and $\mathbf{I}(\cdot)$ is the indicator function, which evaluates to one if its argument is true and zero if its argument is false. In other words, the EAF gives for each objective vector in the objective space the relative frequency that it was attained, i.e., weakly dominated by an approximation set, with respect to the r runs.

The outcomes of two optimizers can be compared by performing a corresponding statistical test on the resulting two EAFs, as will be explained in In addition, EAFs can also be used for visualizing the outcomes of multiple runs of an optimizer. For instance, one may be interested in plotting all the goals that have been attained (independently) in 50% of the runs. This is defined in terms of a *k%-approximation set*:

Definition 3.3. An approximation set A is called the *k%-approximation set* of an EAF $\alpha_r(\boldsymbol{z})$, if and only if it weakly dominates exactly those objective vectors that have been attained in at least k percent of the r runs.

We can then plot the *attainment surface* of such an approximation set, defined as:

Definition 3.4. An attainment surface of a given approximation set A is the union of all tightest goals that are known to be attainable as a result of A. Formally, this is the set $\left\{ \boldsymbol{z} \in \mathbb{R}^p \;\middle|\; A \leqq \boldsymbol{z} \wedge A^{\mathrm{C}} < \boldsymbol{z} \right\}$.

Roughly speaking, the k%-attainment surface divides the objective space in two parts: the goals that have been attained and the goals that have not been attained with a frequency of at least k percent.

Example 3.5. Suppose a stochastic multi-objective optimizer returns the approximation sets depicted in Figure 64 for five different runs on a biobjective optimization problem. Minimization for both the objectives is assumed. The corresponding attainment surfaces are shown in Figure 65; they summarize the underlying empirical attainment function.

Using EAFs to produce a visual representation of an optimizer's performance is effective and complementary to the use of quality indicators for the following reasons:

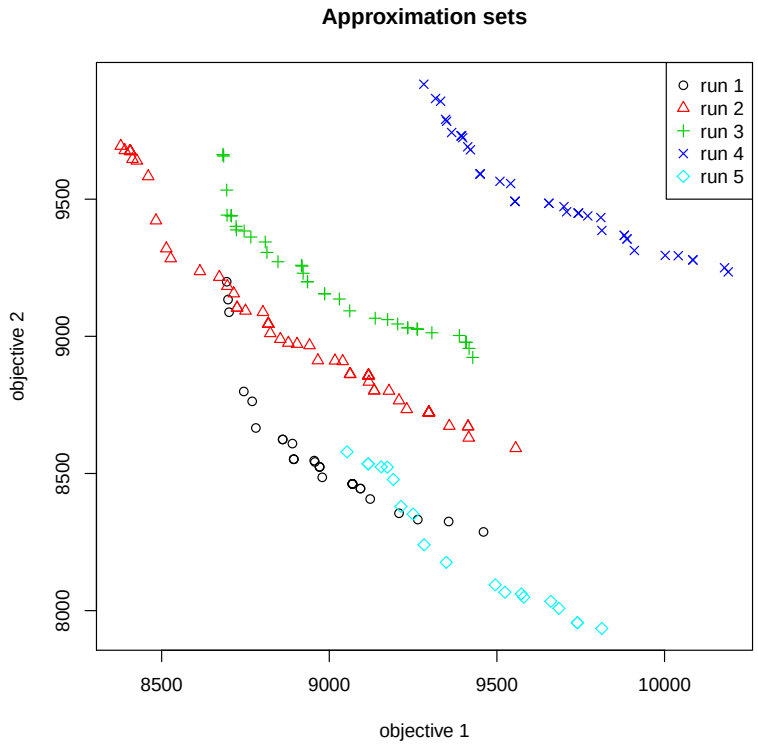


Figure 64: A plot showing five approximation sets. The two objectives are to be minimized.

- Decision makers may have preferences towards certain regions of or shapes of Pareto front, not generally (or easily) expressible before optimization.
- Some quality indicators do not adequately express the amount by which one approximation set should be judged better than another.
- Looking at approximation set shape can provide insight into the strengths and weaknesses of an optimizer, or provide information about how it is working. For example, an optimizer may converge well in the center of the Pareto front only, or more at the extremes, perhaps depending on how it balances elitism and diversity mechanisms.
- Visualization methods can provide a "sanity check" to validate any quality indicators being used.

The attainment function approach distinguishes itself from the dominance ranking and indicator approaches by the fact that the transformed sample are multidimensional, i.e., defined on the objective space and not on $\mathbb{R}$. Thereby less information is lost by the transformation. However, the approach is computationally expensive and therefore only applicable in the case of a few objective functions. In this work, the EAF approach has been applied only to a biobjective parameter extraction problem.

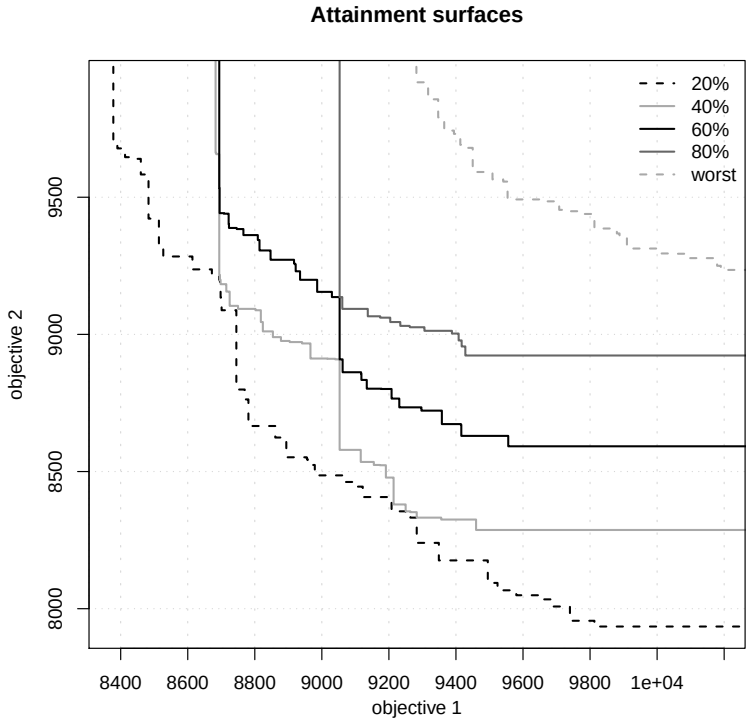


Figure 65: Attainment surface plots for the approximation sets in Figure 64. Both the objectives are to be minimized. Since we considered five runs, the 20%-attainment surface corresponds to the tightest goals attained in at least one run, the 40%-attainment surfaces corresponds to the tightest goals attained in at least two runs, and so on.

3.7.4 *Statistical Testing*

The previous section has described two different transformations that we applied to samples of approximation sets generated from multiple runs of an optimizer. The ultimate purpose of generating the samples and applying the transformations is to (a) describe and (b) make inferences about the underlying random approximation set distributions of the (two or more) optimizers, thus enabling us to compare their performance.

Boxplots are useful to summarize a random sample from a distribution. In Chapter 4 they will be used to supplement statistical inferences made from indicator samples.

The statistical inference we would like to make, if it is true, is that one optimizer's underlying approximation set *distribution* is better than another one's. This fact cannot be determined definitively because we only have access to finite-sized *samples* of approximation sets.

It is standard practice to assume that the data is consistent with an explanation known as the *null hypothesis*, H_0, and then test how likely this is to be true, given the data H_0 will often state that two samples A and B are drawn from the same distribution or from distributions with the same mean value. The probability of obtaining a finding at least as "impressive" as that obtained, assuming the null hypothesis is true, is called the *p-value* and is computed using an inferential *statistical test*. The *significance level*, often denoted as α, defines the largest acceptable p-value and represents a threshold that is user-defined. A p-value lower than the chosen significance level α then implies that there is enough statistical evidence to reject the null hypothesis in favor of an *alternative hypothesis*, H_A, *at a significance level of* α. The definition of the alternative hypothesis usually takes one of two forms. If H_A is of the form "sample A comes from a better distribution than sample B" then the statistical test is a *one-tailed test*.If $H_A$is of the form "sample A and sample B are from different distribution" then it is a *two-tailed* test. The testing procedure that we followed to compare MOEA's performance on parameter extraction problems is based on two-tailed tests.

Statistical inference about experimental data is sometimes performed by using *parametric statistical tests*. These tests are based on assuming the data is drawn from a distribution that closely approximates a known distribution, e.g. the normal distribution or Student's t distribution. Such known distributions are completely defined by their parameters. Parametric tests are powerful–that is, the null hypothesis is rejected in most cases where it is indeed false. However, the assumption of normality cannot be theoretically justified for stochastic optimizer outputs, and it is difficult to empirically test for normality with relatively small samples (less than 100 runs). Therefore it is safer to rely on *nonparametric tests* [15], which make no assumptions about the distributions of the variables.

Two main types of nonparametric tests exist: rank tests and permutation tests. Rank tests are the less powerful but are also less sensitive to outliers and computationally cheap. Permutation tests are more powerful but they can be expensive to compute for large samples.

In the following, we describe the methods we used for nonparametric inference testing. We will also briefly discuss issues relating to matched samples and multiple inference testing.

3.7.4.1 *Comparing Sample Indicators Values*

In this work, the hypervolume and the epsilon indicators, which are *Pareto compliant*, have been used to compare the approximation sets. Different Pareto compliant indicators assess slightly different preferences and this helps to build up a better picture of overall approximation set quality. However, using several indicators does bring into play two possible pitfalls that should be avoided:

- If the locations, e.g., means, of the distributions from two indicators are being tested independently, the independence of the two results should be stated explicitly. For example, a statement of the form "the approximation sets of optimizer A are preferable to those of B under both I_1 and I_2" is not correct, even if the mean values are significantly higher for both indicators. This is because one could interpret that *individual* approximation sets tend to exhibit a higher quality under *both* indicators, while this is not necessarily true. A correct statement would be: "the approximation sets of optimizer A are preferable to those of B under I_1 and, independently, under I_2".
- Multiple testing issues, arising, as in this work, from using the same set of samples for multiple comparisons, are dealt with (see Section 3.7.4.4).

3.7.4.2 *Comparing Empirical Attainment Functions*

The EAF of an optimizer is a generalization of a univariate Empirical Cumulative Distribution Function (ECDF) [19]. In order to test if two ECDFs are different, the Kolmogorov-Smirnov (KS) test can be applied [15]. This test measures the maximum difference between the ECDFs and assesses the statistical significance of this difference. It does not determine whether one algorithm's entire EAF is "above" the other one

$$\alpha_r^A(\boldsymbol{z}) \geq \alpha_r^B(\boldsymbol{z}) \quad \forall \boldsymbol{z} \quad \text{in the objective space,}$$

or not. Visualizing the EAFs allows checking for such specific differences. For two-objective problems, plotting significant differences in the empirical attainment functions of two optimizers, using a pair of plots, can be done by color-coding either:

- levels of difference in the sample probability
- levels of statistical significance of a difference in sample probability, of attaining a goal, for all goals.

For simplicity of implementation, we pursued the first option, even if the second one is more informative.

An example of such a pair of plots is shown in Figure 66.

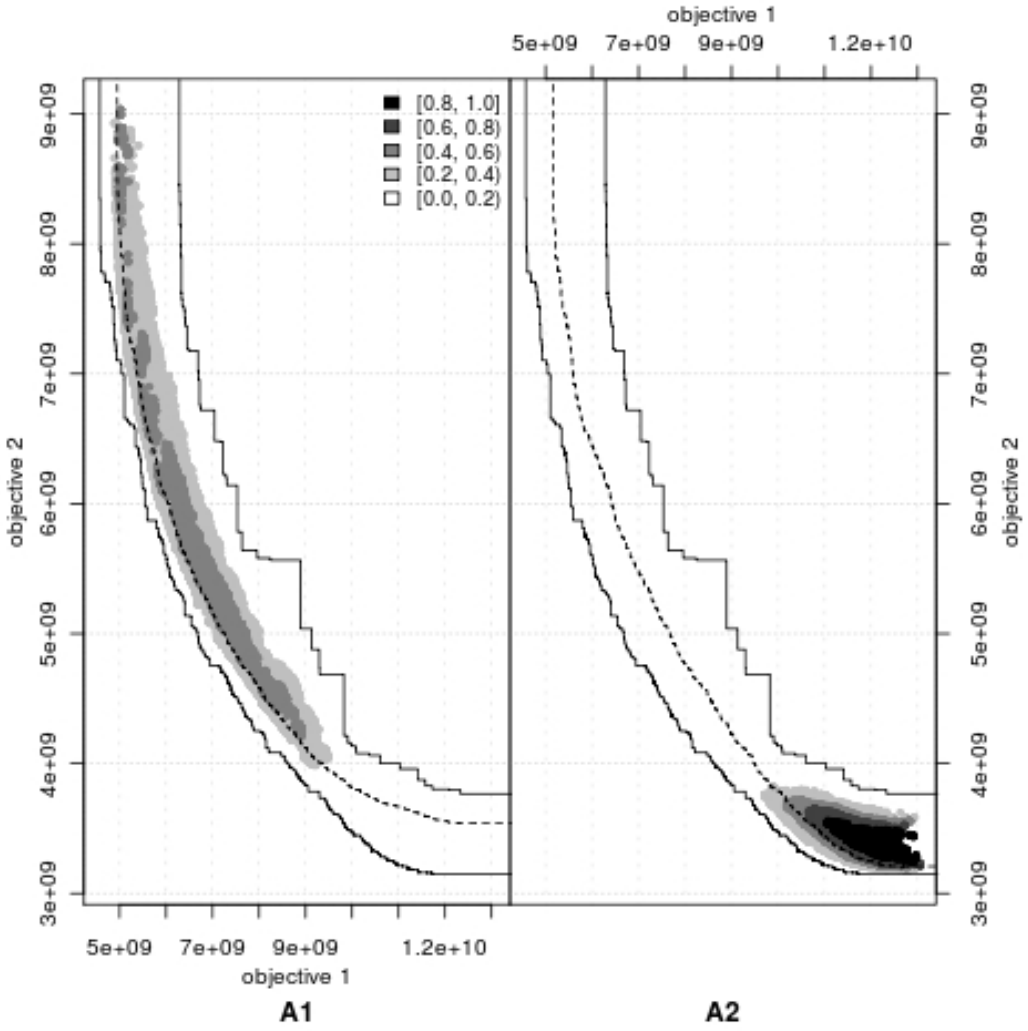


Figure 66: Individual differences between the probabilities of attaining different goals on a two-objective minimization problem with optimizer A_1 and optimizer A_2, shown using a color-coded plot. The figure also shows the grand best and worst attainment surfaces (the same in both plots), that is, the borders beyond which the goals are never attained or always attained, computed from the *combined* collection of approximation sets. This two surfaces delimit the area where differences may exist. In the left plot, darker regions indicate goals that are attained more frequently by A_1 than by A_2. In the right plot, the reverse is shown. In addition, the 50%-attainment surface of each algorithm is also plotted.

3.7.4.3 *Matched Samples*

When comparing stochastic optimizers, there are two slightly different situations. In one case, each run of each optimizer is a completely independent random sample; that is, the initial population, the random seed, and all other random variables are drawn independently and at random on each run. In the other case, the influence of one or more random variables is partially removed from consideration. In our work, we followed the latter strategy. The initial population and the random seed used by the algorithms have been matched in corresponding runs, so that the runs (and hence the final quality indicator values) have been taken together. In the former situation, if the statistical testing detects a difference in the distributions of indicator values resulting from the stochastic optimizers, only a general performance difference can be stated. In the latter situation the statistical testing reveals whether there is a difference in the indicator value distributions *given the same initial population*, and the inference in this case relates to each optimizer's ability to *improve* the initial population. While the former scenario is more general, the latter may give more statistically significant results.

If matched samples have been collected, one can use the Wilcoxon signed rank test [15] for comparison of two optimizers and the Friedman test with the corresponding post-hoc tests for comparison of more optimizers, as it is suggested in [28]. Below, the Friedman test is presented, since it has been used in the two problems shown in Chapter 4.

FRIEDMAN TEST. The Friedman test is a nonparametric equivalent of the repeated measures ANOVA [15]. It ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second best rank 2, and so on. In case of ties average ranks are assigned. In our case, each group of matched runs and the corresponding indicator values corresponds to a single data set, in terms of the Friedman test.

Let r_i^j be the rank of the jth of k algorithms on the ith of N data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N}\sum_j r_i^j$. Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{3.9}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough (as a rule of a thumb, $N > 10$ and $k > 5$). For a smaller number of algorithms and data sets, exact critical values have been computed [15].

Iman and Davenport [54] showed that Friedman's χ_F^2 is undesirably conservative and derived a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{3.10}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

If the null hypothesis is rejected, we can proceed with a post-hoc test in order to find the concrete pairwise comparisons which produce differences. The post-hoc procedure involves comparing all optimizers with each other. When so many tests are made, a certain proportion of the null hypothesis is rejected due to random chance.

Multiple hypothesis testing is a complicated issue. We will only touch on the correct procedure in Section 3.7.4.4.

3.7.4.4 *Multiple Testing*

The confidence levels resulting from a statistical testing procedure for measuring the differences between distributions only has a meaning if certain assumptions are true. One of these assumptions is that the data on which the test has been carried out is not being used to make more than one inference. Imagine the same set of data was used to make five different inferences, and each had a significance level α of 0.05. The chance that at least one of the inferences will be a type-1 error (i.e. the null hypothesis is wrongly rejected) is $1 - (0.95^5) \simeq 23\%$, when assuming that the null hypothesis was true in every case.

Multiple testing issues in the case of assessing stochastic multiobjective optimizers can arise for at least two different reasons:

- There are more than two algorithms and we wish to make inferences about the performance differences between all or a subset of them.
- There are multiple hypothesis that we wish to test with the *same* data, e.g. differences in the distributions of more than one indicator.

The issue of multiple hypothesis testing is a well-known statistical problem. We considered two approaches to minimize the problem:

- For each indicator to be investigated for distribution differences among the optimizers, new independent data has been generated which has not been used for any other indicator.
- For each indicator, pairwise comparisons of all MOEAs have been performed on the same data but we used methods for correcting the p-values for the reduction in confidence associated with data re-use.

In this work, statistical procedures proposed by García and Herrera [40] for comparing $k \times k$ algorithms have been used. In the following, these procedures are shortly described.

A set of pairwise comparisons can be associated with a set or family of hypotheses. As it is explained in [28] explained, the test statistics for comparing the ith and jth classifier is

$$z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}}$$

being R_i the average rank computed through the Friedman test for the ith classifier, k the number of algorithms to be compared and N the number of data sets used in the comparison.

The z value is used to find the corresponding p-value from the table of normal distribution, which is then compared with an appropriate level of significance α. Two basic procedures are:

- Nemenyi [40] procedure: it adjusts the value of α in a single step by dividing the value of α by the number of comparisons performed, $m = k * (k-1)/2$. This procedure is the simplest but it also has little power.

- Holm [40] procedure: it adjusts the value of α in a step down method. Let $p_1, \ldots, p_m$ be the ordered p-values (smallest to largest) and $H_1, \ldots, H_m$ be the corresponding hypotheses. Holm's procedure rejects H_1 to $H_{(i-1)}$ if i is the smallest integer such that $p_i > \alpha/(m-i+1)$. Other alternatives were developed by Hochberg [49], Hommel [50] and Rom [84]. They are easy to perform, but they often have a similar power to Holm's procedure when considering all pairwise comparisons.

The hypotheses being tested belonging to a family of all pairwise comparisons are logically interrelated so that not all combinations of true and false hypotheses are possible. For example, suppose that we want to test the three hypotheses of pairwise equality associated with the pairwise comparisons of three optimizers $A_i, i = 1,2,3$. From the relations among the hypotheses, it can be seen that if any one of them is false, at least one other must be false. For example, if A_1 is better/worse than A_2, then it is not possible that A_1 has the same performance as A_3 and A_2 has the same performance as A_3 . A_3 must be better/worse than A_1 or A_2 or the two classifiers at the same time. Thus, there cannot be one false and two true hypotheses among these three.

Based on this argument, Shaffer proposed two procedures which use the logical relation among the family of hypotheses for adjusting the value of α [86].

- Shaffer's static procedure: following Holm's step down method, at stage i, instead of rejecting H_i if $p_i \leq \alpha/(m-i+1)$, reject H_i if $p_i \leq \alpha/t_i$, where t_i is the maximum number of hypotheses which can be true given that any $(i-1)$ hypotheses are false. It is a static procedure, that is, $t_1, \ldots, t_m$ are fully determined for the given hypotheses $H_1, \ldots, H_m$, independent of the observed p-values.

- Shaffer's dynamic procedure: it increases the power of the first by substituting α/t_i at stage i by the value α/t_i^*, where t_i^* is the maximum number of hypotheses that could be true, given that the previous hypotheses are false. It is a dynamic procedure since t_i^* depends not only on the logical structure of the hypotheses, but also on the hypotheses already rejected at step i. In this work, this procedure has not been used, since it is included in the method proposed by Bergmann and Hommel [6].

In [6] a procedure was proposed based on the idea of finding all elementary hypotheses which cannot be rejected. Let us consider the following definition:

Definition 3.6. An index set of hypotheses $I \subseteq \{1, \ldots, m\}$ is called exhaustive if exactly all $H_j, j \in I$, could be true.

Under this definition, Bergmann-Hommel procedure rejects all H_j with j not in the *acceptance set* A, which is defined as

$$A = \bigcup \{ I \mid I\text{exhaustive}, \min \{ p_i \mid i \in I \} > \alpha/|I| \}.$$

Therefore, the acceptance set is the index set of null hypotheses which are retained. For this procedure, one has to check for each subset I of $\{1, \ldots, m\}$ if I is exhaustive, which leads to intensive computation. Let E be the set containing all the possible exhaustive sets of hypotheses for a certain comparison. A rapid algorithm for computing E was described in [51]. Once the E set is obtained, the hypotheses that do not belong to the A set are rejected.

In order to control the *family-wise error*, i.e. the probability of making at least one type I error in any of the comparisons, instead of adjusting the value of α, p-values can be modified to take into account that multiple tests are conducted. Such modified p-values are called adjusted p-value (APV)s. An APV can be compared directly with any chosen significance level α.

In the following, we show how to compute the APVs for some of the post-hoc procedures described before. The following notation is used:

- Indexes i and j correspond each one to a concrete comparison or hypothesis in the family of hypotheses, according to an incremental order by their p-values. Index i refers to the hypothesis in question whose APV is being computed and index j refers to another hypothesis in the family.
- p_j is the p-value obtained for the jth hypothesis.
- m is the number of possible comparisons in an all pairwise comparisons of k optimizers, that is, $m = \frac{k(k-1)}{2}$.
- t_j is the maximum number of hypotheses which can be true given that any $(j-1)$ hypotheses are false.

Methods of p-value adjustment:

- Nemenyi APV_i: $\min\{v, 1\}$, where $v = m \cdot p_i$;
- Holm APV_i: $\min\{v, 1\}$, where $v = \max \{ (m-j+1)p_j \mid 1 \leq j \leq i \}$;
- Shaffer static APV_i: $\min\{v, 1\}$, where $v = \max \{ t_j p_j \mid 1 \leq j \leq i \}$;
- Bergmann-Hommel APV_i: $\min\{v, 1\}$, where

$$v = \max \{ \mid I| \cdot \min \{ p_j \mid j \in I \} \mid I\text{exhaustive}, i \in I \}.$$

In [40] García and Herrera give some recommendations:

- Nemenyi's test is very conservative and may not detect many of the obvious differences. It should not be used.
- Since the benefit of using information about logically related hypotheses is noticeable, the use of Shaffer static procedure is encouraged.
- Bergmann-Hommel's procedure is the best performing one, but it is also the most computationally expensive. It may be useful to use it when the differences among the optimizers compared are not very significant.

4

EXPERIMENTS

In this chapter, the problem of parameter extraction for Extended Lauritzen model is addressed. Some results have already been discussed in [5, 82].

The complete extraction procedure is outlined in Section 1.6. Here we focus on the use of four MOEAs for the simultaneous optimization of several diode output characteristics:

- NSGA-II;
- R-NSGA-II;
- MO-CMA-ES with population-based step size adaptation, named as MO-CMA-ES-P;
- MO-CMA-ES with population-based step size adaptation and recombination of learning strategy parameters, named as MO-CMA-ES-P-REC.

These algorithms have been described in Chapter 3. The first two optimizers were chosen because NSGA-II is currently used in most MOEA comparisons. The MO-CMA-ES is a promising algorithm for multi-objective optimization. In [94] and [93], it is shown that the population-based step size update rule and the recombination strategy clearly improve its performance. The study of the impact of both these procedures on the performance of the MO-CMA-ES is an ongoing work [94].

Two MOEA experiments were designed in order to investigate if these algorithms could find fast, high-quality solutions to this parameter extraction problem, and answer the following questions:

- Is there any difference in the performance of selected MOEAs?
- Is there a best overall method for the parameter extraction problem?
- What is the effect of considering more than two objective functions on the performance of each optimizer?

The design of every MOEA experiment must conform to an accepted "standard" approach as reflected in any scientific method. When employing the scientific method, the detailed design of MOEA experiments can draw heavily from outlines presented by Barr et al. [4]. This generic article discusses computational experiment design for heuristic methods, providing guidelines for reporting results and ensuring their reproducibility. On the basis of these guidelines:

- optimization codes for the selected MOEAs have been developed;
- a statistical experimental design has been devised to evaluate the relative efficiencies of the algorithms;
- a rigorous statistical analysis of the performance of the algorithms has been implemented following the procedures described in Section 3.7.

SYMBOL	DESCRIPTION	UNIT
C_{jo}	Zero bias junction capacitance	F
F_c	Forward bias depletion capacitance coefficient	
m	Grading coefficient	
R_s	Series resistance	Ω
T_{nom}	Parameter measurement temperature	°C
v_j	Built-in potential	V
α	Hole mobility coefficient	
T_{R_s}	Temperature dependent coefficient of series resistance	K^{-1}
T_{E_r}	Temperature dependent coefficient of E_r	K^{-1}
T_{τ_3}	Temperature dependent coefficient of τ_3	
T_{no}	Electron transit time	s
τ_L	Carrier lifetime at low electric field values	s
τ_H	Carrier lifetime at high electric field values	s
Q_B	Thermal equilibrium electron charge in the N^- region	C
Q_{Bp}	Thermal equilibrium hole charge in the N^- region	C
NN	Low level ideality parameter	
NE	Medium-high level ideality parameter	
E_r	Symmetric end-region recombination parameter	s C
ϕ_B, I_B	Voltage-dependent reverse-recovery parameters	V, A
b	Mobility ratio of electrons and holes	
E_{th}	Threshold value of electric field	$V\,m^{-1}$
a_p, b_p	Impact ionization parameters	m^{-1}, $V\,m^{-1}$

Table 10: Extended Lauritzen model, 2011. Parameter list.

4.1 PROBLEM STATEMENT AND PRELIMINARY STUDIES

The main goal was to find a good parameter set for Extended Lauritzen model in order to fit four separate characteristics:

- junction capacitance vs. reverse bias voltage;
- dc forward-bias i–v characteristic;
- reverse recovery current and reverse recovery voltage at a load current of 6000 A and a supply voltage of 3200 V.

