

SOLVING COMPUTATIONALLY EXPENSIVE PROBLEMS USING
SURROGATE-ASSISTED OPTIMIZATION: METHODS AND APPLICATIONS

By

Julian Blank

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2022

PUBLIC ABSTRACT

SOLVING COMPUTATIONALLY EXPENSIVE PROBLEMS USING SURROGATE-ASSISTED OPTIMIZATION: METHODS AND APPLICATIONS

By

Julian Blank

Optimization is omnipresent in many research areas and has become a critical component across industries. However, while researchers often focus on a theoretical analysis or convergence proof of an optimization algorithm, practitioners face various other challenges in real-world applications. This thesis focuses on one of the biggest challenges when applying optimization in practice: computational expense, often caused by the necessity of calling a third-party software package. To address the time-consuming evaluation, we propose a generalizable probabilistic surrogate-assisted framework that dynamically incorporates predictions of approximation models. Besides the framework's capability of handling multiple objectives and constraints simultaneously, the novelty is its applicability to all kinds of metaheuristics. Moreover, often multiple disciplines are involved in optimization, resulting in different types of software packages utilized for performance assessment. Therefore, the resulting optimization problem typically consists of multiple independently evaluable objectives and constraints with varying computational expenses. Besides providing a taxonomy describing different ways of independent evaluation calls, this thesis also proposes a methodology to handle inexpensive constraints with expensive objective functions and a more generic concept for any type of heterogeneously expensive optimization problem. Furthermore, two case studies of real-world optimization problems from the automobile industry are discussed, a blueprint for solving optimization problems in practice is provided, and a widely-used optimization framework focusing on multi-objective optimization (founded and maintained by the author of this thesis) is presented. Altogether, this thesis shall pave the way to solve (computationally expensive) real-world optimization more efficiently and bridge the gap between theory and practice.

ABSTRACT

SOLVING COMPUTATIONALLY EXPENSIVE PROBLEMS USING SURROGATE-ASSISTED OPTIMIZATION: METHODS AND APPLICATIONS

By

Julian Blank

Significant effort has been made to solve computationally expensive optimization problems in the last two decades. To address the expense of objectives and constraint functions, the usage of surrogate models during optimization has emerged as a predominant approach. Numerous surrogate-based algorithms have been proposed, each answering what to model, what to model it with, and how to utilize the additional information in different ways. However, many optimization methods are tailored to a specific problem or make impractical assumptions, such as having only a single objective, not having any constraints to satisfy, or all objectives and constraints being equally time-consuming. Such assumptions make it challenging for practitioners to find suitable optimization algorithms to apply to their applications and show the need for more generalized methods as well as best practices for solving computationally expensive applications. Thus, this dissertation addresses computationally expensive optimization problems in a holistic manner, including their interdisciplinary character and practicalities. It discusses different viewpoints on computationally expensive optimization and demonstrates relevance by providing an overview of applications in various research fields. Moreover, it proposes a generalizable surrogate-assisted framework for solving computationally expensive problems. The framework endows an existing optimization algorithm fulfilling minimal requirements with surrogate assistance, improving the convergence behavior. The algorithm's search pattern on the computationally inexpensive surrogate is used to determine the subsequent designs to run for the time-consuming simulation. Moreover, this thesis investigates an often overlooked fact that objectives and constraints may be independently computable in practice. Independent target functions result in a heterogeneously expensive optimization problem with some objectives or constraints being less and some more time-consuming than others. A typical scenario of heterogeneity is objectives requiring a time-consuming simula-

tion but computationally inexpensive geometrical or physical constraints. For such cases, this thesis proposes an optimization method exploiting the inexpensiveness of constraints during optimization. Furthermore, the approach is generalized for any type of heterogeneity, requiring addressing partial information on a solution level. Because only little attention has been paid to such a fundamentally important aspect of expensive optimization, the proposed method needs a novel way of treating missing information during optimization. The evaluation order of targets is determined based on an information gain sorting taking the trade-off between prediction error and evaluation time into account. The proposed method then iterates over targets with more information gain first and discards solutions early on during the evaluation process without evaluating them entirely. Besides algorithmic aspects of optimizing time-consuming functions, this dissertation also addresses more practical matters. It presents the architecture and usage of the multi-objective optimization framework pymoo (founded and maintained by the author of this thesis), a widely used and well-known toolkit across academia and industry. While some end-users directly employ the provided state-of-the-art algorithms, others utilize rapid prototyping capabilities or other features such as multi-criteria decision-making and visualization. Moreover, this thesis portrays optimization as a product of a collaboration between two or more disciplines. The interdisciplinary characteristics shall be a substantial part of the problem-solving procedure. Thus the proposal of a blueprint for *collaborative* optimization provides some guidance and best practices for practitioners. Altogether, the optimization of time-consuming functions is looked at from different angles. Practitioners who face time-consuming real-world problems can use the proposed methods for solving various types of optimization problems. Thus, this thesis shall pave the way to solve (computationally expensive) real-world optimization more efficiently and bridge the gap between theory and practice.