The reference temperature was 140 °C. Data to be fitted was obtained from TCAD simulations calibrated in order to closely reproduce the behavior of an ABB fast diode. In the following, the term "measurements" will refer to this set of data.

The complete parameter list of Extended Lauritzen model is recalled in Table 10 for convenience. While using this model, the following choices were made:

- The forward bias depletion capacitance coefficient was set to its default value in the SPICE model $F_c = 0.5$ [1].
- Validity of equations for temperature dependence of model was not verified. For this reason, α, T_{R_s}, T_{E_r} and $T\tau_3$ have never been changed.
- The voltage-dependent reverse recovery parameter ϕ_B was substituted by $R_B = \phi_B / I_B$ in the model implementation.
- The threshold value of electric field was devised empirically

$$E_{th} = 1.3 \times 10^5 \,\mathrm{V\,m^{-1}}.$$

- The impact ionization parameter b_p was set to its initial estimate 1×10^5. A precise estimate of b_p is not necessary as we now explain. In Extended Lauritzen model, the impact ionization current is given by equation (1.68), which is recalled below for convenience

$$i_{ii} = i_{p23} a_p x_n \exp(-b_p / E(x_n)). \tag{4.1}$$

  In equation (4.1), very small changes in parameter b_p can be compensated by the multiplicative factor a_p. A rough estimate of b_p is therefore enough. Moreover, parameter b_p should not be optimized. Since it is the argument of an exponential function, a significant change of parameter b_p could only be compensated by a change of a_p by several orders of magnitudes. In our experience, using a physical range for a_p which is several decades wide would influence all the other model parameter and it would make very difficult for the optimizer to explore the parameter space.

Sensitivity analysis (see Section 1.6.2) shows that the shape of the capacitance characteristic is mainly influenced by the zero bias junction capacitance C_{jo}, the grading coefficient m and the built-in potential v_{jo}, and that they do not affect the other characteristics noticeably. These results are not surprising, since the equations for modeling junction capacitance only depend on C_{jo}, m and v_{jo}. The parameters C_{jo}, m and v_{jo} were therefore extracted by optimizing the capacitance characteristic. The non-elitist CMA-ES (see Section 3.2.4) was used to minimize the RSM starting from the initial estimate which can be computed with the procedure described in Section 1.6.1. Explored parameter ranges for capacitance parameters are reported in Table 11, while the optimum found is depicted in Figure 67. Optimal values of C_{jo}, m and v_{jo} were fixed in subsequent optimizations.

Finally, the parameter space to be explored for fitting dc and transient characteristics is represented in Table 12. Its dimension is $n = 13$.

At this point, there are still three characteristics that have to be fitted:

- dc forward-bias i–v characteristic;
- reverse recovery current and reverse recovery voltage at a load current of 6000 A and a supply voltage of 3200 V.

Mathematically, this fitting problem was translated into a multi-objective minimization problem with three objectives:

PARAMETER	UNIT	LOWER BOUND	UPPER BOUND
C_{jo}	F	5.9×10^{-8}	1.5×10^{-7}
m		0.4	1
v_{jo}	V	0.4	1

Table 11: Ranges for Extended Lauritzen model parameters influencing the capacitance characteristic.

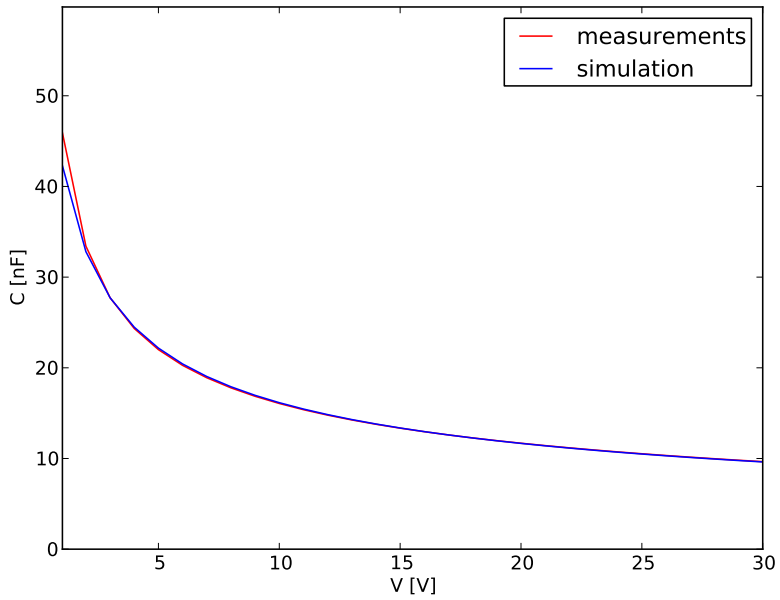


Figure 67: Comparison between measured and optimized simulated capacitance characteristic. The residual sum of magnitudes (RSM) was used as error estimator to make the CMA-ES not to focus on error minimization at low reverse voltage too much. The optimum found by CMA is: $C_{jo} = 7.798\,664 \times 10^{-8}\,\mathrm{F}, m = 0.483\,217\,2, v_{jo} = 0.4\,\mathrm{V}$.

PARAMETER	UNIT	LOWER BOUND	UPPER BOUND
R_s	Ω	2×10^{-4}	8×10^{-4}
T_{no}	s	1×10^{-6}	5×10^{-5}
τ_L	s	5×10^{-7}	5×10^{-6}
τ_H	s	5×10^{-7}	1×10^{-5}
Q_B	C	1×10^{-8}	5×10^{-4}
Q_{Bp}	C	1×10^{-12}	1×10^{-8}
NN		1	4
NE		1	12
E_r	sC	1×10^{-6}	1×10^{-3}
R_B	Ω	1	50
I_B	A	1×10^{2}	1×10^{4}
b		1	6
a_p	m^{-1}	1	50

Table 12: Ranges for Extended Lauritzen model parameters influencing dc and transient characteristics.

- RSM between measured and simulated dc data. This error estimator helps to redress the imbalance between errors at low voltage values, where current values are also low, and errors at high voltage values, where the current is very high.
- RSS between measured and simulated reverse recovery data for both current and voltage. These error estimators focus more on matching the switching losses that the on-state losses (see Section 1.6.4).

Sensitivity analysis shows that optimizations of dc and transient characteristics are clearly conflicting and that a conflict may also exist between optimizations of reverse recovery current and reverse recovery voltage.

In the first MOEA experiment, all the three objectives were retained. Results are reported in Section 4.3. They suggest that, although a conflict between optimizations of transient current and voltage may exist somewhere in the objective space, these two characteristics behave in a less-conflicting manner near the Pareto-optimal region. That is, fitting of reverse recovery voltage benefits from improvements in fitting reverse recovery current. This is not surprising, because current and voltage are physically coupled in the phenomena of turn-off. Therefore, in the second MOEA experiment only two objectives were considered: RSM between measured and simulated dc data, and RSS between measured and simulated reverse recovery current data. Results are shown in Section 4.4.

4.2 ALGORITHMIC ALTERNATIVES AND COMPUTING ENVIRONMENT

In this section, we present some structural characteristics of the four MOEA codes that were tested in this study. Additional material may be found in Appendix A.

All codes are written in Python 2.6 and C++. Programs are based on the *inspyred* Python library [42]. inspyred is a free, open source framework for creating biologically-inspired computational intelligence algorithms in Python, including evolutionary computation, swarm intelligence, and immunocomputing. In particular, the inspyred library provides basic components to easily build MOEAs.

NSGA-II. The NSGA-II is already available in the inspyred library. However we had to implement some corrections to the crowding distance assignment procedure. In [24], the Simulated Binary Crossover (SBX) operator [21] and polynomial mutation operator [22] were used for real-coded GAs. The SBX operator is available in the inspyred library, but its implementation is a little bit different from that proposed in [21]. For this reason, we wrote our own code for the SBX operator. We also wrote a code for the polynomial mutation operator, because it is not available in the current version of the inspyred library.

After performing the two MOEA results described above, we found another error in the implementation of NSGA-II available in the inspyred library: the binary tournament selection is only based on nondomination rank. Crowding is not used as second level sorting criterion. When randomly selecting two solutions $\boldsymbol{x}$ and $\boldsymbol{y}$, the selection operator chooses the solution with the lower rank. But if solutions are in the same nondominated front, the selector chooses the winner randomly instead of selecting the solution with a higher crowding distance. Our code did not correct this error in time. However, we believe that this error did not worsen the convergence of the NSGA-II to the true Pareto optimal front, because it shows similar performances to $R-NSGA-II$, whose selection operator has been correctly implemented, in both the MOEA experiments. In our opinion, only the diversity of solutions generated by this algorithm may have been affected somehow. In Figure 77, the 50%-attainment surfaces for the second test problem are shown. One could guess that the NSGA-II mainly explores the center of the Pareto frontier. This may be explained by the loss of solution diversity due to the error in our code.

R-NSGA-II. The R-NSGA-II is an extension of the NSGA-II for dealing with many-objective optimization problems [25]. Since it is not available in the inspyred library, we had to implement the modified crowding distance operator and tournament selection operator described in Section 3.6. All codes are written in Python 2.6. The interface of the NSGA-II was then used to integrate the algorithm in the inspyred library. The R-NSGA-II uses the same crossover and mutation operators employed by the NSGA-II.

MO-CMA-ES. The MO-CMA-ES is not available in the inspyred library. It is based on an elitist variant of the single-objective CMA-ES, named $(1+\lambda)$-CMA-ES, which is described in Section 3.2.5. The strategy adaptation mechanism of the

FEATURE	DESCRIPTION
Computer model	Acer Aspire 5738ZG
Processor	Intel© Pentium© mobile processor T4300 (1 MB L2 cache, 2.10 GHz, 800 MHz FSB), supporting Intel© 64 architecture
Operating system	Linux 2.6.32-41-generic-pae i686
Programming languages	Python 2.6.5. Additional packages: scipy 0.11.0, numpy 1.6.2, inspyred 1.0
	C++. Compiler: g++. Compiler optimization level: O3

Table 13: Description of the computing environment.

$(1 + \lambda)$-CMA-ES is combined with multi-objective selection using nondominated sorting and the contributing hypervolume as second sorting criterion.

We implemented the $(1 + \lambda)$-CMA-ES in Python. Instead, the computation of the contributing hypervolume is written in C++, because it is computationally expensive. The procedure is based on codes available in the Shark Machine Learning Library© [55]. We then use the C interface to Python, referred to as the Python C Application Programming Interface (API), to write an interface which allows to call C++ code from Python. The official electronic Python documentation has a tutorial for the C API, called "Extending and Embedding the Python Interpreter" [38], and a reference manual for the API [37]. The C API is also covered in books [67].

The population-based step size adaptation and the recombination of learning strategy parameters are written in Python.

The computing environment is described in Table 13.

4.3 FIRST TEST CASE

This section presents the design and analysis phases of the first MOEA experiment. It consisted in a multi-objective minimization problem with three objective functions:

- RSM between measured and simulated dc data;
- RSS between measured and simulated reverse recovery data for both current and voltage.

4.3.1 *Choosing MOEA parameters and quality measures*

In designing this experiment, the goal was to identify the relative quality of the solutions generated by the four MOEAs. In this manner, the experiment should provide some answers to the questions posed in the introduction of this chapter.

For each MOEA, we chose a reasonable set of values and made no effort in finding the best strategy parameter setting.

For NSGA-II and R-NSGA-II, we used a population size of 120 and a generation number of 250. We were guided in choosing these values by the test problems provided in the C library implementing the NSGA-II available at `http://www.iitk.ac.in/kangal/codes.shtml`. The chosen values represent a trade-off between exploration of the objective space and computational effort requested by the optimizers. As Deb et al. did in [24], we set the mutation probability to $p_m = 0.08$, which is slightly higher than the inverse of the decision space dimension, and the crossover probability to $p_c = 0.9$. The distribution indices of the mutation and crossover operators were set to $\eta_m = 20$ and $\eta_c = 10$, respectively. The same values were used in [25] to solve a pool of many-objective optimization problems. For the R-NSGA-II, we chose $\varepsilon = 0.001$ as niching parameter, like Deb et al. did in [25] to solve two engineering design problems, and the following three reference points (objectives are considered in the following order: RSM-dc, RSS-rr-current, RSS-rr-voltage):

1. $(50, 1 \times 10^{-4}, 1 \times 10^{-2})$: good solutions with respect to all the objectives are preferred;
2. $(50, 2, 2)$: solutions achieving a good fitting of dc i–v, but a bad fitting of reverse current and reverse voltage are preferred;
3. $(1 \times 10^{4}, 1 \times 10^{-4}, 0.07)$: solutions achieving a good fitting of reverse current and reverse voltage, but a bad fitting of dc i–v are preferred.

The reference points were chosen after a series of single-objective optimizations of each characteristic.

In the two variants of the MO-CMA-ES, we set the population size μ to 100, the number of offspring per parent λ to 1, and the generation number to 250 to allow for a better comparison with the other two MOEAs. For the strategy parameters of each individual implementing the $(1 + \lambda)$-CMA-ES we used the same setting as Igel et al. [53]. In particular, since we rescaled all the decision variables between 0 and 1, the initial step size $\sigma^{(0)}$ was set to 0.3 [48].

The quality indicators for comparing the optimizers were the hypervolume difference to a reference set I_H^- and the unary additive epsilon indicator $I_{\varepsilon+}^1$. For both the indicators, smaller values correspond to higher quality. The reference set was generated by combining all approximation sets generated by the optimizers and then selecting the nondominated points of this union.

4.3.2 *Design of the experiment*

For each indicator to be investigated for distribution differences among MOEAs, new initial populations, and hence new approximation set samples, were generated which were not used for any other indicator. Moreover, the initial population and the random seed used by the optimizers were matched in corresponding runs, so that the runs, and hence the final quality indicator values, were taken together (see Section 3.7.4.3). For each indicator-MOEA pair, 30 MOEA runs were performed. The size of the approximation set samples used for the statistical test-

ing procedure described in Section 3.7.4 was therefore 30. The total number of initial populations to be generated n_{pop} is then given by

$$\begin{aligned} n_{\text{pop}} &= \text{No. of indicators} \cdot \text{sample size} \\ &= 2 \cdot 30 \\ &= 60, \end{aligned}$$

while the total number of runs n_{runs} is equal to

$$\begin{aligned} n_{\text{runs}} &= \text{No. of indicators} \cdot \text{sample size} \cdot \text{No. of MOEAs} \\ &= 2 \cdot 30 \cdot 4 \\ &= 240. \end{aligned}$$

Algorithm 4.1 Generation of initial populations for a MOEA comparison.

Input: Random number generator *rng-r* from Python module `random`, random number generator *rng-n* from Python module `numpy.random`, number of initial populations to be generated n_{pop}, size of approximation set samples s, number of device model parameters n_{par}, lower and upper bounds of model parameters lb, ub

Output: List of initial populations L

1: Set the internal state of *rng-r* with a seed provided by the user
2: Initialize the population list $L \leftarrow \emptyset$
3: **for** $i = 1, \ldots, n_{\text{pop}}$ **do**
4: $\quad x \leftarrow$ *rng-r*.random() ▷ $x \sim U(0,1)$
5: $\quad y \leftarrow$ *rng-r*.randint$(0,9)$ ▷ $y \sim U\{0,1,\ldots,8\}$
6: $\quad w \leftarrow x \cdot 10^y$
7: $\quad$ Set the internal state of *rng-n* with w
8: $\quad$ Initialize population P_i, $P_i \leftarrow \emptyset$
9: $\quad$ **for** $j = 1, \ldots, s$ **do**
10: $\quad\quad$ Initialize individual $\boldsymbol{c}_j$, $\boldsymbol{c}_j \leftarrow \mathbf{0}$
11: $\quad\quad$ **for** $k = 1, \ldots, n_{\text{par}}$ **do**
12: $\quad\quad\quad c_{j,k} \leftarrow$ *rng-n*.uniform$(\text{lb}_k, \text{ub}_k)$ ▷ $c_{j,k} \sim U(\text{lb}_k, \text{ub}_k)$
13: $\quad\quad$ **end for**
14: $\quad\quad P_i \leftarrow P_i \bigcup \{\boldsymbol{c}_j\}$
15: $\quad$ **end for**
16: $\quad L \leftarrow L \bigcup \{P_i\}$
17: **end for**

In the following, the randomized procedure to set up the MOEA experiment is described. The notation $U(a,b)$ represents a continuous uniform distribution on $[a,b]$, while $U\{1,2,\ldots,n\}$ represents a discrete uniform distribution on the integers from 1 to n.

1. Instantiate a random number generator from Python module `random` and another one from module `numpy.random`. Let us refer to these two generators as *rng-r* and *rng-n*, respectively.

2. Generate initial populations. Algorithm 4.1 illustrates the complete procedure. The seed mentioned at line 1 was set to 7297463985.

Algorithm 4.2 Generate random seeds for MOEA runs.

Input: Random number generator *rng-r* from Python module `random`, number of initial populations n_{pop}
Output: List of MOEA random seeds R

1: Set the internal state of *rng-r* with a seed provided by the user
2: Initialize R, $R \leftarrow \emptyset$
3: **for** $i = 1, \ldots, n_{\text{pop}}$ **do**
4: $\quad x \leftarrow \textit{rng-r}.\text{random}()$ ▷ $x \sim U(0,1)$
5: $\quad y \leftarrow \textit{rng-r}.\text{randint}(0,9)$ ▷ $y \sim U\{0,1,\ldots,8\}$
6: $\quad w \leftarrow x \cdot 10^y$
7: $\quad R \leftarrow R \bigcup \{w\}$
8: **end for**

3. Generate random seeds for matched MOEA runs. The complete procedure is given in Algorithm 4.2. The seed mentioned at line 1 was set to 334961315. Since both the initial population and the random seed used by the optimizers were matched in corresponding runs, we generated as many seeds as matched MOEA runs.

4. Shuffle initial populations. The internal state of *rng-r* is set through a user-provided seed and initial populations are shuffled. For this step, the seed was 498431465.

5. Shuffle random seeds for MOEAs. The internal state of *rng-r* is set through a user-provided seed and MOEA random seeds are shuffled. As a seed, we used 6674654.

6. Create a list of n_{pop} indicator names. Each name refers to the indicator being computed for a single final approximation set. Then shuffle this list. The internal state of *rng-r* is set through a user-provided seed and indicator names are shuffled. As a seed, we used 29843216513.

7. Match the lists of shuffled populations, MOEA random seeds and indicator names element; pair each element of the resulting list with each MOEA in order to define a list of matched MOEA runs.

8. Shuffle the list of MOEA runs. The internal state of *rng-r* is set through a user-provided seed and indicator names are shuffled. As a seed, we used 54431774111.

Finally, the Python package `multiprocessing` was used to perform shuffled MOEA runs in parallel on our two-core machine.

4.3.3 *Analysis of the results*

In this section, performance assessment methodologies described in Section 3.7 are employed.

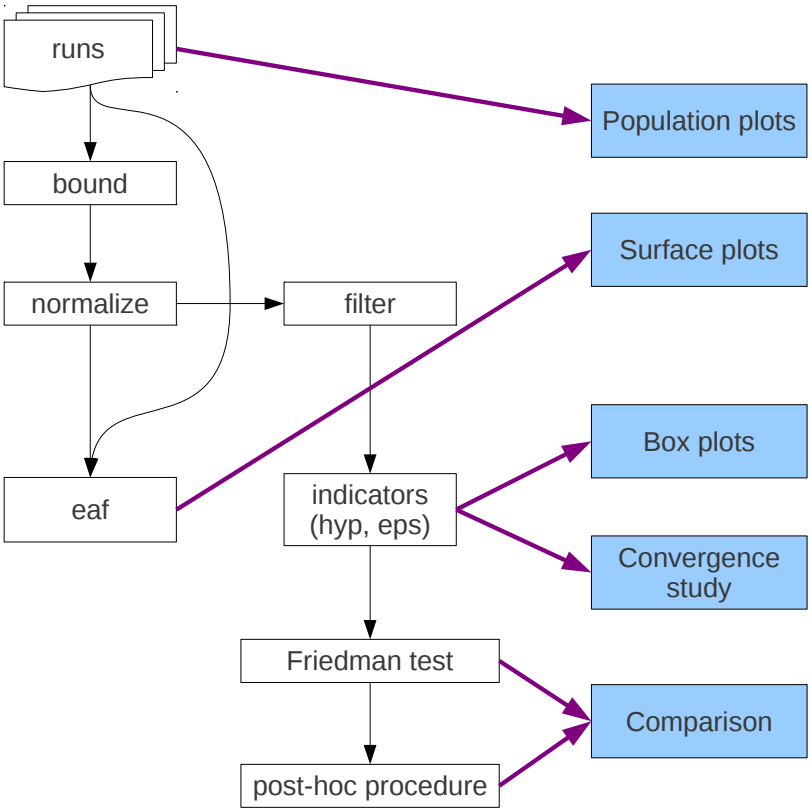


Figure 68: MOEA performance assessment flow.

The starting point is the set of runs for all combinations of quality indicator to compute, optimization methods and initial populations. For each run we obtain three files:

- one file contains all objective vectors explored by the optimizer;
- another file contains the final approximation set;
- the last file contains the approximation set computed at each generation.

According to Figure 68, one can at first try to visualize individual runs using approximation set plots. This can be easily done in biobjective problems (see example 3.5). In the current MOEA experiment, since we had three objectives, these graphs were not used.

There are three procedures that perform a post-processing on the approximation sets found by the optimizers:

BOUND. This procedure calculates lower and upper bounds of the objective vectors. To this end, all objective vectors explored by all the optimizers are collected, and the maximal and minimal values in each objective dimension are determined.

ALGORITHM	RANKING
nsga2	$1.3\overline{6}$
r-nsga2	$1.\overline{6}$
mo-cma-es-p	$3.5\overline{6}$
mo-cma-es-p-rec	3.4

Table 14: First test case: average rankings of the algorithms with respect to the hypervolume indicator. Friedman statistic considering reduction performance (distributed according to chi-square with 3 degrees of freedom): 70.68. P-value computed by Friedman Test $\approx 5 \times 10^{-11}$. Iman and Davenport statistic considering reduction performance (distributed according to F-distribution with 3 and 87 degrees of freedom) ≈ 106.0932. P-value computed by Iman and Daveport Test $\approx 6 \times 10^{-29}$.

NORMALIZE. Based on the bounds determined above, this procedure transforms all objective vectors contained in a given file in such a way that all values lie in the interval $[1, 2]$.

FILTER. This procedure collects all normalized objective vectors from all populations and extracts the objective vectors which are nondominated. In this way, a reference set for the unary indicators is generated. The extraction of the nondominated vectors is carried out with the `filter` tool, which is part of the Platform and Programming Language Independent Interface for Search Algorithms (PISA), available for download from `http://www.tik.ee.ethz.ch/pisa/`.

As a next step, empirical attainment functions can be plotted to obtain performance information complementary to the use of quality indicators. We used two tools to compute and plot EAFs:

- the `eaf` program of the PISA;
- the `eaf` package of the R Project for Statistical Computing, available at `http://cran.r-project.org/web/packages/eaf/`.

These two programs are only designed for a biobjective problem. They will be used in the second test case in Section 4.4.

Based on the normalized approximation sets and the reference set, the unary hypervolume indicator I_H^- and the unary additive epsilon indicator $I_{\varepsilon+}^1$ are employed to compare the different MOEAs. The PISA performance assessment tools currently contain the programs `hyp_ind` and `eps_ind` for computing I_H^- and $I_{\varepsilon+}^1$, respectively. The distribution of the resulting transformed approximation sets are graphically visualized in Figure 69 and in Figure 70. These boxplots suggest that Pareto sets of MO-CMA-ES-P and MO-CMA-ES-P-REC are preferable to those computed by NSGA-II and R-NSGA-II under the hypervolume indicator and, independently, under the epsilon indicator.

The Friedman test (see Section 3.7.4.3) is used to compare indicator values of the four optimizers. Results of the comparisons over the hypervolume and the epsilon indicators are shown in Table 14 and in Table 15, respectively. Since the p-values

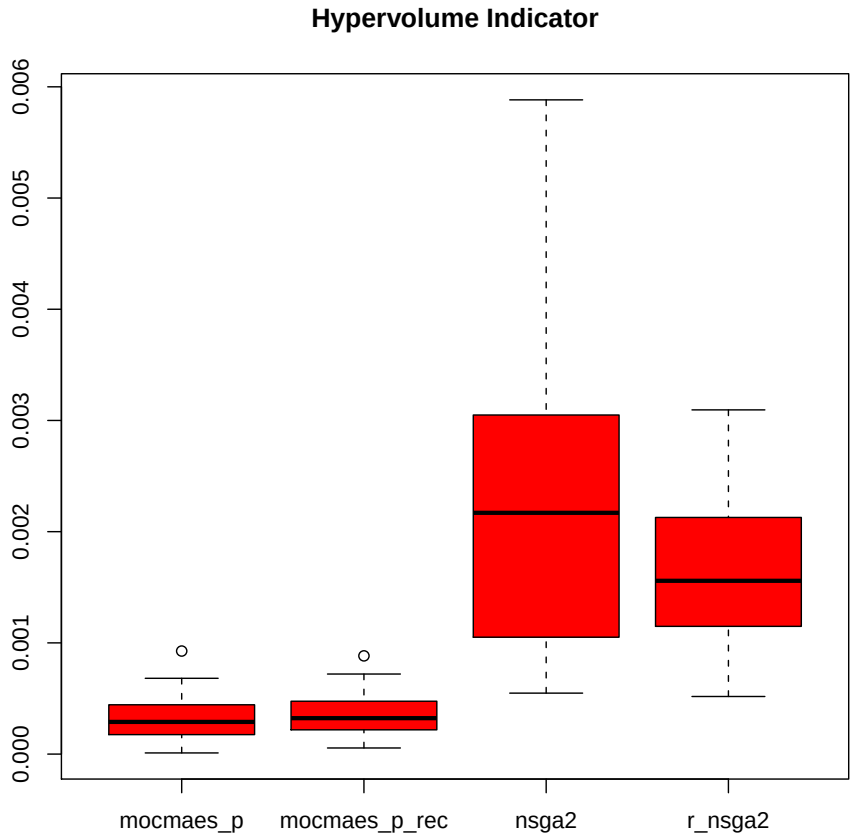


Figure 69: First test case: comparison of hypervolume indicator values through boxplots.

ALGORITHM	RANKING
nsga2	1.8
r-nsga2	1.6
mo-cma-es-p	3.5
mo-cma-es-p-rec	3.1

Table 15: First test case: average rankings of the algorithms with respect to the additive epsilon indicator. Friedman statistic considering reduction performance (distributed according to chi-square with 3 degrees of freedom): 47.88. *P*-value computed by Friedman Test $\approx 3 \times 10^{-10}$. Iman and Davenport statistic considering reduction performance (distributed according to F-distribution with 3 and 87 degrees of freedom) $\approx$ 32.9658. *P*-value computed by Iman and Daveport Test: $\approx 3 \times 10^{-14}$.

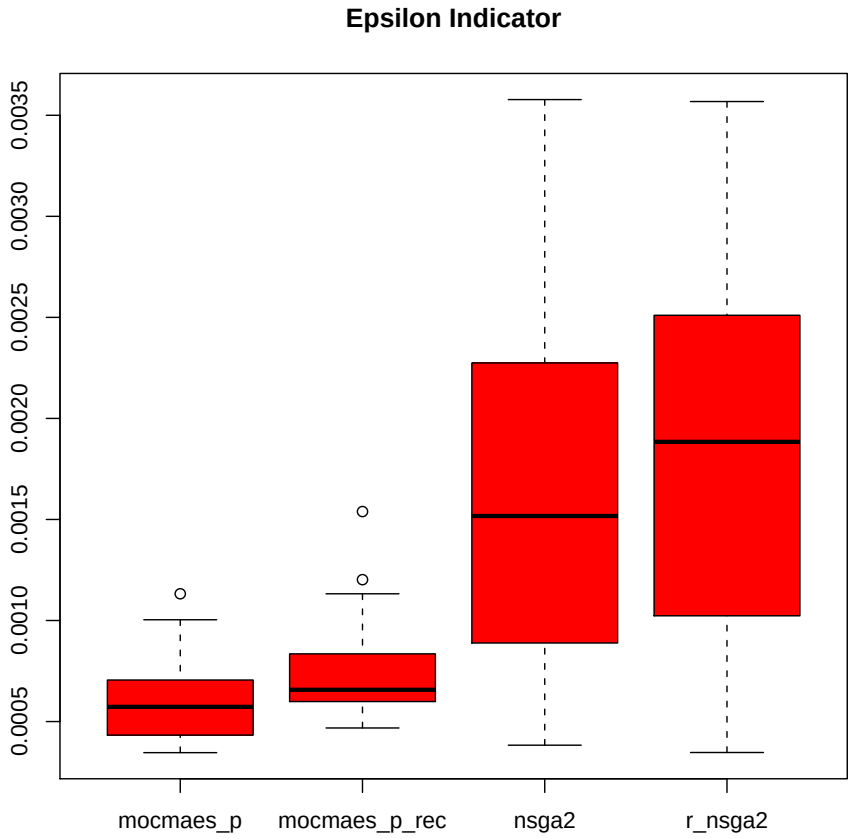


Figure 70: First test case: comparison of additive epsilon indicator values through boxplots.

computed by the Friedman test are extremely small, we can conclude that the optimizers are statistically significantly different with respect to the hypervolume indicator and, independently, the additive epsilon indicator.

Whether the Friedman test rejects the hypothesis that MOEAs have equal indicator distributions, we can proceed with a post-hoc test in order to find the concrete pairwise comparisons which produce differences. The following procedures, which are described in Section 3.7.4.4, are employed:

- Nemenyi's test. This is the most conservative procedure and many of the obvious differences may not be detected.
- Holm's test.
- Shaffer's static test.
- Bergmann-Hommel's dynamic test. This is the most powerful procedure, because it may detect small differences among the optimizers. However, it is also the most difficult to understand and computationally expensive.

The adjusted p-values for pairwise comparisons of the optimizers over the hypervolume and the epsilon indicators are shown in Table 16 and Table 17, respectively. Using an overall significance level of $\alpha = 0.001$ for the post-hoc test of each indicator, the four procedures that we mentioned above reject these hypotheses:

- nsga2 vs. mo-cma-es-p;
- nsga2 vs. mo-cma-es-p-rec;
- r-nsga2 vs. mo-cma-es-p;
- r-nsga2 vs. mo-cma-es-p-rec.

We can conclude that there is enough statistical evidence to support the fact that MO-CMA-ES-P and MO-CMA-ES-P-REC are preferable to NSGA-II and R-NSGA-II under the hypervolume indicator and, independently, under the epsilon indicator.

We rely on the open source software supplied by García and Herrera [40] for the Friedman test and the corresponding post-hoc tests.

Finally, we are interested in measuring the convergence of solutions to the Pareto optimal front (see Figure 68). This study could allow us to understand if the generation number can be decreased while maintaining good quality solutions or it has to be increased. Of course, a change in this parameter will affect the total execution time of the optimizers. In order to measure convergence, we use the I_H^- indicator, i.e., we compute the difference in hypervolume between the reference set (recall that the reference here is the set of pooled nondominated vectors because the true Pareto front is not available) and the current Pareto frontier (approximation set) at each generation and for each MOEA. Then, for each MOEA, we collect hypervolume values of corresponding generations and plot them through boxplots (see Figure 71 and Figure 72). In these figures, smaller values correspond to approximation set which are closer to the reference set, as usual.

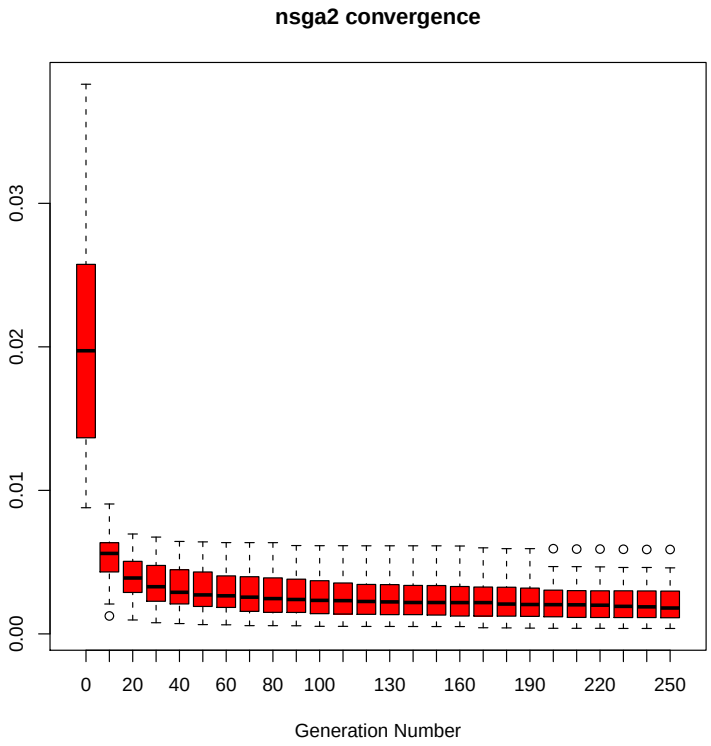
Obviously, a final choice has to be made among Pareto solutions. Figure 73 and Figure 74 show some Pareto points computed by the MO-CMA-ES-P, which

i	HYPOTHESIS	UNADJUSTED p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	nsga2 vs .mo-cma-es-p	4×10^{-11}	2×10^{-10}	2×10^{-10}	2×10^{-10}	2×10^{-10}
2	nsga2 vs .mo-cma-es-p-rec	1×10^{-9}	6×10^{-9}	5×10^{-9}	3×10^{-9}	3×10^{-9}
3	r-nsga2 vs .mo-cma-es-p	1×10^{-8}	7×10^{-8}	5×10^{-8}	4×10^{-8}	4×10^{-8}
4	r-nsga2 vs .mo-cma-es-p-rec	2×10^{-7}	1×10^{-6}	6×10^{-7}	6×10^{-7}	2×10^{-7}
5	nsga2 vs .r-nsga2	0.3681	2.2087	0.7362	0.7362	0.7362
6	mo-cma-es-p vs .mo-cma-es-p-rec	0.6171	3.7025	0.7362	0.7362	0.7362

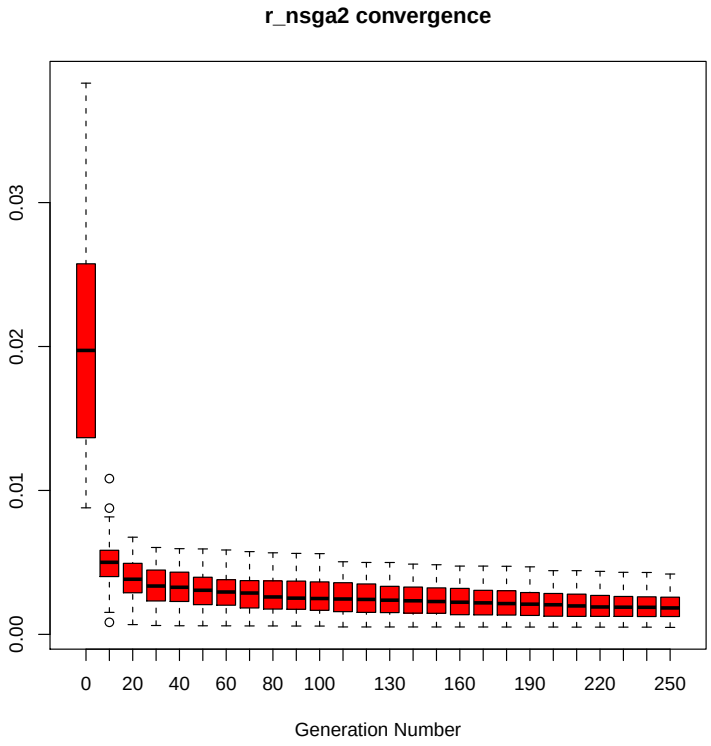
Table 16: First test case: adjusted p-values for the post-hoc test of the hypervolume indicator.

i	HYPOTHESIS	UNADJUSTED p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	r-nsga2 vs .mo-cma-es-p	1×10^{-8}	7×10^{-8}	7×10^{-8}	7×10^{-8}	7×10^{-8}
2	nsga2 vs .mo-cma-es-p	3×10^{-7}	2×10^{-6}	2×10^{-6}	1×10^{-6}	1×10^{-6}
3	r-nsga2 vs .mo-cma-es-p-rec	7×10^{-6}	4×10^{-5}	3×10^{-5}	2×10^{-5}	2×10^{-5}
4	nsga2 vs .mo-cma-es-p-rec	1×10^{-4}	6×10^{-4}	3×10^{-4}	3×10^{-4}	1×10^{-4}
5	mo-cma-es-p vs .mo-cma-es-p-rec	0.2301	1.3808	0.4603	0.4603	0.4603
6	nsga2 vs .r-nsga2	0.5485	3.2910	0.5485	0.5485	0.5485

Table 17: First test case: adjusted p-values for the post-hoc test of the additive epsilon indicator.

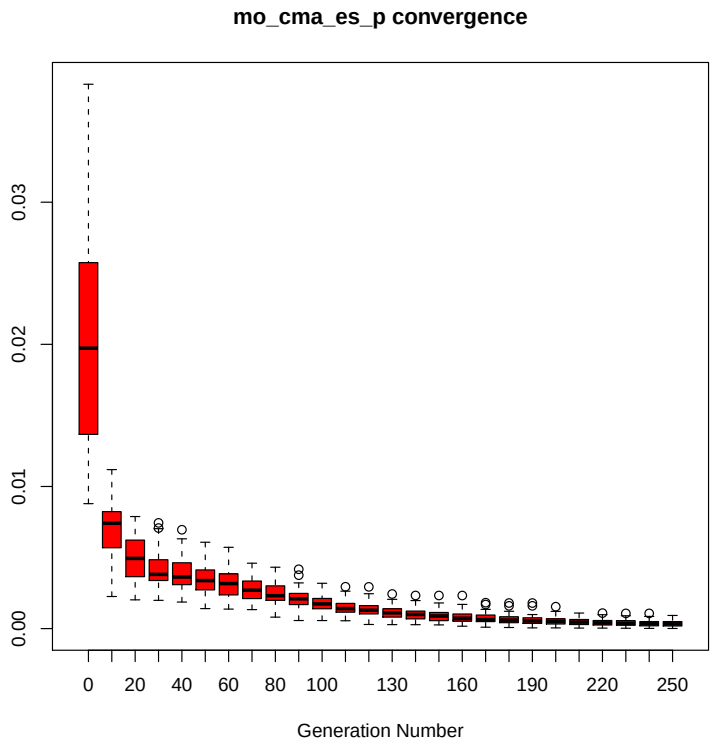


(a) Convergence of NSGA-II.

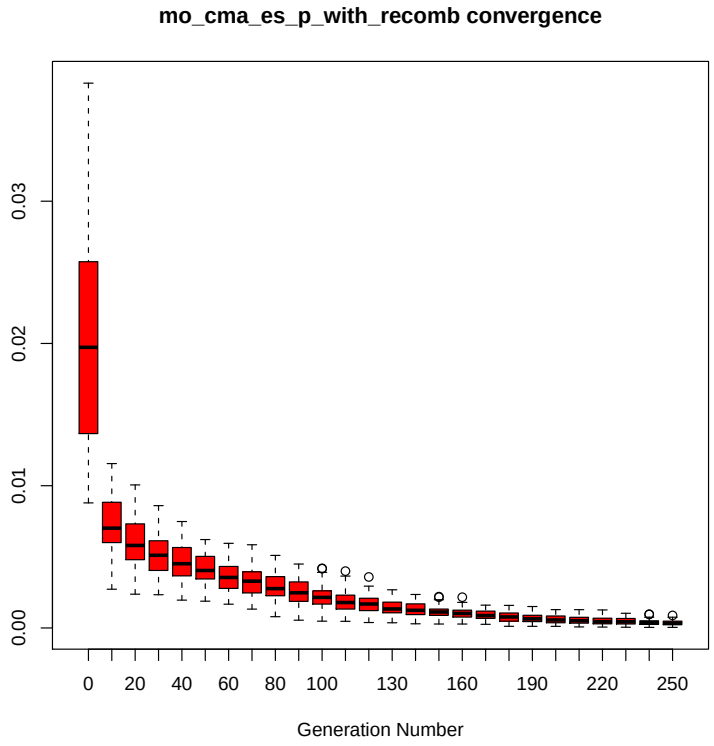


(b) Convergence of R-NSGA-II.

Figure 71: First test case: convergence study for the NSGA-II and the R-NSGA-II. For these two optimizers, the hypervolume distribution does not noticeably improve after 200 and 220 generation, respectively.

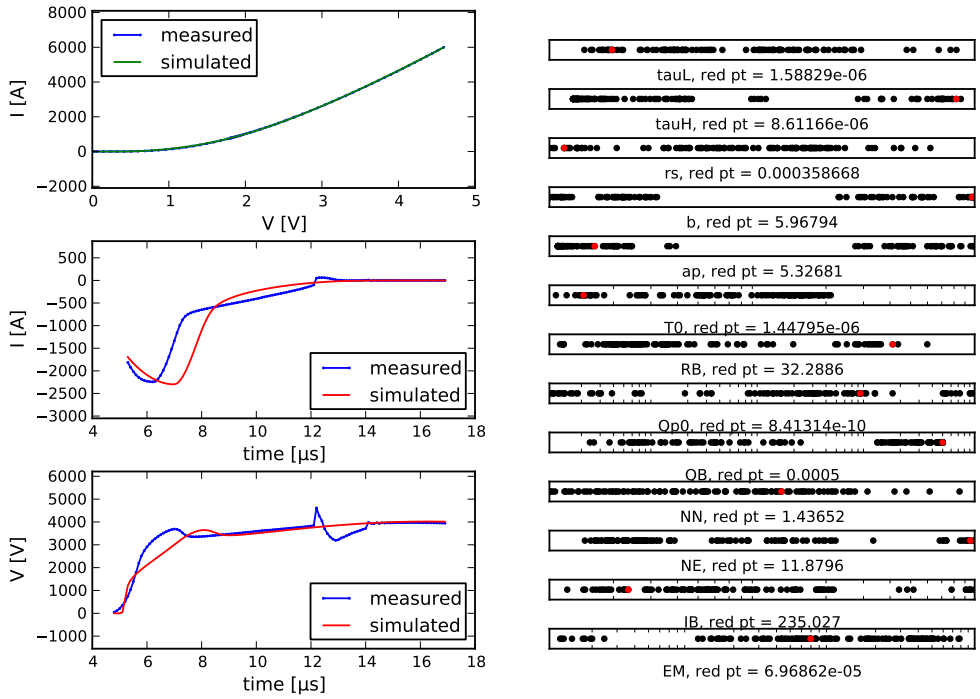


(a) Convergence of MO-CMA-ES-P.

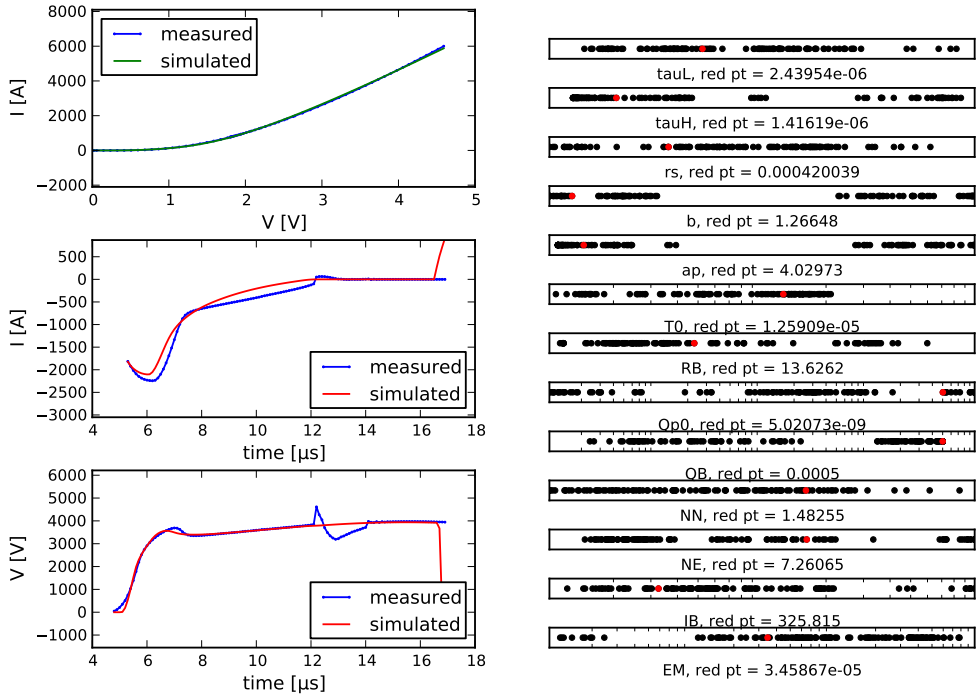


(b) Convergence of MO-CMA-ES-P-REC.

Figure 72: First test case: convergence study for the MO-CMA-ES-P and the MO-CMA-ES-P-REC. For these two optimizers, the hypervolume distribution does not noticeably improve after 220 and 240 generations, respectively.

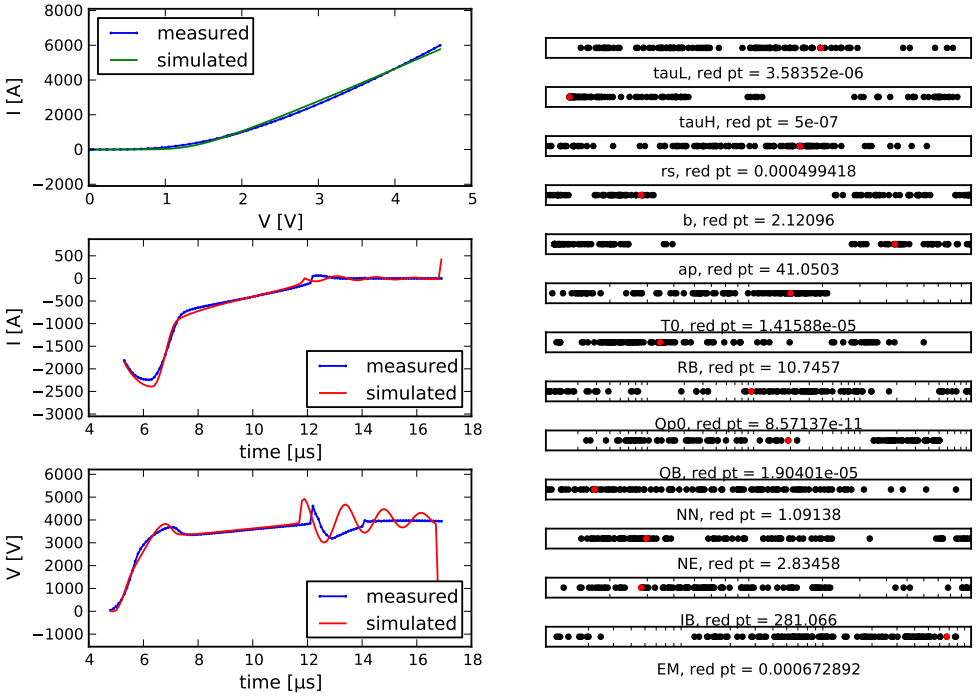


(a)

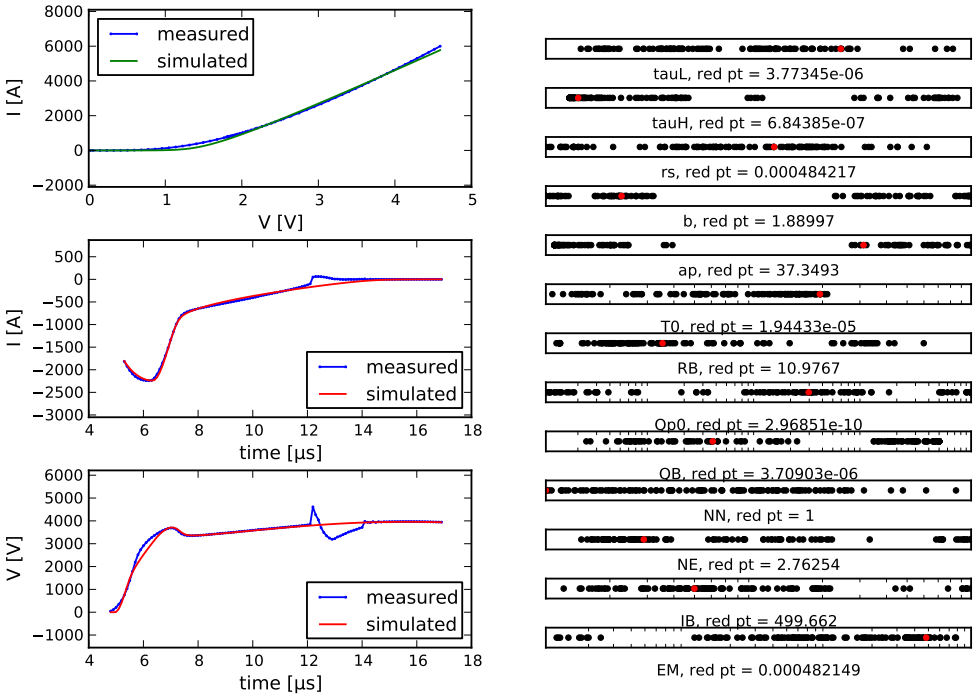


(b)

Figure 73: First test case: two Pareto points found by the MO-CMA-ES-P. The spike in (b) was not penalized because the error was calculated up to 12 microseconds since Extended Lauritzen model does not model the step variation of the current at 12 microseconds.



(a)



(b)

Figure 74: First test case: another two Pareto points found by the MO-CMA-ES-P. The oscillating behavior in (a) was not penalized because the error was calculated up to 12 microseconds since Extended Lauritzen model does not model the step variation of the current at 12 microseconds.

Correlation matrix		
1.0	−0.101 28	−0.202 23
−0.101 28	1.0	0.513 05
−0.202 23	0.513 05	1.0
Eigenvalues of the squared correlation matrix		
1.268 12	0.996 89	0.734 99
Proportion of variance		
0.422 71	0.332 30	0.244 99
Cumulative proportion of variance		
0.422 71	0.755 01	1.0
Eigenvectors of the squared correlation matrix		
−0.134 13	−0.987 55	−0.082 24
−0.696 84	0.153 00	−0.700 72
−0.704 57	0.036 68	0.708 68

Table 18: A principal component analysis of the reference set for the first test case.

is identified as one of the best performing algorithm by the post-hoc procedure. A decision-maker could be interested in selecting the Pareto point depicted in Figure 74b, since it represents a good trade-off between all objectives.

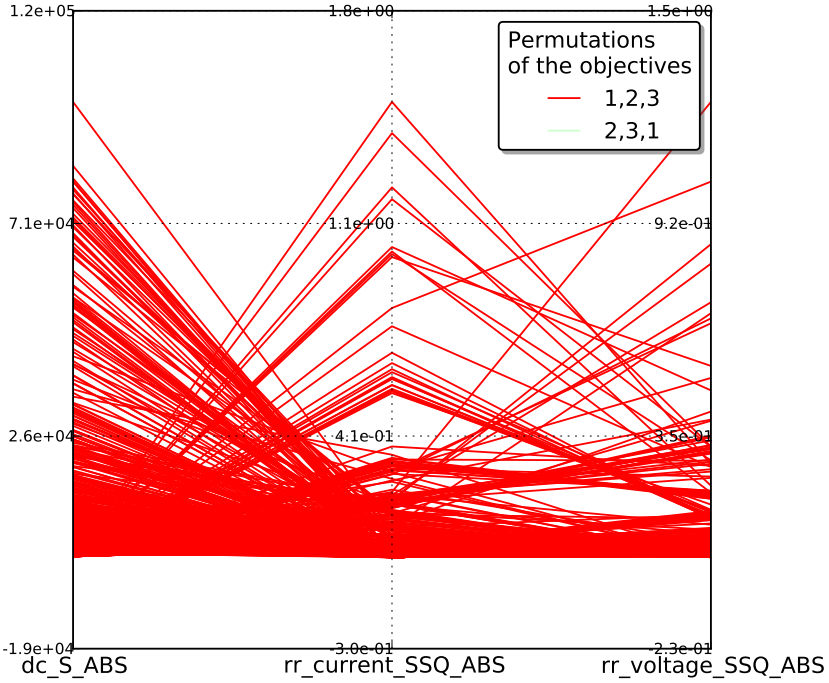
The whole reference set is represented in Figure 75. Crossing lines are evident between DC and reverse recovery current, while reverse recovery current and reverse recovery voltage seem to be in relative harmony with one another. This suggests that the Pareto optimal front PF_{true} could be less than 3-dimensional, and hence one objective between the reverse current RSS and the reverse voltage RSS could be redundant (see Section 3.6).

A PCA (see Algorithm 3.11) is performed on the reference set used for the unary quality indicators in order to identify redundant objectives. Results are summarized in Table 18. No redundant objective is identified by the PCA. Nevertheless, in the second test case, the reverse voltage RSS is removed from further consideration since it appears in Figure 75 that it benefits from improvements in reverse current.

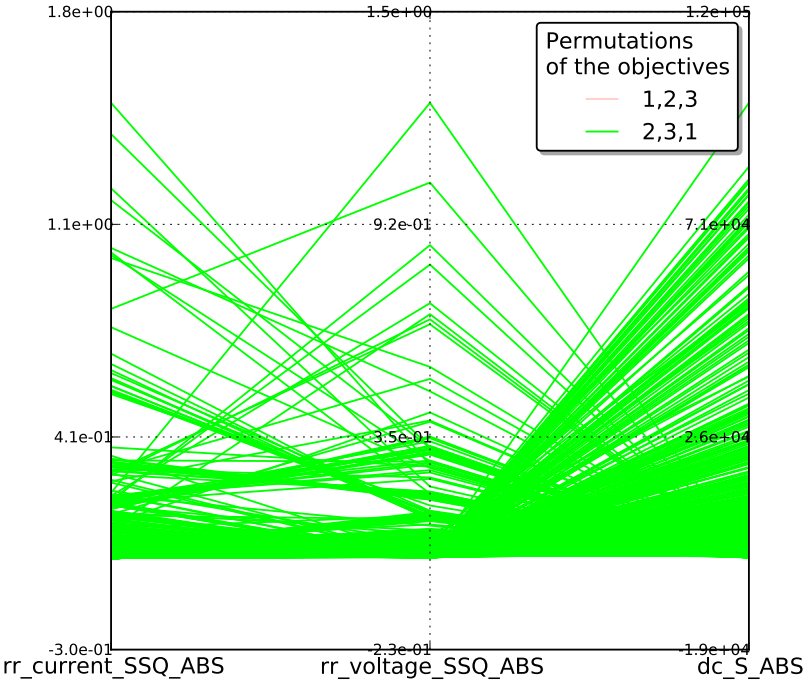
4.4 SECOND TEST CASE

In the second MOEA experiment, the reverse voltage RSS is dropped and the simultaneous optimization of the following two objectives is carried out:

- RSM between measured and simulated dc data;
- RSS between measured and simulated reverse recovery current.



(a) First permutation of objectives.



(b) Second permutation of objectives.

Figure 75: First test case: parallel coordinate plot of an approximation set computed by MO-CMA-ES-P. *dc_S_ABS*, *rr_current_SSQ_ABS* and *rr_voltage_SSQ_ABS* stand for DC RSM, reverse recovery current RSS and reverse recovery voltage RSS, respectively. Only two permutations of the objectives are needed to show every possible adjacency.

4.4.1 *Choosing MOEA parameters and quality measures*

In order to set MOEA strategy parameters for this problem, the following choices were made, with respect to the first test case:

- For each optimizer, a smaller population size of 100 was used, instead of 120. This is because the less the number of objectives the less the number of solutions required for approximating the entire Pareto front (see Section 3.6). In the two variants of the MO-CMA-ES, the number of offspring per parent was always 1.
- The generation number was not changed.
- For NSGA-II and R-NSGA-II, the same mutation probability, crossover probability, mutation distribution index and crossover distribution index were used.
- For the R-NSGA-II, the niching parameter was not modified. Three reference points were specified also in this case (objectives are considered in the following order: RSM-dc, RSS-rr-current):
  1. $(50, 1 \times 10^{-4})$: good solutions with respect to both the objectives are preferred;
  2. $(50, 2)$: solutions achieving a good fitting of dc i–v, but a bad fitting of reverse current are preferred;
  3. $(1 \times 10^{4}, 1 \times 10^{-4})$: solutions achieving a good fitting of reverse current but a bad fitting of dc i–v are preferred.
- In the two variants of the MO-CMA-ES, the strategy parameters of each individual implementing the $(1 + \lambda)$-CMA-ES were initialized as in the first problem.
- The hypervolume difference to a reference set I_H^- and the unary additive epsilon indicator $I_{\varepsilon+}^1$ were again selected as quality indicators. The reference set was generated by pooling all approximation sets and then extracting all nondominated objective vectors.

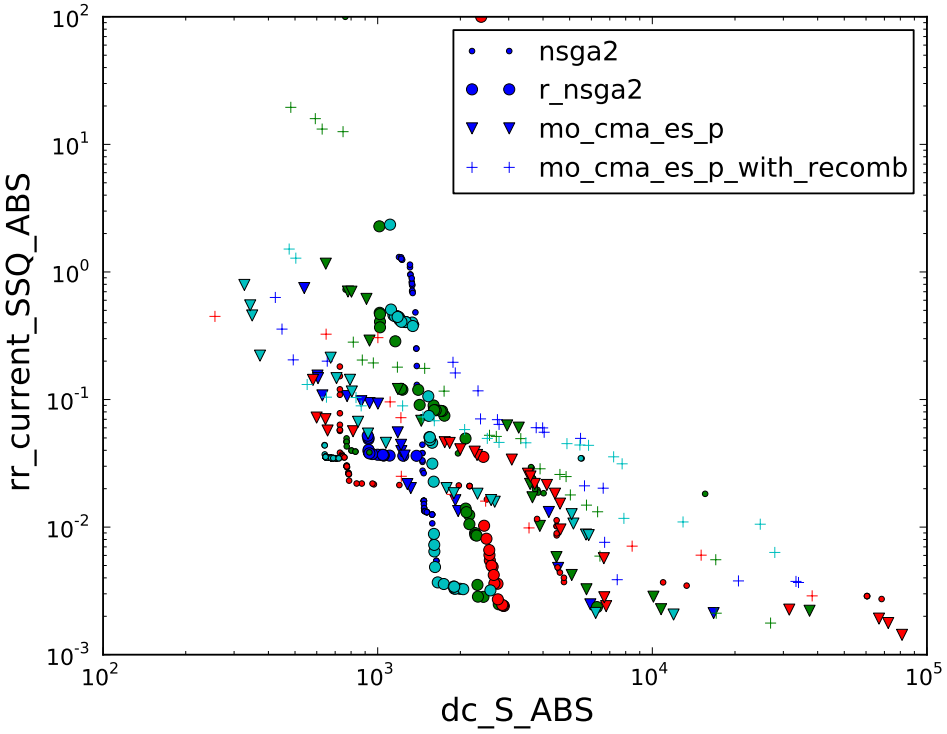
4.4.2 *Design of the experiment*

This experiment was set up by using the same design procedure followed for the first test case (see Section 4.3.2). User-provided seeds are listed in Table 19. For each indicator-MOEA pair we performed 30 MOEA runs, as before.

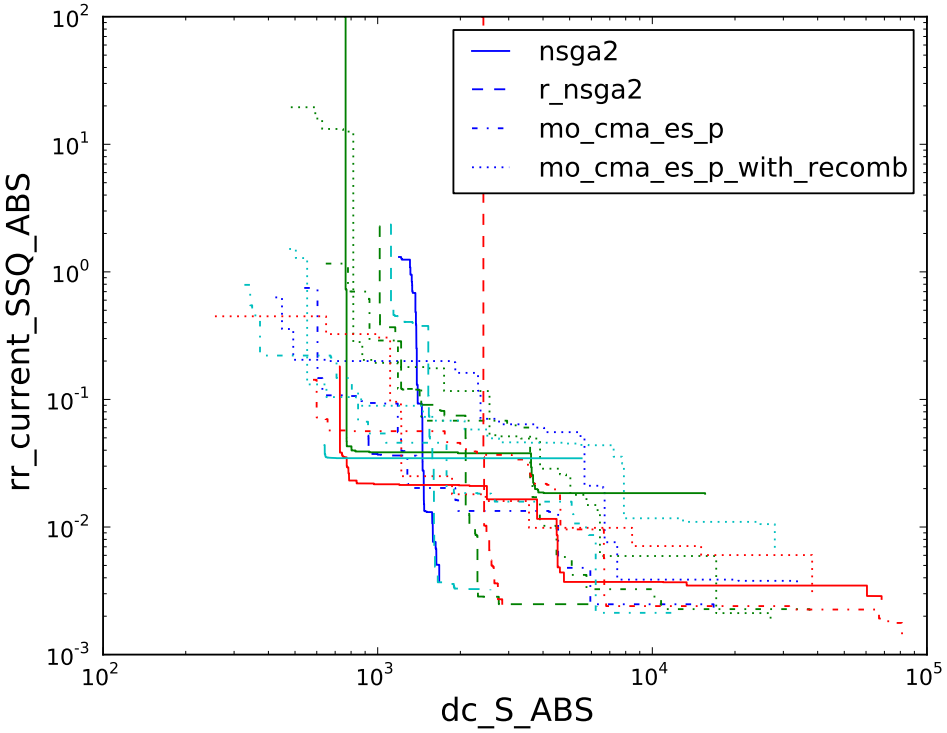
4.4.3 *Analysis of the results*

In Figure 76 the first four runs of NSGA-II, R-NSGA-II, MO-CMA-ES-P and MO-CMA-ES-P-REC are shown. The visual comparison is difficult and no conclusion can be drawn about MOEA performance from this figure.

Since only two objectives are considered in the current problem, we can use empirical attainment functions to provide valuable and complementary information



(a)



(b)

Figure 76: Second test case: representation of the first four approximation sets generated by NSGA-II, R-NSGA-II, MO-CMA-ES-P and MO-CMA-ES-P-REC (Figure (a)), and their corresponding attainment surfaces (Figure (b)).

DESIGN STEP	SEED
Generation of initial populations	1 540 651 080
Generation of MOEA starting seeds	84 161 851 033
Shuffling of initial populations	88 410.32
Shuffling of MOEA starting seeds	3 874 011 982
Shuffling of quality indicators	798 012 133
Shuffling of MOEA runs	47 560 487 564 397

Table 19: Second test case: seeds provided to the experimental design procedure.

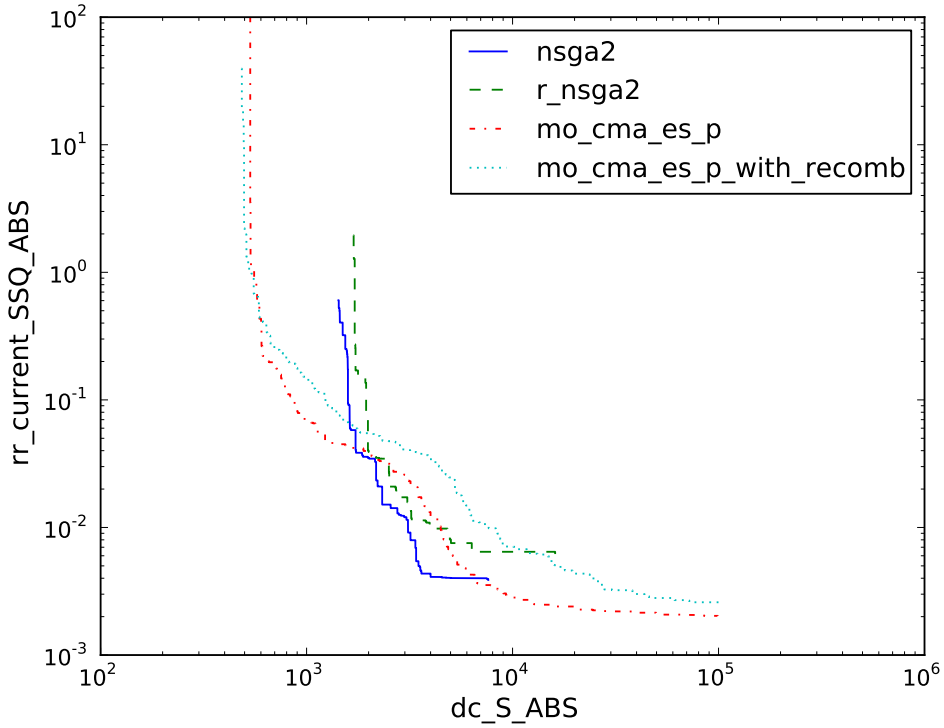


Figure 77: Second test case: 50%-attainment surfaces of the optimizers.

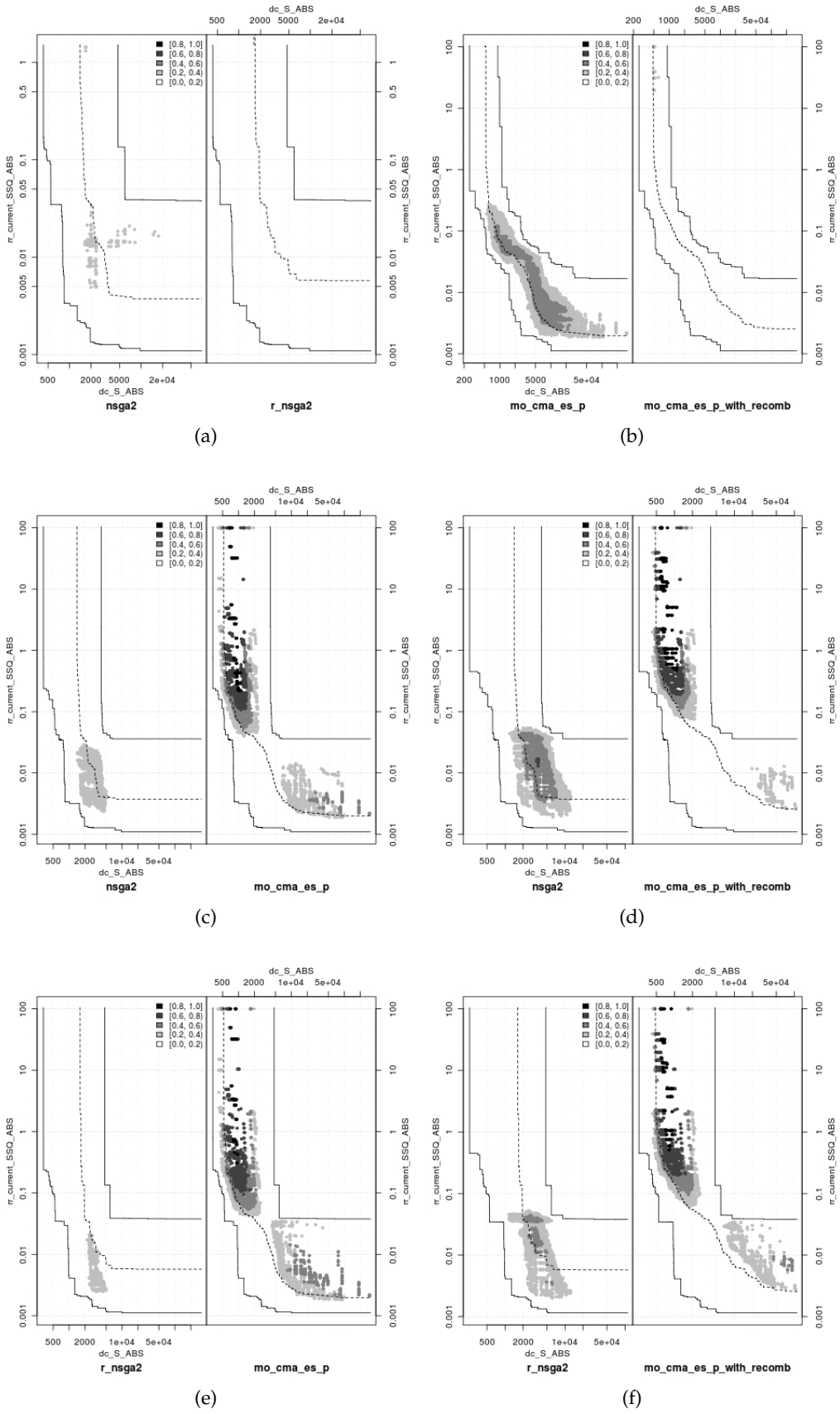


Figure 78: Second test case: individual differences between the probabilities of attaining different goals with NSGA-II, R-NSGA-II, MO-CMA-ES-P and MO-CMA-ES-P-REC.

ALGORITHM	RANKING
nsga2	1.6
r-nsga2	1.4
mo-cma-es-p	3.5
mo-cma-es-p-rec	3.5

Table 20: Second test case: average rankings of the algorithms with respect to the hypervolume indicator. Friedman statistic considering reduction performance (distributed according to chi-square with 3 degrees of freedom): 72.36. P-value computed by Friedman Test $\approx 7 \times 10^{-11}$. Iman and Davenport statistic considering reduction performance (distributed according to F-distribution with 3 and 87 degrees of freedom) ≈ 118.9592. P-value computed by Iman and Daveport Test $\approx 1 \times 10^{-30}$.

to the use of quality indicators. Figure 77 visualizes the 50%-attainment surface of the optimizers. One could guess that MO-CMA-ES-P and MO-CMA-ES-P-REC can better capture the whole spectrum of the Pareto front, while NSGA-II and R-NSGA-II focus on exploring the central region of the front. This behavior of NSGA-II is probably due to an error in our implementation of the algorithm which has been discussed in Section 4.2. From Figure 77, we can also argue that solutions generated by NSGA-II and MO-CMA-ES-P tend to dominate solutions generated by R-NSGA-II and MO-CMA-ES-P-REC, respectively. In Figure 78, differences in the empirical attainment functions of the four optimizers are plotted. We can make the following observations:

- both NSGA-II and R-NSGA-II converge better than the two variants of MO-CMA-ES in the center of the Pareto front (left-side plots of Figure 78c–(f)), while the latter optimizers converge more at the extremes (right-side plots of Figure 78c–(f));
- NSGA-II and R-NSGA-II do not exhibit big performance differences (Figure 78a);
- MO-CMA-ES-P converge better than MO-CMA-ES-P-REC in the center of the Pareto front (Figure 78b).

As a next step, the unary hypervolume indicator I_H^- and the unary additive epsilon indicator $I_{\varepsilon+}^1$ are applied using the normalized approximation sets and the reference set. Indicator distributions are depicted in Figure 79 and in Figure 80. These boxplots suggest that there are performance differences between the two variants of the MO-CMA-ES and the other two optimizers under the hypervolume indicator and, independently, under the epsilon indicator. From the epsilon indicator, one could also infer that MO-CMA-ES-P slightly outperforms MO-CMA-ES-P-REC. This is also confirmed by a statistical comparison based on the Friedman test and the corresponding post-hoc procedure.

Results of Friedman test for the comparisons of the optimizers over the hypervolume and the epsilon indicators are shown in Table 20 and in Table 21, respectively. Since the p-values computed by Friedman test are extremely small, we can con-

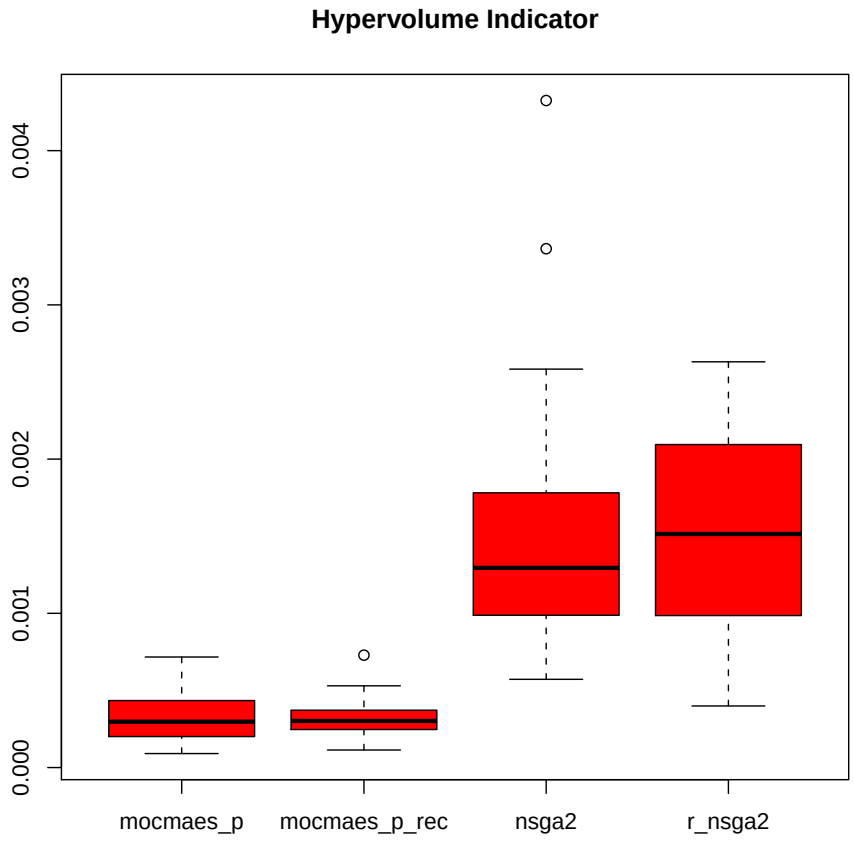


Figure 79: Second test case: comparison of hypervolume indicator values through boxplots.

ALGORITHM	RANKING
nsga2	1.8
r-nsga2	$1.7\overline{6}$
mo-cma-es-p	$3.\overline{6}$
mo-cma-es-p-rec	$2.7\overline{6}$

Table 21: Second test case: average rankings of the algorithms with respect to the additive epsilon indicator. Friedman statistic considering reduction performance (distributed according to chi-square with 3 degrees of freedom): 44.28. *P*-value computed by Friedman Test $\approx 1 \times 10^{-9}$. Iman and Davenport statistic considering reduction performance (distributed according to F-distribution with 3 and 87 degrees of freedom) $\approx$ 28.0866. *P*-value computed by Iman and Daveport Test: $\approx 9 \times 10^{-13}$.

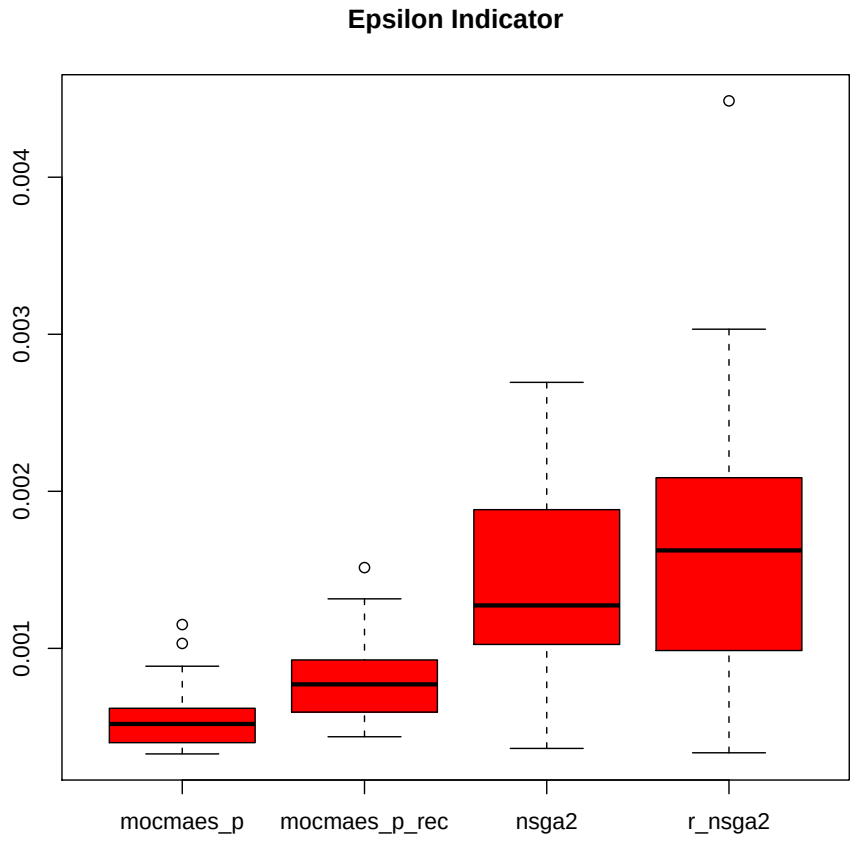


Figure 80: Second test case: comparison of additive epsilon indicator values through boxplots.

clude that the optimizers are statistically significantly different with respect to the hypervolume indicator and, independently, the additive epsilon indicator.

In order to find the concrete pairwise comparisons which produce differences, the following procedures are used (see Section 3.7.4.4):

- Nemenyi's test;
- Holm's test;
- Shaffer's static test;
- Bergmann-Hommel's dynamic test.

The adjusted p-values for pairwise comparisons of the optimizers over the hypervolume and the epsilon indicators are shown in Table 22 and Table 23, respectively. Under the hypervolume indicator, there is strong statistical evidence against the following hypotheses (see Table 22):

- nsga2 vs. mo-cma-es-p;
- nsga2 vs. mo-cma-es-p-rec;
- r-nsga2 vs. mo-cma-es-p;
- r-nsga2 vs. mo-cma-es-p-rec.

Under the epsilon indicator, only the following algorithms are statistically significantly different according to every post-hoc procedure:

- nsga2 vs. mo-cma-es-p;
- r-nsga2 vs. mo-cma-es-p.

Using an overall significance level of $\alpha = 0.05$, there is enough statistical evidence to reject the following additional hypotheses (see Table 23):

- nsga2 vs. mo-cma-es-p-with-recomb;
- r-nsga2 vs. mo-cma-es-p-with-recomb;
- mo-cma-es-p vs. mo-cma-es-p-with-recomb.

Thus, the hypervolume and the epsilon indicators do not agree on the statistical significance of performance differences between the optimizers.

In conclusion, there is enough statistical evidence to argue that MO-CMA-ES-P performs better than both NSGA-II and R-NSGA-II under the hypervolume indicator and, independently, under the epsilon indicator. Additionally, these two quality indicators may contradict one another on the performance ordering of NSGA-II, R-NSGA-II and MO-CMA-ES-P-REC. This then implies that the approximation sets generated by these three algorithms may be incomparable (see Section 3.7).

MOEA convergence to the reference set is depicted in Figure 81 and in Figure 82.

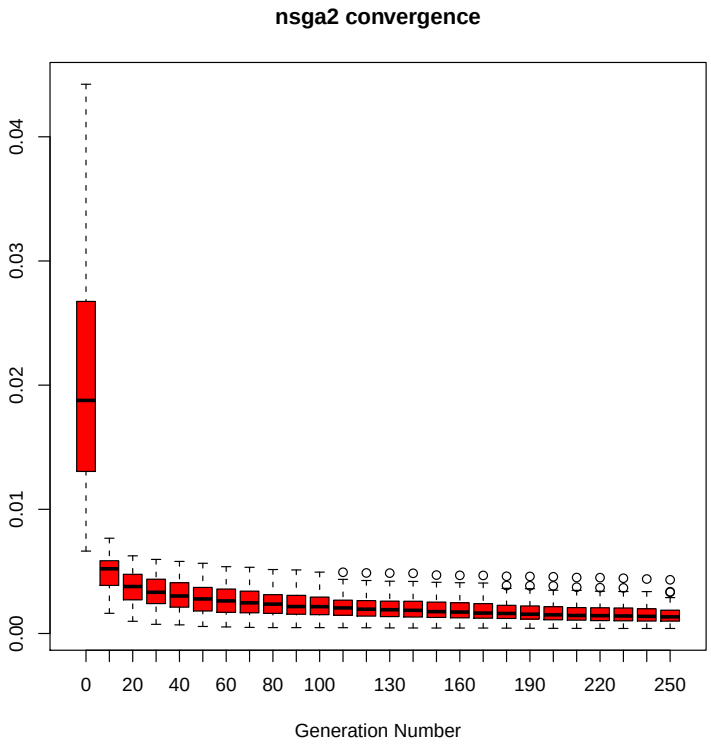
A final choice has to be made among Pareto solutions. Figure 83 and Figure 84 show some Pareto points computed by the MO-CMA-ES-P, which is identified as a good performing algorithm by the post-hoc procedure. Reverse recovery voltage

i	HYPOTHESIS	UNADJUSTED p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	r-nsga2 vs .mo-cma-es-p	3×10^{-10}	2×10^{-9}	2×10^{-9}	2×10^{-9}	2×10^{-9}
2	r-nsga2 vs .mo-cma-es-p-with-recomb	3×10^{-10}	2×10^{-9}	2×10^{-9}	2×10^{-9}	2×10^{-9}
3	nsga2 vs .mo-cma-es-p	1×10^{-8}	7×10^{-8}	5×10^{-8}	4×10^{-8}	4×10^{-8}
4	nsga2 vs .mo-cma-es-p-with-recomb	1×10^{-8}	7×10^{-8}	5×10^{-8}	4×10^{-8}	4×10^{-8}
5	nsga2 vs .r-nsga2	0.5485	3.2910	1.0970	1.0970	1.0970
6	mo-cma-es-p vs .mo-cma-es-p-with-recomb	1.0	6.0	1.0970	1.0970	1.0970

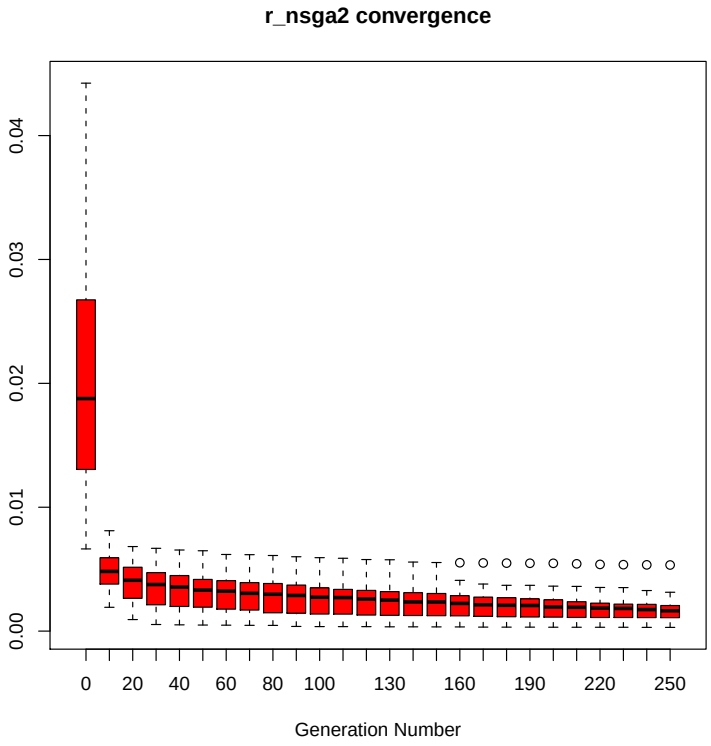
Table 22: Second test case: adjusted p-values for the post-hoc test of the hypervolume indicator.

i	HYPOTHESIS	UNADJUSTED p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	r-nsga2 vs .mo-cma-es-p	1×10^{-8}	7×10^{-8}	7×10^{-8}	7×10^{-8}	7×10^{-8}
2	nsga2 vs .mo-cma-es-p	2×10^{-8}	1×10^{-7}	1×10^{-7}	7×10^{-8}	7×10^{-8}
3	r-nsga2 vs .mo-cma-es-p-with-recomb	0.0027	0.0162	0.0108	0.0081	0.0081
4	nsga2 vs .mo-cma-es-p-with-recomb	0.0037	0.0224	0.0112	0.0112	0.0081
5	mo-cma-es-p vs .mo-cma-es-p-with-recomb	0.0069	0.0416	0.0139	0.0139	0.0139
6	nsga2 vs .r-nsga2	0.9203	5.5221	0.9203	0.9203	0.9203

Table 23: Second test case: adjusted p-values for the post-hoc test of the additive epsilon indicator.

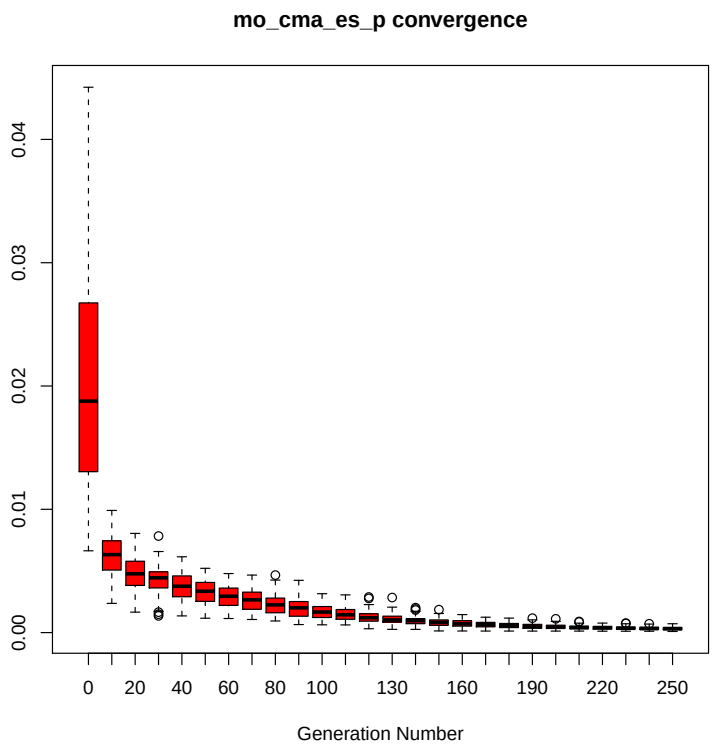


(a) Convergence of NSGA-II.

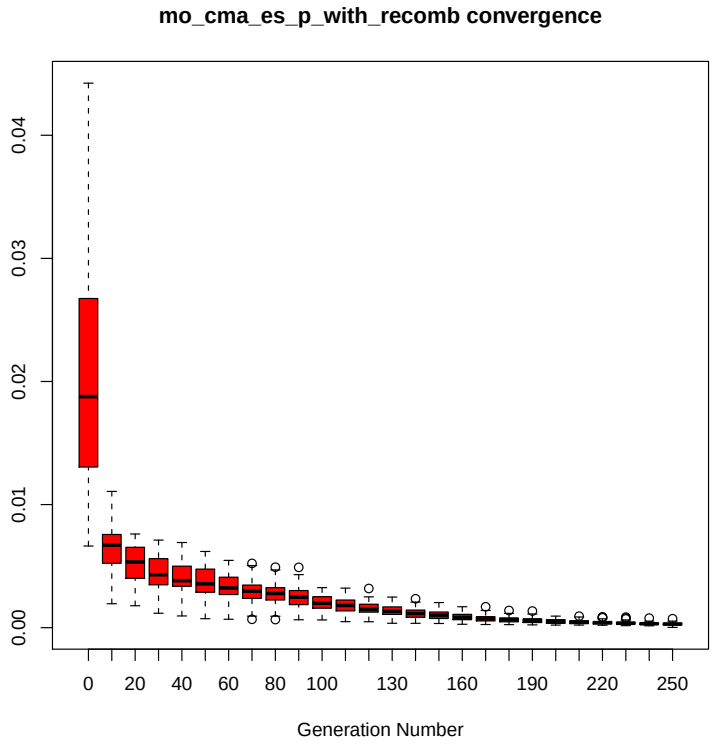


(b) Convergence of R-NSGA-II.

Figure 81: Second test case: convergence study for the NSGA-II and the R-NSGA-II. For these two optimizers, the hypervolume distribution does not noticeably improve after 180 and 200 generation, respectively.

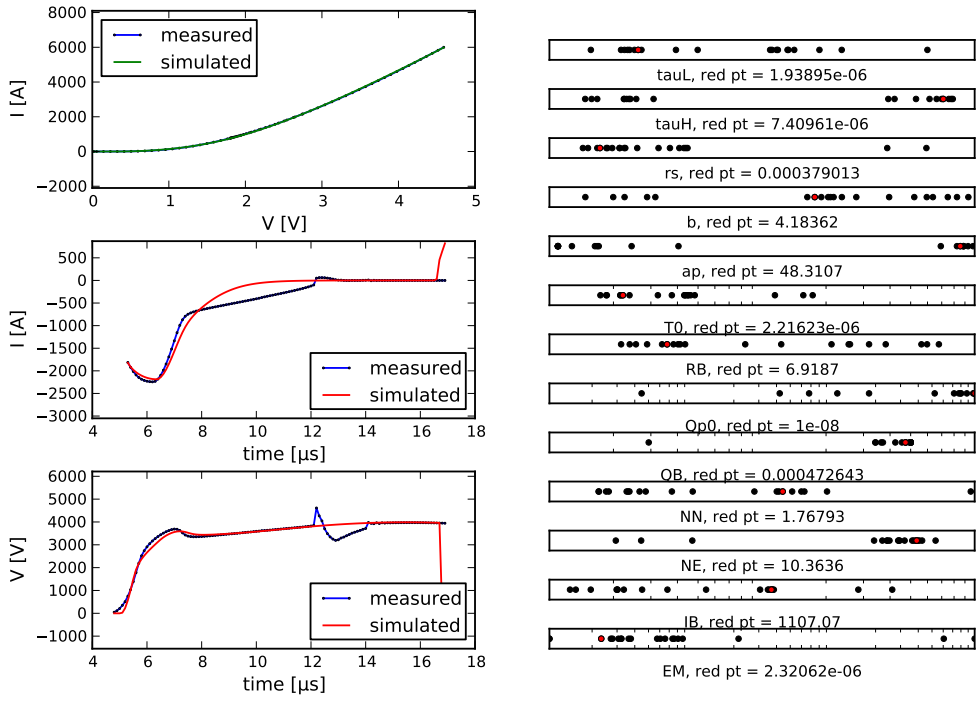


(a) Convergence of MO-CMA-ES-P.

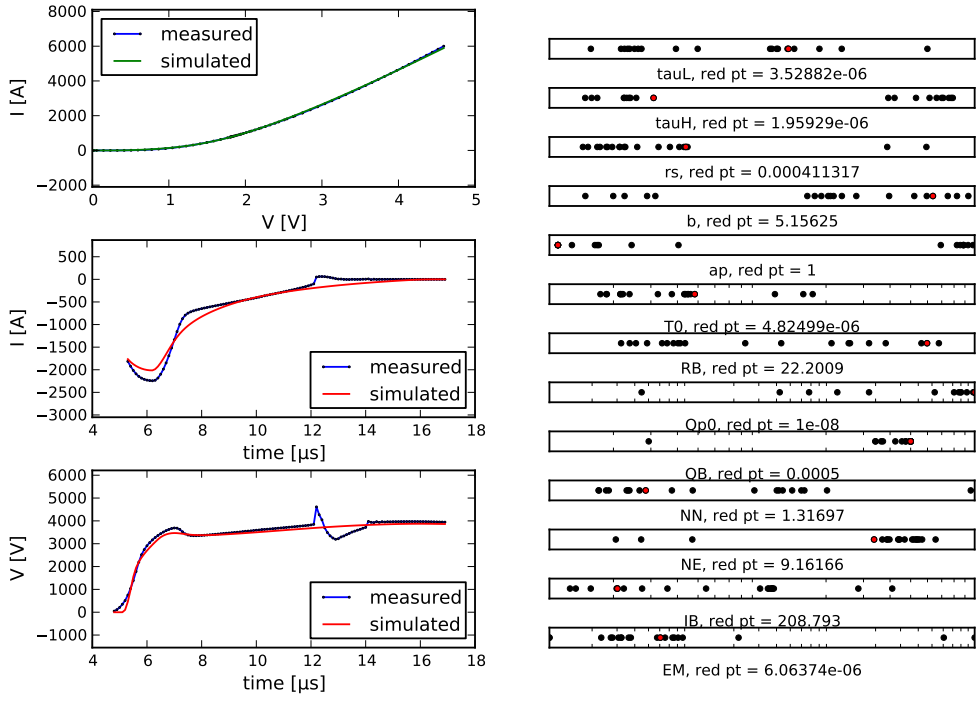


(b) Convergence of MO-CMA-ES-P-REC.

Figure 82: Second test case: convergence study for the MO-CMA-ES-P and the MO-CMA-ES-P-REC. For both the optimizers, the hypervolume distribution does not noticeably improve after 200.

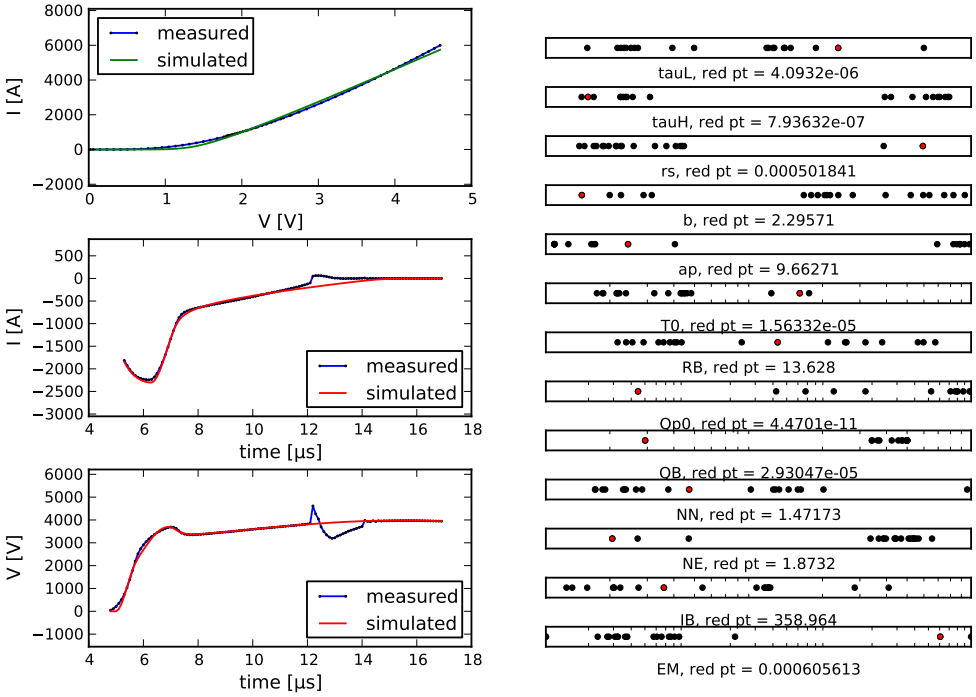


(a)

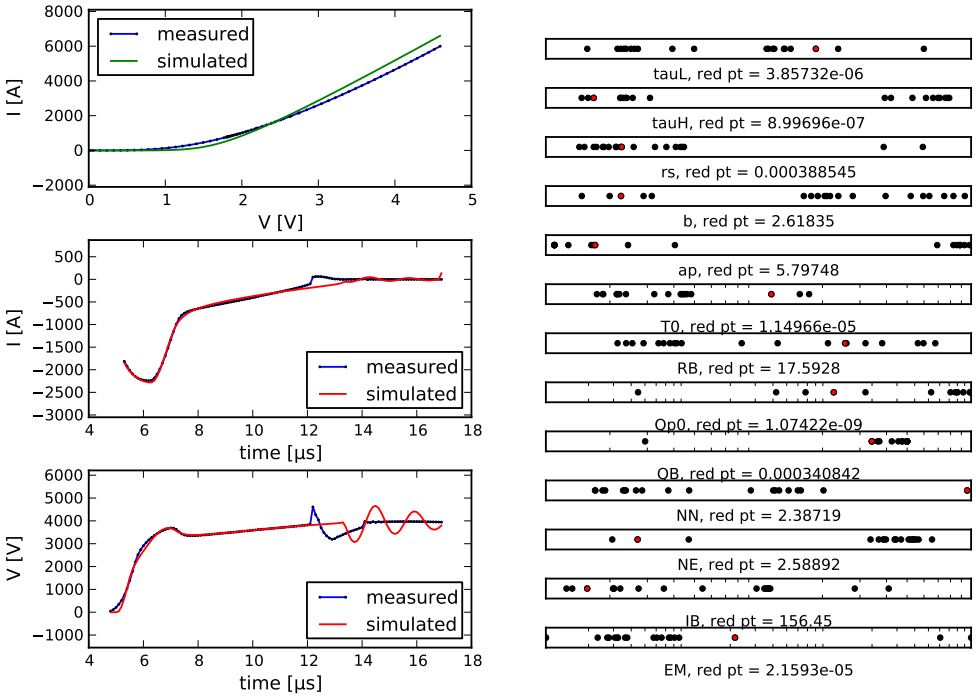


(b)

Figure 83: Second test case: two Pareto points found by the MO-CMA-ES-P. In this MOEA experiment, only the dc and reverse recovery current characteristics were optimized. However, in order to provide additional information, reverse recovery voltage is also shown. The spike in (a) was not penalized because the error was calculated up to 12 microseconds since Extended Lauritzen model does not model the step variation of the current at 12 microseconds.



(a)



(b)

Figure 84: Second test case: another two Pareto points found by the MO-CMA-ES-P. In this MOEA experiment, only the dc and reverse recovery current characteristics were optimized. However, in order to provide additional information, reverse recovery voltage is also shown. The oscillating behavior in (b) was not penalized because the error was calculated up to 12 microseconds since Extended Lauritzen model does not model the step variation of the current at 12 microseconds.

is plotted to check if a good fitting of this characteristic is achieved even if it is not considered as an objective function.

A decision-maker could be interested in selecting the Pareto point depicted in Figure 84a, since it represents a good trade-off between all objectives. This provides further evidence of the physical coupling between current and voltage in the phenomena of turn-off and, hence, of the absence of conflict between these two characteristics near the Pareto optimal region.

5

CONCLUSIONS AND FUTURE WORK

In Chapter 4 two experiments were presented in order to investigate the efficiency and effectiveness of the automated parameter extraction procedure based on MOEAs which we described in Section 1.6.

Several MOEAs are available in the Python library for parameter extraction that we developed during our internship at ABB Corporate Research Center in Switzerland. In experiments we compared NSGA-II, R-NSGA-II and two variants of the MO-CMA-ES.

Empirical evaluation revealed that MOEAs are:

- Robust: they do not require computation of initial guesses of device model parameters. Then the quality of final solutions does not depend on such estimates. Moreover the user is not asked to tune optimizer's parameters.
- Effective: they can effectively solve parameter extraction problems with conflicting objectives and are not affected by objective space convexity or non-convexity. They also return several solutions in a single run, thus providing the decision-maker a true picture of trade-offs.
- Simple and high-impact: once physical ranges of model parameters have been defined, the user does not have to monitor MOEA progress nor to take corrective actions. In this sense, extracting model parameters with MOEAs is a completely automated procedure. Tremendous savings in time and engineering resources could then be achieved. A detailed understanding of neither the device compact model nor the optimization algorithm is requested to perform the extraction.
- Generalizable: since MOEAs are not model specific and they do not require experienced user guidance to succeed, it is easy to port them with confidence from one silicon technology to another.

The MO-CMA algorithm with population-based step size adaptation, named MO-CMA-ES-P, appeared to be the superior method. It significantly outperformed the NSGA-II and R-NSGA-II in both tests. Combining this step size adaptation procedure with recombination of strategy parameters proposed in [93] did not improve performance of the MO-CMA-ES, surprisingly. This could be essentially due to errors in our implementation of the recombination strategy and/or to a very rugged search landscape which prevents effective transfer of information between neighboring individuals.

At present, we suggest the use of the MO-CMA-ES-P to practicing engineers in the power electronics community. In this algorithm the multi-objective selection is based on the contributing hypervolume, whose computation is exponential in the number of objectives [97]. We do not regard this bad scaling behavior as a sever drawback, because in real-world applications the costs for generating offspring

and selection can often be neglected compared to the time needed for objective function evaluation.

Further work is needed in order to investigate whether MOEAs are a universal solution technique for all parameter extraction problems. Extraction of different device compact models should be performed and MOEAs should be "stress tested" by optimizing as many objectives as possible. At present we are working on a parameter extraction problem for Extended Lauritzen model with seven objectives. One could also try to optimize several different devices at the same time. However, in the latter case, the possibility of behavioral models being inadvertently created would increase [7].

A

SOFTWARE STRUCTURE

During our internship at ABB Corporate Research Center in Switzerland we developed a Python library to perform automated parameter extraction of lumped charge models. Implemented algorithms are general enough to be safely ported from one silicon technology to another. The software also provides a pool of post-processing routines to assess performance of multi-objective optimizers.

Here is a list of the software that has been developed:

- *YAML files*: text files which have been written in *YAML* format, defining sets of instructions to perform single objective optimizations, multi objective optimizations and sensitivity analyses.
- *plotting_prm.yaml*: a text file written in YAML sintax with a set of default instructions to plot simulated, measured and errors values. It must be in the same directory where `model_fitting.py` lies.
- `extr_print.py`: functions for reading simulation files generated by SIMetrix.
- `initial_parameter_estimation.py`: compute starting guesses for parameters of Lauritzen, Ma and Extended Lauritzen diode models.
- `model_fitting.py`: computation of fitting errors between measured and simulated device characteristics.
- `moments.py`: functions for calculating moments of a curve.
- `ngspice_extr_print.py`: functions for reading simulation files generated by ngspice.
- `opt_parser.py`: module which provides routines for single and multi objective optimizations and sensitivity analysis.
- `plotSimPrm.py`: this module defines a container class for plotting options of figures with simulated and measured data.
- `performance_assessment_of_moea.py`: module which provide routines for postprocessing of computational experiments with MOEAs.
- `RR.py`: module for aligning measured and simulated diode reverse recovery characteristics and for extracting diode characteristic values.
- `runSimulation.py`: routines for running SIMetrix and ngspice.
- *inspyredABB*: module extending the *inspyred* Python library for creating evolutionary computations in Python.
- *smeasure_module*: C++ library for computing the hypervolume indicator.

- *scripts*: a collection of scripts. The main scripts are:
  - `sensitivity_single_char.py`: sensitivity analysis for a single device characteristic. Not recommended.
  - `new_sensitivity_analysis.py`: sensitivity analysis for all characteristics of a device. Recommended.
  - `optimize_single_char_with_simplex.py`: single objective optimization with the simplex algorithm. Not recommended.
  - `optimize_single_char_with_cma.py`: optimization of a single characteristic with the (μ, λ)-CMA-ES. Recommended.
  - `optimization_loop_with_cma.py`: parameter extraction performed as a succession of single objective optimizations carried out with the CMA-ES. Not recommended.
  - `process_optimization_loop_results.py`: postprocessing of the results of the previous script.
  - `multi_objective_optimization.py`: optimization of several characteristics through a MOEA. Recommended.
  - `run_moea_tests.py`: designing and running computational experiments with MOEAs.
  - `moeas_performance_assessment.py`: analysis of the results of a MOEA comparison.
  - `single_moea_performance_assessment.py`: analysis of the results of a single MOEA run.

All routines were written in Python 2.6.5 and C++. In addition to a standard Python distribution, the following Python packages have been used:

- *PyYAML 3.10*: YAML implementation for Python. It can be downloaded from `http://pyyaml.org/wiki/PyYAML`.
- *NumPy 1.6.2*: array processing for numbers, strings, records, and objects.
- *SciPy 0.11.0*: scientific library for Python.
- *inspyred 1.0*: framework for creating evolutionary computations in Python, available at `http://pypi.python.org/pypi/inspyred/1.0`.
- *scikits.ann*: this Python module provides a numpy-compatible Python wrapper around the Approximate Nearest Neighbor library (`http://www.cs.umd.edu/~mount/ANN/`). It can be downloaded from `http://pypi.python.org/pypi/scikits.ann`.
- `cma.py`: module implementing the (μ, λ)-CMA-ES, available at `http://www.lri.fr/~hansen/cmaesintro.html`.

Other additional programs used in the Python library are:

- *sed*: a stream editor used to filter text. *sed* is available for both Windows (`http://gnuwin32.sourceforge.net/packages/sed.htm`) and GNU operating systems (`http://www.gnu.org/software/sed/sed.html`).

- *taskkill*, a console command available on Microsoft Windows XP, Vista and 7 to terminate a process tree.

First we will present some fundamental classes and functions which are used throughout the code. Then we will explain how to run single objective and multi-objective optimizations and how to perform sensitivity analysis.

A.1 BUILDING A VECTOR OF VARIABLE PARAMETERS

The model parameters to optimize are collected in an instance of the `VariablePrm` class.

The main attributes of this class are:

- `names`, list of names of variable model parameters.
- `values`, array of values of variable model parameters. These values are subject to transformations specified in the `transformation` attribute.
- `bounds`, list of lower and upper bounds of variable model parameters. These values are subject to transformations specified in the `transformation` attribute.
- `transformation`, list of transformations applied to variable parameter values and bounds. Currently supported transformations are: $\log_{10}$, normalization between 0 and 1, affine transformations.

This class has methods to

- transform variable parameter values and bounds;
- invert variable parameter transformations;
- set values and bounds;
- compute a penalty value to be added to the error between simulated and measured data when an optimizer returns a vector of variable model parameters which violates bounds.

A.2 COMPUTING ERRORS BETWEEN MEASURED AND SIMULATED DATA

All classes and functions described in this subsection are implemented in the `model_fitting.py` module.

A.2.1 *The curveFamily class*

The `curveFamily` class can store measured or simulated waveforms. Some operators and functions are provided: largest value, smallest value, average value, addition, subtraction, multiplication, division and power of `curveFamily` objects.

If measured and simulated waveforms are not defined at the same points, simulated data has to be interpolated over measured data points. Interpolation is performed by the `curveFamily.interpolate_sim` method. If data interpolation fails for any reason, it is reported to the calling function.

A.2.2 *The errorFamily class*

The `errorFamily` class calculates a pool of error estimators between two waveforms stored in two `curveFamily` objects. Waveforms must be defined at the same points. These two waveforms can correspond to measured and simulated data or to any pair of waveforms to be compared. Some of the implemented error estimators are listed below:

- sum of absolute errors

$$\text{S\_ABS} = \sum_{n=1}^{N} |y_{\text{interp\_sim}}(n) - y_{\text{meas}}(n)|;$$

- sum of squares of absolute errors

$$\text{SSQ\_ABS} = \sum_{n=1}^{N} (y_{\text{interp\_sim}}(n) - y_{\text{meas}}(n))^2;$$

- sum of squares of relative errors

$$\text{SSQ\_REL} = \sum_{n=1}^{N} \Big(\frac{y_{\text{interp\_sim}}(n) - y_{\text{meas}}(n)}{y_{\text{meas}}(n)} \Big)^2;$$

- variance estimator

$$\text{VAR\_EST} = \frac{1}{N-1} \sum_{n=1}^{N} (y_{\text{interp\_sim}}(n) - y_{\text{meas}}(n))^2;$$

- square root of the sum of absolute errors

$$\text{RMSE} = \sqrt{\text{SSQ\_ABS}};$$

- square root of the sum of relative errors

$$\text{RMSErel} = \sqrt{\text{SSQ\_REL}};$$

- normalized root mean square absolute error

$$\text{NRMSE} = \frac{\text{RMSE}}{y_{\text{meas,max}} - y_{\text{meas,min}}};$$

- largest absolute error

$$\max_{n=1,\ldots,N} |y_{\text{interp\_sim}}(n) - y_{\text{meas}}(n)|;$$

- largest relative error

$$\max_{n=1,\ldots,N} \Big| \frac{y_{\text{interp\_sim}}(n) - y_{\text{meas}}(n)}{y_{\text{meas}}(n)} \Big|,$$

where N is the number of samples. We recommend the use of SSQ_ABS to assign more significance to any large errors between measured and simulated waveforms and the use of S_ABS to redress the imbalance between errors which are high in magnitude and errors which are smaller in magnitudes. The use of other error estimators is deprecated.

A.2.3 *SSQ class hierarchy*

Usually measured and simulated data are processed in different ways for each diode or IGBT characteristic before computing errors. After processing data, methods of the `SSQ` class can be used to:

- compute error estimators: `SSQ.calc_error`;
- plot absolute and relative errors: `SSQ.plot_AbsErr` and `SSQ.plot_RelErr`, respectively;
- plot simulated and measured data: `SSQ.plot_sim_data`.

`SSQ.plot_AbsErr`, `SSQ.plot_RelErr` and `SSQ.plot_sim_data` methods use methods of `model_fitting.plotSim` class to plot figures.

The `plotSim` class reads the file *plotting_prm.yaml* which has a set of default instructions for data plotting written in YAML syntax. `plotSim` is a subclass of the `plotSimPrm` class which is defined in `plotSimPrm.py`. The `plotSimPrm` class is specially designed to read *plotting_prm.yaml*.

From the `SSQ` class several subclasses are derived, one for each diode or IGBT characteristic. Here is the complete list of the derived classes:

- `SSQ_ID_VG`: to be used with IGBT transfer characteristic simulations (gate-emitter voltage vs collector current).
- `SSQ_QG_VG`: designed for IGBT gate charge simulations.
- `SSQ_OC_ALL`: designed for IGBT "output" characteristics (collector-to-emitter voltage vs. collector-to-emitter current) simulations at several gate-to-emitter voltage values.
- `SSQ_OC_VG`: designed for IGBT output characteristics simulations at a single gate-to-emitter voltage value.
- `SSQ_ONSTATE`: designed for IGBT on-state simulations.
- `SSQ_EonEoff_metrics`: designed for IGBT turn-off and turn-on simulations.
- `SSQ_DiodeDC`: designed for diode dc i–v simulations.
- `SSQ_DiodeRR`: designed for diode reverse recovery current simulations. This class uses functions and classes defined in the `RR.py` module to automatically synchronize measured and simulated waveforms and to compute diode characteristic parameters. The module `moments.py` is used to compute current curve moments.
- `SSQ_DiodeRR_V`: designed for diode reverse recovery voltage simulations. It uses functions and classes from the `RR.py` module to automatically synchronize measured and simulated waveforms.
- `SSQ_DiodeCap`: designed for diode capacitance simulations.

Classes for diode simulations can be used for parameter extraction, while classes for IGBT simulations are not up to date.

The following section explains how simulation files generated by SIMetrix or ngspice are read.

A.3 READING SIMULATION FILES

Simulation files generated by SIMetrix are read by the `extr_print` function defined in the `extr_print.py` module. Whenever a simulation fails, this is reported in the simulation file through a warning or an error string. If `extr_print()` finds such a kind of string, it will raise an exception which will be catched by the `runSimulation` function in the `runSimulation.py` module.

A.4 RUNNING A SIMULATION

The `runSimulation` function in the `runSimulation.py` module is used to run a circuit simulation with SIMetrix or ngspice. Below we briefly explain how it works:

- It takes a realization of the vector of compact model parameters.
- It writes these values to a text file, named "sed" file.
- It takes the path to a "template" file describing the circuit simulation written in SIMetrix or ngspice syntax. In this file, some circuit, diode and/or IGBT parameters are undefined.
- It creates a circuit file ready to be used in SIMetrix or ngspice by running the program `sed` on the sed and template files.
- It runs SIMetrix or ngspice.
- It reads the simulation file produced by SIMetrix or ngspice.
- It calculates error estimators between simulated and measured data.
- It saves simulated and measured data used in error calculation to text files.
- It returns error estimators calculated before.

Whether a simulation or the error computation fails, an exception is raised. This exception is caught by `runSimulation`. The next section describes how exceptions are handled.

A.5 EXCEPTION HANDLING

Exceptions are used to report errors that may happen when running a simulation or computing fitting errors between measured and simulated data. Several exceptions have been identified in order to implement a different handling approach for each of them.

All the exceptions that have been identified are:

- Exceptions defined in `extr_print.py`:
  - `TempFileError`: in the simulation file created by SIMetrix an error message appears which says that SIMetrix could not open temporary files for writing because another instance of SIMetrix was writing to the same

files. The corresponding circuit file is saved to a directory for failed simulations. This may happen when several SIMetrix instances are started in an interleaved way during a multiobjective optimization. If this exception is thrown several times for the same parameter set, `runSimulation` will kill the whole program.

  - `SimulationError`: in the simulation file created by SIMetrix an error message appears which says that the current simulation is aborted because of a simulation error. `runSimulation` saves the simulation file created by SIMetrix to a directory for failed simulations.
  - `SingularMatrix`: in the simulation file created by SIMetrix an error message appears which says that the current simulation is aborted because the current set of model parameters give a singular matrix. The `runSimulation` function saves the simulation file created by SIMetrix to a directory for failed simulations.
  - `MissingLicense`: in the simulation file created by SIMetrix an error message says that the user has no license to load Verilog-A models. If this exception is thrown several times for the same parameter set, `runSimulation` will kill the whole program.
  - `MissingFile`: SIMetrix did not produce any output file. The reason why this happens is probably due to SIMetrix instances clashing for some reasons. The current circuit file is saved to a directory for failed simulations. If this exception is thrown several times for the same parameter set, `runSimulation` will kill the whole program.

- Exceptions defined in `nspice_extr_print.py`:
  - `AdmsCodeError`: in the simulation file containing what ngspice prints to the standard output, a message may be found saying that an error occurred in the interface code to the Verilog-A model. This is due to weird combinations of model parameters.
  - `SimulationError`: an error message was printed to the standard error because the current simulation was aborted. This is due to weird combinations of model parameters.
  - `NgspiceFatalError`: an error message was printed to the standard error saying that ngspice was killed because of a fatal error. This may happen, for example, when there is an error in the circuit file or ngspice cannot find a shared library where the Verilog-A model is defined.

- Exceptions defined in `model_fitting.py`: `InterpolationError`. This exception is thrown when simulated data points cannot be interpolated over measured data points because simulated data are weird.

- Exceptions defined in `RR.py`: `RRerror`. The `SSQ_DiodeRR` and `SSQ_DiodeRR_V` classes defined in `model_fitting.py` raise this exception whether simulated and measured data are too different, or diode characteristic parameters cannot be extracted or synchronization of simulation and measured waveforms fails.

- Exceptions defined in `runSimulation.py`:
  - `ConvergenceError`: a simulation cannot converge within the time limit set in `runSimulation`. It will then be killed.
  - `TreeKillError`: Python tried to kill a SIMetrix instance and the parent process twice. This error is handled as a `ConvergenceError` exception by `runsimulation`.

Now we have all the basic ingredients to deal with the problem of fitting one or several device characteristics at the same time.

A.6 FITTING OF A SINGLE CHARACTERISTIC

Given a lumped charge model, fitting a device characteristic means to find model parameter values which minimize an error estimator between measured and simulated data. This is a single objective optimization problem.

In our case, the chosen objective function (error estimator) is not easy to minimize because:

- model equations are coupled;
- small changes in parameter values can lead to very different results;
- there is no explicit expression for the objective as function of model parameters so we cannot infer anything about its structure which can help us to choose the optimization algorithm, except that it is usually nonlinear.

Some commonly used nonlinear optimization technique are the Nelder-Mead method [78], quasi-Newton methods (like BFGS), nonlinear conjugate gradient methods and the Levenberg-Marquardt method [79]. These algorithms may fail to find a good solution if the search landscape is rugged (e.g. it has sharp bends, discontinuities, outliers, noise, and local optima). The CMA-ES can still perform very well on rugged landscapes.

Two scripts have been written to fit one characteristics at a time:

- `scripts/optimize_single_char_with_simplex.py`: optimization of a single characteristic with the Nelder-Mead algorithm. Deprecated.
- `scripts/optimize_single_char_with_cma.py`: single objective optimization with the CMA-ES. Recommended.

In the next section we explain how to define an objective function to minimize.

A.6.1 *Designing a single objective optimization*

A lot of instructions have to be specified in order to set up a single objective optimization.

All these instructions must be written in a text file using the YAML language. By parsing this file with function `load` from the `yaml` Python module, a Python dictionary is created. Each *(key, value)* pair of this dictionary stores information written in YAML language.

In the following, a YAML file for optimization of diode reverse recovery current is described. The complete code can be found in Appendix B.

- `title`: this should be a string identifying the simulation type.

  ```
  title: Reverse recovery
  ```

- `variable`: the value associated to this key is a dictionary. Each key of this dictionary is the name of a parameter to optimize. For example:

  ```
  variable:
    # ap model impact ionization current.
    ap:
      startpt: !!float 2.729878e+01
      bounds: [1, 50]
      transformation: scale01
      ...
  ```

  In this example, *ap* is the name of a parameter to optimize. *startpt* is the starting value of the parameter. This field is mandatory. The tag `!!float` forces YAML to interpret the following expression as a float number rather than a string. *bounds* is a list specifying the search interval for parameter *ap*. In this example we set the lower bound to 1 and the upper bound to 50. We could set just the lower bound

  ```
  bounds: [ 1, !!null ]
  ```

  or the upper bound

  ```
  bounds: [ !!null , 50]
  ```

  or none of them by omitting the *bounds* field. Note that a white space character is always required after the YAML tag `!!null`. *transformation* is a string or a list of strings which must match names of parameter transformations implemented in `opt_parser.py`. Transformations are applied in the same order they are listed. In our example, parameter *ap* must be normalized between 0 and 1

  ```
  transformation: scale01
  ```

  Currently supported transformations are:

  - Normalization between 0 and 1, indicated by the string `scale01`.
  - $\log_{10}$-transformation, indicated by the string `log10`.
  - Shifting by a fixed value, indicated by the string `shifting`. The value of the shifting must be specified.
  - Scaling by a fixed factor, indicated by the string `scaling`. The scaling factor must be specified.

If we want to apply all the previous transformations to a parameter, with a shifting factor of 4 and a scaling factor of 5, we should write

```
transformation:
- log10
- scale01
- shift: 4
- scaling: 5
```

If a variable parameter does not have to be transformed, the *transformation* field can be omitted.

- `fixed`: the value associated to this key is a dictionary with fixed parameters. These are device model parameters that must not be optimized and circuit parameters. For each *(key, value)* pair, *key* is the name of a fixed parameter and *value* is its numeric value. For example

```
fixed:
  # Zero biased junction capacitance
  cj0: !!float 7.798664e-08
  ...
  # Load current
  ILOAD: &ILOAD !!float 6000
  ...
```

- `simulation`: the value associated to this key is a dictionary containing information which is necessary to run a circuit simulation. In our example:

```
simulation:

  SIM_TEMPLATE: ...

  DEST_DIR: ...

  PARAMETER_FILE: parameters.txt

  ITER_FILE: iterRR.txt

  SHARED_LIBRARIES: ...

  SIMULATION_FILE_BASENAME: simulationRR

  SIMULATION_SAVE: 0

  NGSPICE_COMMAND_LINE_OPTIONS: ""
```

  - `SIM_TEMPLATE`. This is the path to the template of the circuit we want to simulate. This file is a template because no value is assigned to model

parameters and to some circuit parameters. These values will be specified by `runSimulation` which will produce a circuit file ready to be run in SIMetrix or ngspice (see Section A.4).

Both absolute and relative paths can be used. If relative paths are specified they must be consistent with the current working directory.

- `DEST_DIR`: name of the destination directory where all files generated during a single objective optimization are saved.
- `PARAMETER_FILE`: name of the file where model parameters of all simulations are saved.
- `ITER_FILE`: basename of the file where a simulation counter is stored.
- `SHARED_LIBRARIES`: name of the shared library created from a Verilog-A description of the device model under test. If only models embedded into the circuit simulator are used, this field can be left empty. Both relative and absolute paths can be used. Relative paths must be consistent with the current working directory.
- `SIMULATION_FILE_BASENAME`: basename of the file with interpolated simulated data and measured data used for error calculation.
- `SIMULATION_SAVE`: a flag. If this flag is true, simulation and measured data used for error calculation will be saved to files.
- `NGSPICE_COMMAND_LINE_OPTIONS`: additional ngspice command line options needed to run the current simulation. No additional command must be specified to run a reverse recovery simulation.

- `fitting`: the value associated to this key is a dictionary containing information which is needed to compute the error between measured and simulated data. In our example:

```
fitting:

  MEAS_DATA: !!python/object/apply:numpy.loadtxt
             args: ["..."]
             kwds: {skiprows: 0, comments: "#"}

  SSQ: DiodeRR

  OPTIONS_FOR_READING_SIMULATION_DATA:
    time:
      column_index: 0
    current:
      column_index: 1
      sign: -1
    voltage:
      column_index: 2
      sign: -1
```

```
SCALING_OF_WAVEFORM: *ILOAD

SCALING_FACTOR: !!float 1e-6

EXTRACT_FROM_CHAR:
- Qrr
- Erec
- IrMoments:
    wfunctions:
      wfunc_1st_derivative: {}
      wfunc_2nd_derivative: {}
    momentOrders: [2,3,4]

FITTING_KWARGS:
  step_over_Irr_ratio: 0.05
  delta_fraction_before_fold: 0.05
  delta_fraction_after_fold: 0.05
  factor_for_dVdt: 0.9

CHECK_CURVE_MOMENTS: 0

MAX_LOG_LOSS: !!float 10

BOUNDS: [ !!null , !!float 12e-6 ]

ERROR_WRT_YLOG: 0
```

- `MEAS_DATA`: a call to the `numpy.loadtxt` function is written in YAML language in order to load measured data. After parsing the text file with the whole set of YAML instructions, the value associated to this key is an array with measured data. In this way, data loading can be performed once for all.
- `SSQ`: ending name of the `SSQ` class used for error calculation.
- `OPTIONS_FOR_READING_SIMULATION_DATA`: a set of instructions to read the simulation file generated by SIMetrix or ngspice correctly. Each variable (time, current and voltage) is mapped to the corresponding column index in simulation data matrix. Moreover, depending on conventions used in the circuit file, column signs may have to be inverted.
- `SCALING_OF_WAVEFORM`: measured and simulated waveforms are divided by this value. For example, diode reverse recovery current is normalized to the load current to allow comparisons between different operating conditions, as suggested in [8].
- `SCALING_FACTOR`: a conversion factor used to express measured time points in seconds.
- `EXTRACT_FROM_CHAR`: a list of diode/IGBT characteristic values to be extracted. This list is currently used only in the `SSQ_DiodeRR` class. If no

characteristic has to be extracted, this field can be left empty. In our example, we extract the reverse recovery charge Q_{rr}, the reverse recovery energy E_{rec} and some moments of the current characteristic $i(t)$.

- `FITTING_KWARGS`: this is a dictionary of optional keyword arguments passed to the class which computes the error between measured and simulated data. It is currently used only in the `SSQ_DiodeRR` class. If no optional keyword argument is needed, an empty dictionary must be given.
- `CHECK_CURVE_MOMENTS`: this flag is used only in the `SSQ_DiodeRR` class. If it evaluates to true, moments of a simulated current waveform are compared to a statistical distribution to check if the simulation can be rejected before computing the error [41]. This statistical distribution is updated each time a simulation is carried out by applying an online unsupervised learning approach [105]. A statistic is computed as function of moments of the simulated curve. If the value of this statistic is too high, the simulation is rejected.
- `MAX_LOG_LOSS`: this is a threshold value which the statistic computed online from simulated waveform moments is compared to. If this statistic is bigger than the threshold, the simulation is rejected.
- `BOUNDS`: a list defining lower and/or upper bounds of the explanatory variable in the fitting error calculation. These bounds have nothing to do with model parameter lower and upper bounds. They have to be specified in SI base units. The `!!null` YAML tag indicates that there is no bound. In our example, we considered only time points up to 12 µs.
- `ERROR_WRT_YLOG`: if this flags evaluates to true, measured and simulated data are $\log_{10}$-transformed before the fitting error is computed.

- `optimization`: the value associated to this key is a dictionary containing information given to the optimization procedure. In our example:

```
optimization:
  value: SSQ_ABS

  DEFAULT_ERR: [!!float 1e3]

  SUMMARY_FILE: summary.txt

  optimizer_options:
    sigma0: !!float 0.3

    maxfevals: 999999

    maxiter: 2000

    tolfun: !!float 1e-4
```

```
tolfunhist: !!float 1e-5

tolx: !!float 1e-8

eval_initial_x: False
```

  - `value`: name of the error between measured and simulated data to compute. It must be one of the error computed by the `errorFamily` class (see Section A.2.2).
  - `DEFAULT_ERROR`: a list of one element representing a high error value returned when a simulation fails. As we wrote in Section 3.4.4, the default error value should be kept as low as possible, just above the limit below which infeasible solutions are optimal. Of course, this default value depends on the error estimator to be minimized.
  - `SUMMARY_FILE`: basename of the file where the following information is saved at the end of an optimization:
    * the termination criterion that was met;
    * parameters that minimize the objective function;
    * minimum of the objective function;
    * number of performed iterations of the optimizer;
    * number of objective function calls made.
  - `optimizer_options`: a dictionary of options given to the chosen optimizer. In our example, some options for the CMA-ES are shown.

- `options`: a dictionary where various options are collected. In our example, we have:

```
options:
  MEAS_LINES: 'color="b", xdata=self.MEAS.x*1e6'

  SIM_LINES: 'color="r", xdata=self.SIM.x*1e6'

  SIMULATION_FIGURE: >
    ylabel('i [A]');
    ...

  ABSERR_LINE: 'color="k", xdata=AbsErr.x*1e6'
  RELERR_LINE: 'color="k", xdata=RelErr.x*1e6'

  ABSERR_FIGURE: >
    ylabel('Absolute error');

  RELERR_FIGURE: >
    xlabel('t [' + unichr(181) + 's]');
    ylabel('Relative error');
```

```
FAILURES_DIRNAME: "..."

FAILURES_FILE: failures.log

ERROR_SAVE: 0

ERROR_FILE: errorsRRI.txt

ORDER_OF_ERRORS:
  - S_ABS
  - SSQ_ABS
  - SSQ_REL
  - VAR_EST
  - RMSE
  - RMSErel
  - NRMSE
  - MAX_ABSERR
  - MAX_RELERR
  - RelErr_Qrr
  - RelErr_Erec
```

- `MEAS_LINES`: options for plotting measured waveforms.
- `SIM_LINES`: options for plotting simulated waveforms.
- `SIMULATION_FIGURE`: additional options for figures where measured and simulated data are plotted.
- `ABSERR_LINE`: options for plotting the absolute error.
- `RELERR_LINE`: options for plotting the relative error.
- `ABSERR_FIGURE`: additional options for figures where the absolute error is plotted.
- `RELERR_FIGURE`: additional options for figures where the relative error is plotted.
- `FAILURES_DIRNAME`: name of the directory where files generated by a failed simulation are saved;
- `FAILURES_FILE`: basename of the file where failed simulations are reported.
- `ERROR_SAVE`: if this flag evaluates to true, several fitting errors will be saved each time the objective function is evaluated.
- `ERROR_FILE`: basename of the text file where fitting errors are saved if `ERROR_SAVE` is true.
- `ORDER_OF_ERRORS`: a list of strings specifying in which order fitting errors are saved. In our example, we also save the relative error in the reverse recovery charge Q_{rr} and in the reverse recovery energy E_{rec}.

Once a YAML file for single objective optimization has been properly edited, an optimization can be run using the `optimize_single_char_with_cma.py` script.

A.6.2 *Post processing of single objective optimization results*

Once a single objective optimization has terminated, the destination directory will contain the following files:

- A copy of the YAML file used to set up the optimization.
- A file where the simulation counter has been saved.
- A sub-directory containing a plot of the optimum parameter set.
- A file with parameter values of every simulation, including failed simulations. Parameter values of each simulation are saved column-wise.
- If `ERROR_SAVE` was set to true, a file with several error estimators computed for every simulation, including failed simulations. For each simulation, different error estimators are saved column-wise.
- If `SIMULATION_SAVE` was set to true, simulation data used for error calculation for each parameter set.

When loading parameters and errors of all simulations into two dimensional arrays, parameters and errors of the ith simulation correspond to the ith row of parameter and error arrays, respectively.

This correspondence is exploited by post-processing routines which plot parameter and error trends versus simulation number. We expect that as the simulation number increases, parameters converge and all errors decrease, with slightly different trends among different error estimators.

Postprocessing of single objective optimizations can be performed by running the script `script/plot_single_char.py`. Actually, this script can be used while an optimization is still running to check how an optimizer is working.

A.7 CONCURRENT FITTING OF SEVERAL CHARACTERISTICS

In Chapter 1, parameter extraction of power semiconductor devices has been formulated as a multi-objective optimization problem. In our work, MOEAs have been chosen as preferred solving methods for this kind of problems.

In the Python library for parameter extraction, the following MOEAs are available:

- NSGA-II;
- PAES.
- R-NSGA-II;
- SPEA2;
- SPEA2+;
- MO-CMA-ES.

A.7.1 *Designing a multi objective optimization*

As for single objective optimization, the set of instructions for running a multi-objective optimization must be specified in a text file using the YAML language. In the following section, a YAML file for optimization of diode i–v, reverse recovery current and reverse recovery voltage with the NSGA-II is briefly described. This file was used within the first MOEA experiment presented in Section 4.3. The complete code can be found in Appendix B.

- As for single objective optimization, the following three fields must be specified:
  - `title`: brief description of the YAML file;
  - `variable`: dictionary of parameters to be optimized;
  - `fixed`: dictionary of fixed parameters.

  Structures of both the `variable` and the `fixed` dictionary are the same as before. A starting value for each parameter in `variable` must be specified anyway, even if it is not used by the multi-objective algorithm.

- `multiobjopt`: a dictionary with two entries:
  - `characteristics`: a dictionary defining a multi-objective optimization problem. This dictionary has an entry for each characteristic to be optimized. In our example:

    ```
    multiobjopt:
      characteristics:
        dc:
          ...
        rr_current:
          ...
        rr_voltage:
          ...
    ```

    The entry associated to each characteristic is a dictionary with a very similar structure to that of a YAML file for single objective optimization. Indeed, each dictionary has the following entries: `title`, `simulation`, `fitting`, `optimization` and `options`. The first three entries are identical to the corresponding entries of a YAML file for single objective optimization. Some changes appear in the `optimization` field. In our example:

    ```
    multiobjopt:
      characteristics:
        ...
        rr_current:
          ...
          optimization:
              value: SSQ_ABS

              DEFAULT_ERR: [!!float 100]
    ```

```
OBJECTIVE_LIST: [SSQ_ABS]

REFERENCE_POINTS:
    SSQ_ABS: [!!float 1e-4, 2, !!float 1e-4]

WEIGHT_LIST: [1]
```

  * `value`: this is always a string with the name of one of the error estimators computed by the `errorFamily` class (see Section A.2.2). Two other strings are possible: `WeightedSum` and `ManyObjectives`. In the former case, a weighted sum of the objectives specified in `OBJECTIVE_LIST` is computed. In the latter case, objectives specified in `OBJECTIVE_LIST` are optimized separately.
  * `DEFAULT_ERR`: a list with as many default error values as the objectives defined in both `value` and `OBJECTIVE_LIST` fields. These errors are returned when the simulation fails.
  * `OBJECTIVE_LIST`: list of objectives extracted from current characteristic. Each objective can be one of the error estimators computed by the `errorFamily` class or the relative error between measured and simulated diode characteristic values, like the reverse recovery charge Q_{rr}.
  * `REFERENCE_POINTS`: reference points for the R-NSGA-II. For each objective in `OBJECTIVE_LIST`, values must be specified to define the reference points which will be used by the R-NSGA-II. In our example, three reference points are provided.
  * `WEIGHT_LIST`: list of weights. These weights can be used to build a weighted sum of the objectives in `OBJECTIVE_LIST` or to bias some of these objectives more than others in the R-NSGA-II.

  In the `options` field, some keys which can be found in files for single objective optimization are missing because they have been collected in another position of the global YAML file.

- `moea_options`: options given to the multi-objective evolutionary algorithm. See Chapter 3 for a proper explanation of these options. In our example:

```
multiobjopt:
  characteristics:
    ...
  moea_options:
    pop_size: 120

    max_generations: 250

    nb_of_subgroups: 5

    epsilon_for_niching: !!float 1e-3
```

```
        seed_file: "..."

        crossover_rate: 0.9

        mutation_rate: 0.08

        distribution_index_for_crossover: 10

        distribution_index_for_mutation: 20

        EPS: !!float 1e-14
```

- `options`: various options for the multi-objective optimization procedure. In our example:

```
options:
    ROOT_DIR: "..."

    SIMULATION_SAVE: 0

    FAILURES_DIRNAME: "..."

    FAILURES_FILE: failures.log

    ERROR_SAVE: 0

    STATISTICS_FILE: statistics.txt

    INDIVIDUALS_FILE: individuals.txt

    FINAL_PARETO_FRONTIER: ...

    GENERATION_WISE_PARETO_FRONTIER: ...

    SUCCESS_REPORT: success.txt
```

  - `ROOT_DIR`: name of the destination directory where multi-objective optimization results are saved.
  - `SIMULATION_SAVE`: if this flag evaluates to true, simulation and measured data used for error calculation will be saved for each characteristic. In YAML files for single objective optimization, this flag can be found in the `simulation` field.
  - `ERROR_SAVE`: if this flag evaluates to true, several fitting errors will be saved for each characteristic. In YAML files for single objective optimization, this flag can be found in the `options` field.
  - `STATISTICS_FILE`: basename of the file with generation statistics of the population throughout the run.

- `INDIVIDUALS_FILE`: basename of the file with all individuals generated by the multi-objective optimizer.
- `FINAL_PARETO_FRONTIER`: basename of the file with the approximated Pareto frontier found by the optimizer.
- `GENERATION_WISE_PARETO_FRONTIER`: basename of the file with the approximated Pareto frontier computed generation-wise.
- `SUCCESS_REPORT`: basename of the file used to report that the optimizer ended successfully.

Once a YAML file for multi objective optimization has been properly edited, the `multi_objective_optimization.py` script can be used to run the optimization.

A.7.2 *Post processing of multi objective optimization results*

The `opt_parser.processMOEAresults` class provides methods to analyze the results of a multi-objective optimization run. An example of application is given in the `single_moea_performance_assessment.py` script.

The `processMOEAresults` class provides methods to:

- plot a scattergraph or a scattergraph matrix of solutions generated by a MOEA in the objective space.
- plot a parallel coordinate graph of solutions generated by a MOEA in the objective space. This diagram is particularly useful to represent high dimensional data. It can also help to detect some relationships between plotted data, as shown in Section 3.6.
- extract nondominated vectors at each generation.
- make a GIF animation of the progression of the approximated Pareto frontier of the nondominated vectors throughout generations.
- extract the final approximated Pareto frontier.
- control the extent of obtained Pareto solutions by using the same ε-clearing procedure implemented in the R-NSGA-II. This is usually done to provide the decision-maker with a limited number of representative Pareto points.
- perform dimensional reduction (i.e., objective selection) with the principal component analysis proposed in [23].

All these methods are employed in the `single_moea_performance.py` script. This script also plots simulated characteristics corresponding to Pareto solutions in order to help the decision-maker in selecting a final parameter set.

A.8 PERFORMANCE COMPARISON OF SEVERAL MOEAS

The `performance_assessment_of_moeas` module provides routines to design a performance comparison of MOEAs and to analyze its results, as shown in Chapter 4. These routines are employed in two scripts:

- `run_moea_tests.py`: design and execute a MOEA experiment;
- `moea_performance_assessment.py`: analyze the data.

A set of instructions must be provided to both these scripts through a text file written in YAML language. In the following, an example of such a file is described. It was used to perform the first MOEA experiment presented in Section 4.3. The complete code is given in Appendix B.

- `MOEA_LIST`: list of MOEAs to compare. In our example:

```
MOEA_list: [nsga2, r_nsga2, mo_cma_es_p, mo_cma_es_p_with_recomb]
```

- `yaml_file_list`: list of YAML files associated to MOEAs included in the comparison. In the first test case, for the NSGA-II and the R-NSGA-II, we used the YAML file described in Section A.7.1.
- `Performance_indicators`: list of quality indicators used in the MOEA comparison.

```
Performance_indicators: [hyp, eps]
```

- `Sample_size`: size of approximation set samples in the MOEA comparison;
- `ROOT_DIR`: destination directory of the experiment's results;
- `INITIAL_POPS`: name of the child directory of `ROOT_DIR` where initial populations are saved;
- `Number_of_processes`: number of parallel CPU processes which MOEA runs are distributed over;
- `List_of_runs`: basename of the file where the list of tests to run is saved;
- `BOUNDS`: basename of the file where lower and upper bounds of the objective vectors are saved;
- `Script_for_post_process_of_single_moea`: this is the absolute path to the Python script for postprocessing of single MOEA run;
- `Script_for_eafdiffplot`: absolute path to the R script for plotting differences in the empirical attainment functions of two data set in the biobjective case;
- `Script_for_boxplots`: absolute path to the R script for plotting quality indicator boxplots;
- `REFERENCE_SET`: basename of the file where the reference set for computing unary indicators is saved;
- `PISA`: dictionary containing absolute paths to PISA software tools used in MOEA comparison analysis;
- `multipleTest`: absolute path to the source code by García and Herrera that implements post-hoc procedures described in Section 3.7.4.4.

B

YAML FILE EXAMPLES

Listing 1: A YAML file for optimization of diode reverse recovery waveform.

```
# YAML

title: Reverse recovery

# Variable used in sensitivity analysis
variable:
    # ap model impact ionization current.
    ap:
        startpt: !!float 2.729878e+01
        bounds: [1, 50]
        transformation: scale01

    # tauL, tauH and Eth model a smooth change
    # in the carrier life time depending on the LT parameter.
    tauL:
        startpt: !!float 3.829042e-06
        bounds: [!!float 1e-6, !!float 1e-5]
        transformation: scale01

    tauH:
        startpt: !!float 7.272543e-07
        bounds: [!!float 1e-7, !!float 5e-6]
        transformation: scale01

    EM:
        startpt: !!float 1.724436e-04
        bounds: [!!float 1e-6, !!float 1e-3]
        transformation: [log10, scale01]

    # IB and RB are the two new parameters for
    # voltage dependent transit time function.
    IB:
        startpt: !!float 4.591270e+02
        bounds: [!!float 1, !!float 300]
        transformation: [log10, scale01]

    RB:
        startpt: !!float 1.123330e+01
        bounds: [100, 1000]
        transformation: scale01

    # Transit time of electrons
    T0:
        startpt: !!float 1e-5
```

```
        bounds: [!!float 1e-6, !!float 1e-5]
        transformation: scale01

    # Mobility ratio
    b:
        startpt: 3.639822
        bounds: [1, 6]
        transformation: scale01


# Fixed parameters
fixed:
    ## Diode parameters

    # bp model impact ionization current.
    bp: !!float 1e5
#        startpt: !!float 1e5
#        bounds: [!!float 5e4, !!float 5e5]
#        trasformation: scale01

    # Threshold between tauL and tauH
    Eth: !!float 1.3e5
#        startpt: !!float 1.3e5
#        bounds: [!!float 6e4, !!float 2e5]
#        transformation: scale01

    # Thermal equilibrium electron charge in the base
    QB: !!float 4.999933e-04

    # Thermal equilibrium hole charge in the base
    Qp0: !!float 5.477687e-10

    # Series resistance
    rs: !!float 3.617722e-04

    # Ideality factors
    NN: 1.68713

    NE: !!float 1.159609e+01

    # Built-in potential - T=300K
    vj0: 0.4

    # Grading coefficient
    m: 0.4832172

    # Zero biased junction capacitance
    cj0: !!float 7.798664e-08

    # Coefficient for depletion cap formula in forward bias
    fc: !!float 0.5

    # Hole mobility coefficient
```

```
    alpha: !!float -2.2

    # Temperature coefficient of series resistance
    Trs: !!float 0.0

    # Temperature coefficient of EM
    TEM: !!float 0

    # Temperature dependent coefficient of tau3
    Ttau: !!float 0.66

    # Default Device temperature in deg C
    Tnom: !!float 140

    ## Circuit parameters

    VCC: &VCC !!float 3200

    INITIAL_V: !!float -4.590986

    LSTRAY: !!float 2.15e-6

    ILOAD: &ILOAD !!float 6e3

    LCL: !!float 3e-7

    CCL: !!float 1e-5

    LS2: !!float 1.5e-6

    RS: 7.3

    ABSTOL: !!float 5e-4

    VNTOL: !!float 1e-3

    RELTOL: !!float 1e-3

    # Final time of transient analysis in MICROSECONDS.
    TSTOP: 17


simulation:

    # Circuit file
    SIM_TEMPLATE: "/home/daniele/workspace/parameter_extraction
       /dpinth_tau_ii_idf_RR_circuit.ngspice"

    # Destination directory where simulation files are saved.
    DEST_DIR: "/media/ACER/Users/Daniele/Documents/Pollitecnico
       /01_tesi_simulazioni/tmp"
```

```
    # Name of the file with parameter values for each
       simulation.
    PARAMETER_FILE: parameters.txt

    # File with iteration number
    ITER_FILE: iterRR.txt

    # Shared libraries with diode model definition
    SHARED_LIBRARIES: "/home/daniele/workspace/
       parameter_extraction/libdpinth_tau_ii_idf.so"

    # Basename of the file where simulation data are saved
    SIMULATION_FILE_BASENAME: simulationRR

    # If SIMULATION_SAVE is True, simulation data will be saved
        to files
    SIMULATION_SAVE: 0

    # Command line options for starting ngspice
    NGSPICE_COMMAND_LINE_OPTIONS: ""


fitting:

    # Measured data array
    MEAS_DATA: !!python/object/apply:numpy.loadtxt
               args: ["/media/ACER/Users/Daniele/Documents/ABB
                  /02_measurement_data/02_TCAD_DATA/RR_I_V_int.
                  txt"]
               kwds: {skiprows: 0, comments: "#"}

    # SSQ class used for error calculation
    SSQ: DiodeRR

    # Map variables to column indices in simulation data matrix
       .
    # Moreover, according to conventions used in circuit files,
        vector signs have or do not have to be inverted.
    # If sign = 1, there is no need for inversion. Otherwise
       set sign = -1
    OPTIONS_FOR_READING_SIMULATION_DATA:

        time:
            column_index: 0

        current:
            column_index: 1
            sign: -1

        voltage:
            column_index: 2
            sign: -1
```

```
    # Scale for waveforms. Reverse recovery current is divided
       by SCALING_OF_WAVEFORM
    # before error is computed.
    SCALING_OF_WAVEFORM: *ILOAD

    # Scaling factor to express time values from data files in
       seconds
    SCALING_FACTOR: !!float 1e-6

    # Parameters that should be extracted from current
       simulation
    EXTRACT_FROM_CHAR:
#     - Qrr
#     - Erec
#     - IrMoments:
#         wfunctions:
#             wfunc_1st_derivative: {}
#             wfunc_2nd_derivative: {}
#         momentOrders: [2,3,4]

    # Optional keyword arguments to be passed to the class used
        for error calculation
    FITTING_KWARGS: {}
#         step_over_Irr_ratio: 0.05
#         delta_fraction_before_fold: 0.05
#         delta_fraction_after_fold: 0.05
#         factor_for_dVdt: 0.9

    # Choose if curve moments should be checked for outlier
       detection
    CHECK_CURVE_MOMENTS: 0

    # Maximum logarithmic loss. If the logaritmic loss
       associated to moments of
    # a simulated characteristics is bigger than this value,
       such characteristics
    # is rejected and an exception is raised.
    MAX_LOG_LOSS: !!float 10

    # Independent variable bounds for error calculation. Use
       base SI units.
    # Use !!null tag if there is no lower/upper bound.
    BOUNDS: [ !!null , !!float 12e-6 ]

    # If ERROR_WRT_YLOG is True, measured and simulated data
       have to be
    # log-transformed before error computation.
    ERROR_WRT_YLOG: 0

    # Threshold used to decide when absolute error has to be
       used instead of relative error in SSQ_ABS_REL
       computation
    RELERR_th: !!float 10
```

```
optimization:

    # Error to be extracted
    value: SSQ_ABS

    # Error returned by the optimizer if an exception occurs.
    # Choose DEFAULT_ERR value according to the error estimator
        specified in the 'value' field.
    DEFAULT_ERR: [!!float 1e3]

    # Name of the file with optimization results
    SUMMARY_FILE: summary.txt

    optimizer_options:
        # Initial global step-size. If the optimum is expected
            to be in the
        # initial search interval [a,b]**n we may choose sigma0
            = 0.3*(b-a),
        # as it is suggested in Nikolaus Hansen, "The CMA
            Evolution Strategy: A Tutorial", March 12, 2011.
        # So if parameters are scaled between [0,1] eventually,
            sigma0 = 0.3*(1-0) = 0.3.
        sigma0: !!float 0.3

        # Maximum number of function evaluation
        maxfevals: 999999

        # Maximum number of iteration
        maxiter: 2000

        # Termination criterion: tolerance in function value
        tolfun: !!float 1e-4

        # Termination criterion: tolerance in function value
            history
        tolfunhist: !!float 1e-5

        # Termination criterion: tolerance in x-changes
        tolx: !!float 1e-8

        eval_initial_x: False


options:

    # Option for plotting measured data line
    MEAS_LINES: 'color="b", xdata=self.MEAS.x*1e6'

    # Option for plotting simulated data line
    SIM_LINES: 'color="r", xdata=self.SIM.x*1e6'
```

```
    # Options for figures that will be generated
    SIMULATION_FIGURE: >
        ylabel('i [A]');
        legend(['measured', 'simulated'], loc='lower right',
           prop=FontProperties(size='small'));

    # Option for plotting error lines
    ABSERR_LINE: 'color="k", xdata=AbsErr.x*1e6'
    RELERR_LINE: 'color="k", xdata=RelErr.x*1e6'

    # Options for absolute error figure
    ABSERR_FIGURE: >
        ylabel('Absolute error');

    # Option for relative error figure
    RELERR_FIGURE: >
        xlabel('t [' + unichr(181) + 's]');
        ylabel('Relative error');

    # Directory where wrong simulation files and an exception
       report are saved
    FAILURES_DIRNAME: "/media/ACER/Users/Daniele/Documents/
       Pollitecnico/01_tesi_simulazioni/bctm_diode_failures"

#    # File where exceptions are reported
#    FAILURES_FILE: failures.log

    # Flag used to choose if errors should be saved or not
    ERROR_SAVE: 0

    # Name of the file where all types of errors will be saved
    ERROR_FILE: errorsRRI.txt

    # Errors are saved in the following order
    ORDER_OF_ERRORS:
        - S_ABS
        - SSQ_ABS
        - SSQ_ABS_L
        - SSQ_REL
        - SSQ_REL_L
        - VAR_EST
        - RMSE
        - RMSErel
        - NRMSE
        - MAX_ABSERR
        - MAX_RELERR
        - MEAN_RELERR
#         - SSQ_ABS_REL
#         - RelErr_Qrr
#         - RelErr_Erec
```

Listing 2: A YAML file for multi-objective optimization with the NSGA-II/R-NSGA-II.

```yaml
# YAML

# Caution: spacing is important.
# For example:
# - always keep at least one blank space character after a
   colon
# - use the same indentation level for each key of a dictionary
# - always keep at least one blank space character after YAML
   tags like !!float, !!null (when it appears in a list)


title: Fitting of TCAD DC and RR data with BCTM model and NSGA-
   II/R-NSGA-II algorithms


# Variable parameters
variable:

    # Medium-high level ideality factor
    NE:
        startpt: 6
        bounds: [1, 12]
        transformation: scale01

    # Low level ideality factor
    NN:
        startpt: 2
        bounds: [1, 4]
        transformation: scale01

    # Thermal equilibrium electron charge in the base
    QB:
        startpt: !!float 7.343108e-08
        bounds: [!!float 1e-8, !!float 5e-4]
        transformation: [log10, scale01]

    # Thermal equilibrium hole charge in the base
    Qp0:
        startpt: !!float 8.93262473015e-10
        bounds: [!!float 1e-12, !!float 1e-8]
        transformation: [log10, scale01]

    # Series resistance
    rs:
        startpt: !!float 5e-4
        bounds: [!!float 2e-4, !!float 8e-4]
        transformation: scale01

    EM:
        startpt: !!float 1e-5
        bounds: [!!float 1e-6, !!float 1e-3]
        transformation: [log10, scale01]
```

```
    # ap model impact ionization current.
    ap:
        startpt: !!float 7.676612
        bounds: [1, 50]
        transformation: scale01

    # Mobility ratio
    b:
        startpt: 3
        bounds: [1, 6]
        transformation: scale01

    # tauL, tauH and Eth model a smooth change
    # in the carrier life time depending on the Emax parameter.
    tauL:
        startpt: !!float 9.2553992915e-07
        bounds: [!!float 5e-7, !!float 5e-6]
        transformation: scale01

    tauH:
        startpt: !!float 3.71485855789e-06
        bounds: [!!float 5e-7, !!float 1e-5]
        transformation: scale01

    # Transit time of electrons
    T0:
        startpt: !!float 1.874952e-06
        bounds: [!!float 1e-6, !!float 5e-5]
        transformation: [log10, scale01]

    # RB and IB are the two new parameters for
    # voltage dependent transit time function.
    IB:
        startpt: !!float 1e3
        bounds: [!!float 1e2, !!float 1e4]
        transformation: [log10, scale01]

    RB:
        startpt: !!float 10
        bounds: [!!float 1, !!float 50]
        transformation: scale01


# Fixed parameters
fixed:
    ## Diode parameters

    # Built-in potential - T=300K
    vj0: 0.4

    # Grading coefficient for junction capacitance
    m: 0.4832172
```

```
    # Zero biased junction capacitance
    cj0: !!float 7.798664e-08

    # bp models impact ionization current.
    bp: !!float 1e5

    # Threshold between tauL and tauH
    Eth: !!float 1.3e5

    # Point where SPICE capacitance becomes linear
    fc: 0.5

    # Hole mobility coefficient
    alpha: !!float -2.2

    # Temperature coefficient of series resistance
    Trs: !!float 0.0

    # Temperature coefficient of EM
    TEM: !!float 0

    # Temperature dependent coefficient of tau3
    Ttau: !!float 0.66

    # Default Device temperature in deg C
    Tnom: !!float 140

    ## Circuit parameters

    VCC: &VCC !!float 3200

    INITIAL_V: !!float -4.590985744762529741

    LSTRAY: !!float 2.150000e-06

    ILOAD: &ILOAD !!float 6000

    LCL: !!float 3e-7

    CCL: !!float 1e-5

    LS2: !!float 1.5e-6

    RS: 7.3

    ABSTOL: !!float 1e-3

    VNTOL: !!float 1e-3

    RELTOL: !!float 1e-3

    # Final time of transient analysis in MICROSECONDS.
    TSTOP: 18
```

```
# Inputs for multi-objective optimization
multiobjopt:

    characteristics:

        dc:

            title: I-V characteristic

            simulation:

                # Circuit file
                SIM_TEMPLATE: "/home/daniele/workspace/
                    parameter_extraction/
                    dpinth_tau_ii_idf_DC_circuit.ngspice"

                # Destination directory where files produced by
                     capacitance simulations are saved.
                DEST_DIR: "/media/ACER/Users/Daniele/Documents/
                    Pollitecnico/01_tesi_simulazioni/22
                    aa_dpinth_tau_ii_idf_TCAD_mop_DC_RR_nsga2/dc
                    "

                # Name of the file with parameter values for
                    each simulation.
                PARAMETER_FILE: parameters.txt

                # File with iteration number
                ITER_FILE: iterDC.txt

                # Shared libraries with diode model definition
                SHARED_LIBRARIES: "/home/daniele/workspace/
                    parameter_extraction/libdpinth_tau_ii_idf.so
                    "

                # Basename of the file where simulation data
                    are saved
                SIMULATION_FILE_BASENAME: simulationDC

                # Command line options for starting ngspice
                NGSPICE_COMMAND_LINE_OPTIONS: ""

            fitting:

                # Measured data array
                MEAS_DATA: !!python/object/apply:numpy.loadtxt
                           args: ["/media/ACER/Users/Daniele/
                               Documents/ABB/02_measurement_data
                               /02_TCAD_DATA/
                               TCAD_not_measurement_T140C_IV.txt
                               "]
```

```
                        kwds: {skiprows: 0, comments: "#"}

            # SSQ class used for error calculation
            SSQ: DiodeDC

            # Scale for waveforms. Reverse recovery current
                is divided by SCALING_OF_WAVEFORM
            # before error is computed.
            SCALING_OF_WAVEFORM: 1

            # Independent variable bounds for error
               calculation. Use base SI units.
            # Use !!null tag if there is no lower/upper
               bound.
            BOUNDS: [ !!null , !!null ]

            # If ERROR_WRT_YLOG is True, measured and
               simulated data have to be
            # log-transformed before error computation.
            ERROR_WRT_YLOG: 0

            # Threshold used to decide when absolute error
               has to be used instead of relative error in
               SSQ_ABS_REL computation
            RELERR_th: !!float 10

            # Keyword arguments to be passed to the class
               used for error calculation
            FITTING_KWARGS: {}

        optimization:

            # Error to be extracted
            # If you change this, change also ['multiobjopt
               ']['characteristics'][*]['optimization']['
               DEFAULT_ERR']
            value: S_ABS

            # Error returned by the optimizer if an
               exception occurs.
            # Choose DEFAULT_ERR value according to the
               error estimator specified in ['multiobjopt
               ']['characteristics'][*]['optimization']['
               value']
            # If ['multiobjopt']['characteristics'][*]['
               optimization']['value'] is equal to '
               ManyObjective', DEFAULT_ERR
            # must be a list with as many error values as
               the parameters specified in ['multiobjopt
               ']['characteristics'][*]['optimization']['
               OBJECTIVE_LIST'].
            DEFAULT_ERR: [!!float 1e6]
```

```
        # List of objective functions extracted from
           current characteristic.
        OBJECTIVE_LIST: [S_ABS]

        # Needed by R-NSGA-II algorithm.
        # There must be a key entry for each element in
            'OBJECTIVE_LIST'.
        REFERENCE_POINTS:
            S_ABS: [50, 50, !!float 1e4]

        # This weight list can be used in different
           ways:
        # - to build a weighted sum of the values
           specified in 'OBJECTIVE_LIST' if a weighted
           sum approach
        #   should be used for a many objective problem
        # - to bias some objectives of 'OBJECTIVE_LIST'
            more than others in the reference point
           method by Deb et all.
        # - ...
        WEIGHT_LIST: [1]

    options:

        # Option for plotting measured data line
        MEAS_LINES: 'marker=".", markersize=2'

        # Option for plotting simulated data line
        SIM_LINES: ''

        # Options for figures that will be generated
        SIMULATION_FIGURE: >
            xlabel('V [V]', size='small');
            ylabel('I [A]', size='small');
            legend(['measured', 'simulated'], loc='
               upper left', prop=FontProperties(size='
               small'));

        # Option for plotting error lines
        ABSERR_LINE: 'color="k"'
        RELERR_LINE: 'color="k"'

        # Options for absolute error figure
        ABSERR_FIGURE: >
            ylabel('Absolute error');

        # Option for relative error figure
        RELERR_FIGURE: >
            xlabel('V [V]');
            ylabel('Relative error');

        # Name of the file where all types of errors
           will be saved
```

```
            ERROR_FILE: errorsDC.txt

            # Errors are saved in the following order
            ORDER_OF_ERRORS:
                - S_ABS
                - SSQ_ABS
                - SSQ_ABS_L
                - SSQ_REL
                - SSQ_REL_L
                - VAR_EST
                - RMSE
                - RMSErel
                - NRMSE
                - MAX_ABSERR
                - MAX_RELERR
                - MEAN_RELERR
        #         - SSQ_ABS_REL

    rr_current:

        title: reverse recovery current

        simulation:

            # Circuit file - check
               OPTIONS_FOR_READING_SIMULATION_DATA also.
            SIM_TEMPLATE: "/home/daniele/workspace/
               parameter_extraction/
               dpinth_tau_ii_idf_RR_circuit.ngspice"

            # Destination directory where files produced by
                capacitance simulations are saved.
            DEST_DIR: "/media/ACER/Users/Daniele/Documents/
               Politecnico/01_tesi_simulazioni/22
               aa_dpinth_tau_ii_idf_TCAD_mop_DC_RR_nsga2/
               rr_current"

            # Name of the file with parameter values for
               each simulation.
            PARAMETER_FILE: parameters.txt

            # File with iteration number
            ITER_FILE: iterRR.txt

            # Shared libraries with diode model definition
            SHARED_LIBRARIES: "/home/daniele/workspace/
               parameter_extraction/libdpinth_tau_ii_idf.so
               "

            # Basename of the file where simulation data
               are saved
            SIMULATION_FILE_BASENAME: simulationRR
```

```
    # Command line options for starting ngspice
    NGSPICE_COMMAND_LINE_OPTIONS: ""

fitting:

    # Measured data array
    MEAS_DATA: !!python/object/apply:numpy.loadtxt
               args: ["/media/ACER/Users/Daniele/
                  Documents/ABB/02_measurement_data
                  /02_TCAD_DATA/RR_I_V_int.txt"]
               kwds: {skiprows: 0, comments: "#"}

    # SSQ class used for error calculation
    SSQ: DiodeRR

    # Map variables to column indices in simulation
       data matrix.
    # Moreover, according to conventions used in
       circuit files, vector signs have or do not
       have to be inverted.
    # If sign = 1, there is no need for inversion.
       Otherwise set sign = -1
    OPTIONS_FOR_READING_SIMULATION_DATA:

        time:
            column_index: 0

        current:
            column_index: 1
            sign: -1

        voltage:
            column_index: 2
            sign: -1

    # Scale for waveforms. Reverse recovery current
       is divided by SCALING_OF_WAVEFORM
    # before error is computed.
    SCALING_OF_WAVEFORM: *ILOAD

    # Scaling factor to express time values from
       data files in seconds
    SCALING_FACTOR: !!float 1e-6

    # Parameters that should be extracted from
       current simulation
    EXTRACT_FROM_CHAR:
#    - Qrr
#    - Erec
#    - IrMoments:
#        wfunctions:
#            wfunc_1st_derivative: {}
#            wfunc_2nd_derivative: {}
```

```
#         momentOrders: [2,3,4]

    # Optional keyword arguments to be passed to
      the class used for error calculation
    FITTING_KWARGS: {}
#         step_over_Irr_ratio: 0.05
#         delta_fraction_before_fold: 0.05
#         delta_fraction_after_fold: 0.05
#         factor_for_dVdt: 0.9

    # Choose if curve moments should be checked for
       outlier detection
    CHECK_CURVE_MOMENTS: 0

    # Maximum logarithmic loss. If the logaritmic
      loss associated to moments of
    # a simulated characteristics is bigger than
      this value, such characteristics
    # is rejected and an exception is raised.
    MAX_LOG_LOSS: !!float 10

    # Independent variable bounds for error
      calculation. Use base SI units.
    # Use !!null tag if there is no lower/upper
      bound.
    BOUNDS: [ !!null , !!float 12e-6 ]

    # If ERROR_WRT_YLOG is True, measured and
      simulated data have to be
    # log-transformed before error computation.
    ERROR_WRT_YLOG: 0

    # Threshold used to decide when absolute error
      has to be used instead of relative error in
      SSQ_ABS_REL computation
    RELERR_th: !!float 10

optimization:

    # Error to be extracted
    # If you change this, change also ['multiobjopt
      ']['characteristics'][*]['optimization']['
      DEFAULT_ERR']
    value: SSQ_ABS

    # Error returned by the optimizer if an
      exception occurs.
    # Choose DEFAULT_ERR value according to the
      error estimator specified in ['multiobjopt
      ']['characteristics'][*]['optimization']['
      value']
```

```
        # If ['multiobjopt']['characteristics'][*]['
           optimization']['value'] is equal to '
           ManyObjective', DEFAULT_ERR
        # must be a list with as many error values as
           the parameters specified in ['multiobjopt
           ']['characteristics'][*]['optimization']['
           OBJECTIVE_LIST'].
        DEFAULT_ERR: [!!float 100]

        # List of objective functions extracted from
           current characteristic.
        OBJECTIVE_LIST: [SSQ_ABS]

        # Needed by R-NSGA-II algorithm
        # There must be a key entry for each element in
            'OBJECTIVE_LIST'.
        REFERENCE_POINTS:
            SSQ_ABS: [!!float 1e-4, 2, !!float 1e-4]

        # This weight list can be used in different
           ways:
        # - to build a weighted sum of the values
           specified in 'OBJECTIVE_LIST' if a weighted
           sum approach
        #   should be used for a many objective problem
        # - to bias some objectives of 'OBJECTIVE_LIST'
            more than others in the reference point
           method by Deb et all.
        # - ...
        WEIGHT_LIST: [1]

    options:

        # Option for plotting measured data line
        MEAS_LINES: 'color="b", xdata=self.MEAS.x*1e6,
           marker=".", markersize=2'

        # Option for plotting simulated data line
        SIM_LINES: 'color="r", xdata=self.SIM.x*1e6'

        # Options for figures that will be generated
        SIMULATION_FIGURE: >
            xlabel('time [' + unichr(181) + 's]', size=
               'small');
            ylabel('I [A]', size='small');
            legend(['measured', 'simulated'], loc='
               lower right', prop=FontProperties(size='
               small'));

        # Option for plotting error lines
        ABSERR_LINE: 'color="k", xdata=AbsErr.x*1e6'
        RELERR_LINE: 'color="k", xdata=RelErr.x*1e6'
```

```
            # Options for absolute error figure
            ABSERR_FIGURE: >
                ylabel('Absolute error');

            # Option for relative error figure
            RELERR_FIGURE: >
                xlabel('t [' + unichr(181) + 's]');
                ylabel('Relative error');

            # Name of the file where all types of errors
               will be saved
            ERROR_FILE: errorsRR.txt

            # Errors are saved in the following order
            ORDER_OF_ERRORS:
                - S_ABS
                - SSQ_ABS
                - SSQ_ABS_L
                - SSQ_REL
                - SSQ_REL_L
                - VAR_EST
                - RMSE
                - RMSErel
                - NRMSE
                - MAX_ABSERR
                - MAX_RELERR
                - MEAN_RELERR
        #         - RelErr_Qrr
        #         - RelErr_Erec

    rr_voltage:

        title: reverse recovery voltage

        simulation:

            # Circuit file - check
               OPTIONS_FOR_READING_SIMULATION_DATA also.
            SIM_TEMPLATE: "/home/daniele/workspace/
               parameter_extraction/
               dpinth_tau_ii_idf_RR_circuit.ngspice"

            # Destination directory where files produced by
                capacitance simulations are saved.
            DEST_DIR: "/media/ACER/Users/Daniele/Documents/
               Politecnico/01_tesi_simulazioni/22
               aa_dpinth_tau_ii_idf_TCAD_mop_DC_RR_nsga2/
               rr_voltage"

            # Name of the file with parameter values for
               each simulation.
            PARAMETER_FILE: parameters.txt
```

```
        # File with iteration number
        ITER_FILE: iterRRV.txt

        # Shared libraries with diode model definition
        SHARED_LIBRARIES: "/home/daniele/workspace/
            parameter_extraction/libdpinth_tau_ii_idf.so
            "

        # Basename of the file where simulation data
            are saved
        SIMULATION_FILE_BASENAME: simulationRRV

        # Command line options for starting ngspice
        NGSPICE_COMMAND_LINE_OPTIONS: ""

    fitting:

        # Measured data array
        MEAS_DATA: !!python/object/apply:numpy.loadtxt
                   args: ["/media/ACER/Users/Daniele/
                       Documents/ABB/02_measurement_data
                       /02_TCAD_DATA/RR_I_V_int.txt"]
                   kwds: {skiprows: 0, comments: "#"}

        # SSQ class used for error calculation
        SSQ: DiodeRR_V

        # Map variables to column indices in simulation
            data matrix.
        # Moreover, according to conventions used in
            circuit files, vector signs have or do not
            have to be inverted.
        # If sign = 1, there is no need for inversion.
            Otherwise set sign = -1
        OPTIONS_FOR_READING_SIMULATION_DATA:

            time:
                column_index: 0

            current:
                column_index: 1
                sign: -1

            voltage:
                column_index: 2
                sign: -1

        # Scale for waveforms. Reverse recovery current
            is divided by SCALING_OF_WAVEFORM
        # before error is computed.
        SCALING_OF_WAVEFORM: *VCC
```

```
            # Scaling factor to express time values from
               data files in seconds
            SCALING_FACTOR: !!float 1e-6

            # Independent variable bounds for error
               calculation. Use base SI units.
            # Use !!null tag if there is no lower/upper
               bound.
            BOUNDS: [ !!null , !!float 12e-6 ]

            # If ERROR_WRT_YLOG is True, measured and
               simulated data have to be
            # log-transformed before error computation.
            ERROR_WRT_YLOG: 0

            # Threshold used to decide when absolute error
               has to be used instead of relative error in
               SSQ_ABS_REL computation
            RELERR_th: !!float 10

        optimization:

            # Error to be extracted
            # If you change this, change also ['multiobjopt
               ']['characteristics'][*]['optimization']['
               DEFAULT_ERR']
            value: SSQ_ABS

            # Error returned by the optimizer if an
               exception occurs.
            # Choose DEFAULT_ERR value according to the
               error estimator specified in ['multiobjopt
               ']['characteristics'][*]['optimization']['
               value']
            # If ['multiobjopt']['characteristics'][*]['
               optimization']['value'] is equal to '
               ManyObjective', DEFAULT_ERR
            # must be a list with as many error values as
               the parameters specified in ['multiobjopt
               ']['characteristics'][*]['optimization']['
               OBJECTIVE_LIST'].
            DEFAULT_ERR: [!!float 100]

            # List of objective functions extracted from
               current characteristic.
            OBJECTIVE_LIST: [SSQ_ABS]

            # Needed by R-NSGA-II algorithm
            # There must be a key entry for each element in
                'OBJECTIVE_LIST'.
            REFERENCE_POINTS:
                SSQ_ABS: [!!float 1e-2, 2, 0.07]
```

```
        # This weight list can be used in different
           ways:
        # - to build a weighted sum of the values
           specified in 'OBJECTIVE_LIST' if a weighted
           sum approach
        #   should be used for a many objective problem
        # - to bias some objectives of 'OBJECTIVE_LIST'
            more than others in the reference point
           method by Deb et all.
        # - ...
        WEIGHT_LIST: [1]

    options:

        # Option for plotting measured data line
        MEAS_LINES: 'color="b", xdata=self.MEAS.x*1e6,
           marker=".", markersize=2'

        # Option for plotting simulated data line
        SIM_LINES: 'color="r", xdata=self.SIM.x*1e6'

        # Options for figures that will be generated
        SIMULATION_FIGURE: >
            xlabel('time [' + unichr(181) + 's]', size=
               'small');
            ylabel('V [V]', size='small');
            legend(['measured', 'simulated'], loc='
               lower right', prop=FontProperties(size='
               small'));

        # Option for plotting error lines
        ABSERR_LINE: 'color="k", xdata=AbsErr.x*1e6'
        RELERR_LINE: 'color="k", xdata=RelErr.x*1e6'

        # Options for absolute error figure
        ABSERR_FIGURE: >
            ylabel('Absolute error');

        # Option for relative error figure
        RELERR_FIGURE: >
            xlabel('t [' + unichr(181) + 's]');
            ylabel('Relative error');

        # Name of the file where all types of errors
           will be saved
        ERROR_FILE: errorsRR.txt

        # Errors are saved in the following order
        ORDER_OF_ERRORS:
            - S_ABS
            - SSQ_ABS
            - SSQ_ABS_L
            - SSQ_REL
```

```
                    - SSQ_REL_L
                    - VAR_EST
                    - RMSE
                    - RMSErel
                    - NRMSE
                    - MAX_ABSERR
                    - MAX_RELERR
                    - MEAN_RELERR
            #         - RelErr_Qrr
            #         - RelErr_Erec

    # Options needed by the multi-objective evolutionary
       algorithm
    moea_options:

        ## Mandatory options

        # Population size
        pop_size: 120 #200 #300

        # Maximum number of generations
        max_generations: 250 #400 #500

        # Number of subgroups in which the list of new
           candidates should be divided
        # before evaluating candidates.
        nb_of_subgroups: 5

        # Parameter used in reference-point-based NSGA-II
           procedure.
        epsilon_for_niching: !!float 1e-3

        # File with seeds
        seed_file: "/media/ACER/Users/Daniele/Documents/
           Pollitecnico/01_tesi_simulazioni/random_pop.txt"

        ## Non-mandatory options

        # The rate at which crossover is performed
        crossover_rate: 0.9

        # The rate at which mutation is performed
        mutation_rate: 0.08

        # The non-negative crossover distribution index
        distribution_index_for_crossover: 10

        # The non-negative mutation distribution index
        distribution_index_for_mutation: 20

        # Minimum distance between coordinates to perform
           crossover
        EPS: !!float 1e-14
```

```
options:

    # Destination directory with multi objective optimization
       results
    ROOT_DIR: "/media/ACER/Users/Daniele/Documents/Pollitecnico
       /01_tesi_simulazioni/22
       aa_dpinth_tau_ii_idf_TCAD_mop_DC_RR_nsga2"

    # If SIMULATION_SAVE is True, simulation data will be saved
        to files
    SIMULATION_SAVE: 0

    # Directory where wrong simulation files and an exception
       report are saved
    FAILURES_DIRNAME: "/media/ACER/Users/Daniele/Documents/
       Pollitecnico/01_tesi_simulazioni/bctm_diode_failures"

#    # File where exceptions are reported
#    FAILURES_FILE: failures.log

    # Flag used to choose if errors should be saved or not
    ERROR_SAVE: 0

    # Basename of the file with generational statistics of the
       population throughout the run
    STATISTICS_FILE: statistics.txt

    # Basename of the file with every individual during each
       generation of the run
    INDIVIDUALS_FILE: individuals.txt

    # File with the final Pareto frontier
    FINAL_PARETO_FRONTIER: final_pareto_frontier.txt

    # File with the Pareto frontier computed generation-wise
    GENERATION_WISE_PARETO_FRONTIER:
       generation_wise_pareto_frontier.txt

    # File where MOEA success is reported
    SUCCESS_REPORT: success.txt
```

Listing 3: A YAML file for designing a MOEA performance comparison.

```
# YAML

# Caution: spacing is important.
# For example:
# - always keep at least one blank space character after a
   colon
# - use the same indentation level for each key of a dictionary
```

```
# - always keep at least one blank space character after YAML
   tags like !!float, !!null (when it appears in a list)


# MOEAs to be compared.
#MOEA_list: [nsga2, r_nsga2, paes, spea2, spea2plus,
   mo_cma_es_p, mo_cma_es_p_with_recomb]
MOEA_list: [nsga2, r_nsga2, mo_cma_es_p,
   mo_cma_es_p_with_recomb]

# YAML files with inputs for each MOEA.
yaml_file_list:
    - "/home/daniele/workspace/parameter_extraction/
       dpinth_tau_ii_idf_tcad_mop_dc_rri_rrv_nsga2_input.txt"
    - "/home/daniele/workspace/parameter_extraction/
       dpinth_tau_ii_idf_tcad_mop_dc_rri_rrv_nsga2_input.txt"
    - "/home/daniele/workspace/parameter_extraction/
       dpinth_tau_ii_idf_tcad_mop_dc_rri_rrv_mo_cma_es_input.
       txt"
    - "/home/daniele/workspace/parameter_extraction/
       dpinth_tau_ii_idf_tcad_mop_dc_rri_rrv_mo_cma_es_input.
       txt"


# MOEA performance is evaluated w.r.t. the following
   performance indicators.
#Performance_indicators: [dom_rank, hyp, eps, r2, r3, eaf1,
   exec_time]
Performance_indicators: [hyp, eps]


# Number of different runs for each algorithm and for each
   performance indicator to be considered.
Sample_size: 30


ROOT_DIR: "/media/ACER/Users/Daniele/Documents/Pollitecnico/01
   _tesi_simulazioni/100
   _dpinth_tau_ii_idf_tcad_mop_dc_rri_rrv_moea_comparison_01"


# Relative path of the directory where initial populations are
   saved.
# The parent directory will be ROOT_DIR.
INITIAL_POPS: initial_pops


# Number of processes MOEA runs are distributed over
Number_of_processes: 2


# Relative path of the file where the list of tests to be run
   is saved.
```

```
# The parent directory will be ROOT_DIR.
List_of_runs: run_list.txt


# File with lower and upper bounds of the objective vectors
BOUNDS: bounds.txt


# Script for running a complete analysis of a single MOEA run
Script_for_post_process_of_single_moea: "/home/daniele/
    workspace/parameter_extraction/scripts/
    single_moea_performance_assessment.py"


# R script for plotting differences between the empirical
    attainment functions of two
# data set in the biobjective case.
Script_for_eafdiffplot: "/home/daniele/workspace/
    parameter_extraction/scripts/plot_eaf_differences.R"


# R script for plotting indicator boxplots
Script_for_boxplots: "/home/daniele/workspace/
    parameter_extraction/scripts/indicator_boxplot.R"


# Basename of the file with the reference set for the unary
    indicators
REFERENCE_SET: reference_set.txt


# Specify paths to PISA software tools
PISA:

    filter: "/home/daniele/Scaricati/distribution_final/
        tools_c_source/filter"

    eaf: "/home/daniele/Scaricati/distribution_final/
        attainment_c_source/eaf"

    eps_ind: "/home/daniele/Scaricati/distribution_final/
        indicators_c_source/eps_ind"

    hyp_ind: "/home/daniele/Scaricati/distribution_final/
        indicators_c_source/hyp_ind"


# Path to the multiple testing software by Garcia and Herrera
multipleTest: "/home/daniele/Scaricati/multipleTest/Friedman"
```

BIBLIOGRAPHY

[1] P. Antognetti and G. Massobrio. *Semiconductor Devices Modeling with SPICE*. McGraw-Hill, Inc., New York, NY, USA, 1990.

[2] N. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In B. McKay et al., editors, *The 2005 IEEE International Congress on Evolutionary Computation (CEC'05)*, volume 2, pages 1769–1776, 2005.

[3] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, NY, USA, 1996.

[4] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende, and W. R. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1:9–32, 1995.

[5] M. Bellini, I. Stevanovic, and D. Prada. Improved lumped charge model for high voltage power diode and automated extraction procedure. In *Proceedings of the IEEE Bipolar/BiCMOS Circuits and Technology Meeting*, 2011.

[6] G. Bergmann and G. Hommel. Improvements of general multiple test procedures for redundant systems of hypothses. *Multiple Hypotheses Testing*, pages 100–115, 1988.

[7] A. T. Bryant, X. Kang, E. Santi, P. R. Palmer, and J. L. Hudgins. The use of a formal optimization procedure in automatic parameter extraction of power semiconductor devices. In *Power Electronics Specialist Conference*, volume 2, pages 822–827, 2003.

[8] A. T. Bryant, X. Kang, E. Santi, P. R. Palmer, and J. L. Hudgins. Two-step parameter extraction procedure with formal optimization for physics-based circuit simulator IGBT and p-i-n diode models. *IEEE Transactions on power electronics*, 21(2):295–309, March 2006.

[9] I. Budihardjo and P. O. Lauritzen. The lumped-charge power mosfet model, including parameter extraction. In *IEEE Transactions on Power Electronics*, volume 10, pages 379–387, 1995.

[10] A. G. Chynoweth. Ionization rates for electrons and holes in silicon. *Physical Review*, 109:1537–1540, 1958.

[11] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation Series. Springer, second edition, 2007.

[12] The Designer's Guide community. Verilog-A/MS, 2012. URL http://www.designers-guide.org/VerilogAMS/.

[13] The gplEDA community. GPL electronic design automation tools, 2012. URL http://www.gpleda.org/.

[14] The Ngspice community. Ngspice circuit simulator, 2012. URL http://ngspice.sourceforge.net/.

[15] J. W. Conover. *Practical nonparametric statistics*. John Wiley and Sons, New York, NY, USA, 3 edition, 1999.

[16] ©SIMetrix Technologies. *Simulator Reference Manual*, March 2011.

[17] ©SIMetrix Technologies. Analog, mixed signal circuit simulation software tool, SIMetrix, SIMPLIS, Micron VX, DVM, 2012. URL http://www.simetrix.co.uk/index.html.

[18] D. W. Corne and J. D. Knowles. Techniques for highly multiobjective optimization: Some nondominated points are better than others. In *Proc. of 2007 Genetic and Evolutionary Computation Conference*, pages 773–780, 2007.

[19] V. Grunert da Fonseca, C. M. Fonseca, and A. O. Hall. Inferential performance assessment of stochastic optimizers and the attainment function. In *Lectures Notes in Computer Science 1993: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization - EMO 2001*, pages 213–225. Springer-Verlag, 2001.

[20] L. Davis. *Genetic Algorithms and Simulated Annealing*. Morgan Kaufman, Los Altos, CA, 1987.

[21] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. Technical report, Indian Institute of Technology Kanpur, 1994.

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

[23] K. Deb and D. K. Saxena. On finding pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. KanGAL report number 2005011, Indian Institute of Technology Kanpur, 2005.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6: 182–197, 2000.

[25] K. Deb, J. Sundar, U. B. Rao, and S. Chaudhuri. Reference point based multi-objective optimization using evolutionary algorithms. *International Journal of Computational Intelligence Research*, 2(3):273–286, 2006.

[26] K. Deep and M. Thakur. A new crossover operator for real coded genetic algorithms. *Applied Mathematics and Computation*, 188:895–911, 2007.

[27] K. Deep and M. Thakur. A new mutation operator for real coded genetic algorithms. *Applied Mathematics and Computation*, 193:211–230, 2007.

[28] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[29] M. Ehrgott. Location of rescue helicopters in south tyrol. *International Journal of Industrial Engineering*, 9:16–22, 2002.

[30] M. Ehrgott. *Multicriteria Optimization*. Springer, 2nd edition, 2005.

[31] M. Ehrgott and S. Ruzika. Improved ε-Constraint method for multiobjective programming. *Journal of Optimization Theory and Applications*, 138(1), 2008.

[32] M. Ehrgott and D. Ryan. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, 11(3):139–150, 2002.

[33] M. Ehrgott and M. Wiecek. Multiobjective programming. In J. Figueira, S. Greco, and M. Ehrgott, editors, *Multicriteria Decision Analysis: State of the Art Surveys*, pages 667–722. Kluwer Academic Publishers, 2005.

[34] M. Emmerich, N. Beume, and B. Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410, pages 62–76. Springer-Verlag, 2005.

[35] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe. Many-objective optimization: An engineering design perspective. In *Lecture Notes in Computer Science 3410: Evolutionary Multi-Criterion Optimization - EMO 2005*, pages 14–32. Springer, 2005.

[36] C. M. Fonseca, V. Grunert da Fonseca, and L. Paquete. Exploring the performance of stochastic multiobjective optimizers with the second-order attainment function. In *Lectures Notes in Computer Science 3410: Third International Conference on Evolutionary Multi-Criterion Optimization - EMO 2005*, pages 250–264. Springer, 2005.

[37] The Python Software Foundation. Python/C API reference manual, 2012. URL http://docs.python.org/2/c-api/.

[38] The Python Software Foundation. Extending and embedding the python interpreter, 2012. URL http://docs.python.org/2/extending/index.html{#}extending-index.

[39] N. Galster, M. Frecker, E. Carroll, J. Vobecky, and P.Hazdra. Application-specific fast-recovery diode: Design and performance. In *PCIM*, April 1998.

[40] S. García and F. Herrera. An extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[41] J. M. Gareth. Curve alignment by moments. *Annals of Applied Statistics*, 1(2): 480–501, 2007.

[42] Aaron Garrett. inspyred: Bio-inspired algorithms in python, 2012. URL http://pypi.python.org/pypi/inspyred/1.0.

[43] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, first edition, 1989.

[44] The ABB Group. The abb group - automation and power technologies, 2012. URL http://www.abb.com/.

[45] N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, pages 2389–2395. ACM, July 2009.

[46] N. Hansen. The CMA Evolution Strategy, 2012. URL http://www.lri.fr/~hansen/cmaesintro.html.

[47] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In X. Yao et al., editors, *Parallel Problem Solving from Nature PPSN VIII*, volume 3242 of *LNCS*, pages 282–291. Springer, 2004.

[48] Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial*, June 2011.

[49] Y. Hochberg. A sharper Bonferroni procedure for multiple tests significance. *Biometrika*, 75:800–802, 1988.

[50] G. Hommel. A stagewise rejective multiple test procedure. *Biometrika*, 75: 383–386, 1988.

[51] G. Hommel and G. Bernhard. A rapid algorithm and a computer program for multiple test procedures using logical structures of hypotheses. *Computer Methods and Programs in Biomedicine*, 43:213–216, 1994.

[52] Z. Hossain et al. A physics-based MCT model using the lumped-charge modeling technique. In *IEEE Power Electronics Specialists Conf.*, pages 23–28, 1996.

[53] C. Igel, N. Hansen, and S. Roth. Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[54] R. L. Iman and J. M. Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics*, pages 571–595, 1980.

[55] Ruhr-Universität Bochum Institut für Neuroinformatik. Shark machine learning library, 2012. URL http://shark-project.sourceforge.net/index.html.

[56] H. Ishibuchi and Y. Nojima. Optimization of scalarizing functions through evolutionary multiobjective optimization. In *Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007*, pages 51–65. Springer, 2007.

[57] H. Ishibuchi, T. Doi, and Y. Nojima. Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms. In *Lecture Notes in Computer Science 4193: Parallel Problem Solving from Nature - PPSN IX*, pages 493–502. Springer, 2006.

[58] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *Proc. of 2008 IEEE Congress on Evolutionary Computation*, pages 2424–2431, 2008.

[59] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis, 6/E*. Pearson, New York, NY, USA, 2008.

[60] M. Keser and K. Joardar. Genetic algorithm based mosfet model parameter extraction. *Technical Proceedings of the 2000 International Conference on Modeling and Simulation of Microsystems*, pages 341–344, 2000.

[61] V. Khara, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *Lecture Notes in Computer Science 2632: Evolutionary Multi-Criterion Optimization - EMO 2003*, pages 367–390. Springer, 2003.

[62] M. Kim, T. Hiroyasu, M. Miki, and S. Watanabe. SPEA2+: Improving the performance of the strength pareto evolutionary algorithm 2. In *Lecture Notes in Computer Science, Vol. 3242 (Proc. of PPSN VIII)*, pages 742–751. Springer, 2004.

[63] J. D. Knowles and D. W. Corne. The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. In *Congress on Evolutionary Computation*, pages 98–105. IEEE Service Center, 1999.

[64] J. D. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic. TIK-report no. 214, ETH Zürich, Computer Engineering and Networks Laboratory, February 2006.

[65] M. Köppen and K. Yoshida. Substitute distance assignments in NSGA-II for handling many-objective optimization problems. In *Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007*, pages 727–741. Springer, 2007.

[66] R. Kraus, K. Hoffmann, and H. J. Mattausch. A precise model for the transient characteristics of power diodes. In *Conf. Rec. IEEE PESC '92*, pages 863–869, 1992.

[67] H. P. Langtangen. *Python Scripting for Computational Science*. Springer, 3rd edition, 2009.

[68] P. O. Lauritzen and C. L. Ma. A simple diode model with reverse recovery. *IEEE Transactions on Power Electronics*, 6(2), April 1991.

[69] P. O. Lauritzen, G. K. Andersen, and M. Helsper. A basic igbt model with easy parameter extraction. In *IEEE Proc. of Power Electronics Specialists Conf.*, 2001.

[70] J. G. Linvill and J. F. Gibbons. *Transistors and Active Circuits*. McGraw-Hill, 1962.

[71] J. Lutz, H. Schlangenotto, U. Scheuermann, and R. De Doncker. *Semoconductor Power Devices*. Springer, 2011.

[72] C. L. Ma, P. O. Lauritzen, and P. Y. Lin. A physically-based lumped charge p-ν-n diode model. In *Proc. European Power Electronics Conf.*, pages 23–28, Brighton, England, 1993.

[73] C. L. Ma, P. O. Lauritzen, and J. Sigg. A physics-based gto model for circuit simulation. In *IEEE Power Electronics Specilists Conf.*, pages 872–878, 1995.

[74] C. L. Ma, P. O. Lauritzen, and J. Sigg. Modeling of high-power thyristors using the lumped-charge modeling technique. In *6^{th} European Conf. on Power Electronics and Applications*, 1995.

[75] C. L. Ma, P. O. Lauritzen, and J. Sigg. Modeling of power diodes with the lumped-charge modeling technique. *IEEE Transactions on PowerElectronics*, 12 (3):398–405, May 1997.

[76] C. L. Ma et al. A systematic approach to modeling power semiconductor devices based on charge control principles. In *Proceedings of IEEE Power Electronics Specialists Conference*, pages 31–37, 1994.

[77] N. Mohan, T. M. Undeland, and W. P. Robbins. *Power electronics: converters, applications, and design*. John Wiley & Sons, 2003.

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

[79] J. Nocedal and S. Wright. *Numerical Optimization, Second Edition*. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[80] R. Van Overstraeten and H. De Man. Measurement of the ionization rates in diffused silicon p-n junctions. *Solid-State Electronics*, 13:583–608, 1970.

[81] V. Pareto. *Manuel d'économie politique*. F. Rouge, 1896.

[82] D. Prada, M. Bellini, I. Stevanovic, J. Victory, and J. Vobecky. Parameter extraction procedure with multiobjective optimization for high voltage power diode. preprint.

[83] R. C. Purshouse and P. J. Fleming. Evolutionary many-objective optimization: An exploratory analysis. In *Proc. of 2003 IEEE Congress on Evolutionary Computation*, pages 2066–2073, 2003.

[84] D. M. Rom. A sequentially rejective test procedure based on a modified Bonferroni inequality. *Biometrika*, 77:663–665, 1990.

[85] H. Sato, H. E. Aguirre, and K. Tanaka. Controlling dominance area of solutions and its impact on the performance of MOEAs. In *Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007*, pages 5–20. Springer, 2007.

[86] J. P. Shaffer. Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826–831, 1986.

[87] K. Sheng, W. Williams, and S. J. Finney. A review of igbt models. *IEEE Transactions on Power Electronics*, 15:1250–1266, November 2000.

[88] A. Sülflow, N. Drechsler, and R. Drechsler. Robust multi-objective optimization in high dimensional spaces. In *Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007*, pages 715–726. Springer, 2007.

[89] S. M. Sze and Kwok K. Ng. *Physics of Semiconductor Devices*. John Wiley and Sons, 2007.

[90] N. Talwalkar, P. O. Lauritzen, B. Fatemizadeh, D. Perlman, and C. L. Ma. A power bjt model for circuit simulation. In *IEEE Power Electronics Specialists Conference Record.*, volume 1, pages 50–55, 1996.

[91] Cher Ming Tan and King-Jet Tseng. Using power diode models for circuit simulations-a comprehensive review. *IEEE Transactions on Industrial Electronics*, 46:637–645, 1999.

[92] Sentinel IC Technologies. Abb power diode model and extraction analysis. Confidential and proprietary report, ABB Switzerland Ltd and Sentinel IC Technologies, November 2011.

[93] T. Voß, N. Hansen, and C. Igel. Recombination for learning strategy parameters in the MO-CMA-ES. In *Fifth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2009)*, pages 155–168. Springer-Verlag, 2009.

[94] T. Voß, N. Hansen, and C. Igel. Improved step size adaptation for the MO-CMA-ES. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 487–494. ACM, 2010.

[95] E. Weber et al. Lexicographic optimisation for water resources planning: The case of lake Verbano, Italy. In A. Rizzoli and A. Jakeman, editors, *Integrated Assessment and Decision Support – Proceedings of the First Biennial Meeting of the International Environmental Modelling and Software Society*, pages 235–240, June 2002.

[96] E. J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.

[97] L. While. A new analysis of the LebMeasure algorithm for calculating hypervolume. In *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410, pages 326–340. Springer-Verlag, 2005.

[98] Wikipedia. Flexible ac transmission system, 2011. URL http://en.wikipedia.org/wiki/Flexible_AC_transmission_system.

[99] Wikipedia. High-voltage direct current, 2011. URL http://en.wikipedia.org/wiki/High-voltage_direct_current.

[100] Wikipedia. Adjustable-speed drives, 2011. URL http://en.wikipedia.org/wiki/Variable_speed_drives.

[101] Wikipedia. Hardware description language, 2012. URL http://en.wikipedia.org/wiki/Hardware_description_language.

[102] Wikipedia. Curse of dimensionality, 2012. URL http://http://en.wikipedia.org/wiki/Curse_of_dimensionality.

[103] D. H. Wolpert and W. G. Macready. No free lunch theorem for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.

[104] X.Kang, A. Caiafa, E. Santi, J.L. Hudgins, and P.R. Palmer. Parameter extraction for a power diode circuit simulator model including temperature dependent effects. In *Applied Power Electronics Conference and Exposition*, volume 1, pages 452–458, 2002.

[105] K. Yamanishi, Jun-Ichi Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, 2004.

[106] A. T. Yang, Y. Liu, and J. T. Yao. An efficient nonquasistatic diode model for circuit simulation. *IEEE Trans. Computer-Aided Design*, 13:231–239, February 1994.

[107] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. In *IEEE Transactions on Evolutionary Computation*, volume 3, pages 257–271, 1999.

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

[109] 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*, pages 95–100, 2002.

[110] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.