ACKNOWLEDGEMENTS

First and foremost, I am highly grateful to my supervisor, Kalyanmoy Deb, for his continuous support, valuable advice, and patience during my Ph.D. study. His knowledge and experience have encouraged me in all the time of my academic research and daily life. Moreover, I would like to thank my Ph.D. committee, Erik Goodman, Sandeep Kulkarni, and Ronald Averill, for their valuable feedback and comments throughout my academic journey. I would also like to thank all my co-workers and fellow students for their steady support in and outside the laboratory and classroom; especially Yashesh Dhebar for the fruitful discussions, Zichao Lu for the steady feedback concerning my optimization framework, Ritam Guha for the accompany on my lunch walks, Proteek Chandan Roy for being my patient desk neighbor, Rayan Hussein for his advice and reliability at any time, and Yash Vesikar for the great experience working together on a real-world problem. Also, I would like to thank Nicholle Robertson for reviewing my thesis for language and grammar.

Additionally, I would like to express my gratitude to my whole family and relatives, especially my mother, Barbara Blank, for the emotional and financial support, my son, Emil Babilon, for being the most beautiful enrichment of my life, my grandmother, Hannelore Blank, for the steady support, my grandfather Rudi Blank, for teaching me many useful lessons for succeeding in life, and my uncle, Holger Blank, for mentoring me during my stay. Moreover, I would like to thank all my friends in the Lansing area, especially Justin Miller and his children Elinor and Cora Miller, his dad Bill Miller, and Carolyn Jehners, for the family-like environment during my stay and the adventures we have had together. Finally, I would like to thank everyone I have been in touch with within the last few years and who accompanied me on my journey in any way.

TABLE OF CONTENTS

LIST OF TABLES	ix
LIST OF FIGURES	xi
LIST OF ALGORITHMS	xiv
KEY TO ABBREVIATIONS	xv
CHAPTER 1 INTRODUCTION	1
1.1 Motivation	1
1.1.1 Problem Characteristics and Challenges	2
1.1.2 Facing a Computationally Expensive Problem (CEP)	5
1.1.3 The Role of Surrogate Model in Optimization and Some Terminology	6
1.2 Research Goals and Contributions	10
1.3 Structure of the Thesis	13
CHAPTER 2 FUNDAMENTALS	14
2.1 Single-Objective Optimization (SOO)	14
2.2 Multi-Objective Optimization (MOO)	15
2.3 Genetic Algorithms (GAs)	17
2.4 Data Modeling and Predictors	19
2.4.1 Polynomial Regression (PR)	19
2.4.2 Radial Basis Functions (RBFs)	21
2.4.3 Kriging	22
2.5 Efficient Global Optimization (EGO)	23
2.6 Summary of the Chapter	24
CHAPTER 3 LITERATURE REVIEW	25
3.1 Different Viewpoints on Computationally Expensive Optimization	25
3.2 Surrogate-Based Optimization: History and Recent Trends	27
3.3 Relevant Literature and Applications	30
3.4 Summary of the Chapter and Open Issues	43
CHAPTER 4 A GENERALIZED PROBABILISTIC SURROGATE-ASSISTED FRAME- WORK	44
4.1 Introduction	44
4.2 Background	47
4.3 Interfacing Metaheuristics	51
4.4 Probabilistic Surrogate-Assisted Framework (PSAF)	53
4.4.1 Methodology	53
4.4.1.1 Influence of Surrogate through Tournament Selection Pressure (α)	55
4.4.1.2 Continue Optimization on Surrogate (β)	56

4.4.1.3	Balancing the Utilization of Surrogate (ρ)	58
4.4.1.4	Surrogate Management	59
4.4.2	Experimental Results	60
4.4.2.1	What are suitable values for (α , β and ρ)?	60
4.4.2.2	Is it beneficial to update ρ each iteration?	63
4.4.2.3	How does PSAF perform compared to other surrogate-based algorithms?	63
4.4.3	Summary of Section 4.4	65
4.5	Generalized Probabilistic Surrogate-Assisted Framework (GPASF)	66
4.5.1	Methodology	66
4.5.1.1	Generalized α and β -phase	67
4.5.1.2	Balancing the Exploration and Exploitation (γ)	70
4.5.1.3	Surrogate Management	71
4.5.2	Experimental Results	72
4.5.2.1	(Unconstrained) Single-Objective Optimization	74
4.5.2.2	Constrained Single-Objective Optimization	75
4.5.2.3	(Unconstrained) Multi-Objective Optimization	77
4.5.2.4	Constrained Multi-Objective Optimization	79
4.5.3	Summary of Section 4.5	80
4.6	Summary of the Chapter	81
CHAPTER 5 HETEROGENEOUS EXPENSIVE OBJECTIVES AND CONSTRAINTS		83
5.1	Introduction	83
5.2	Related Work	85
5.3	Background	89
5.3.1	Evaluations Jobs	90
5.3.2	Job Scheduling	91
5.4	Computationally Expensive Objectives and Inexpensive Constraints	93
5.4.1	Design of Experiments	94
5.4.2	Methodology	96
5.4.3	Results	99
5.4.4	Summary of Section 5.4	103
5.5	Constrained Multi-Objective Optimization Problems With Heterogeneous Evaluation Times	103
5.5.1	How to Exploit Heterogeneity of an Optimization Problem?	105
5.5.2	Methodology	107
5.5.2.1	Survival Under Uncertainty	107
5.5.2.2	Probabilistic Surrogate-Guided Mating	109
5.5.2.3	Heterogeneously Expensive Evolutionary Algorithm (HE-EA)	110
5.5.2.4	Surrogate Management	115
5.5.3	Results and Discussions	115
5.5.4	Summary of Section 5.5	122
5.6	Summary of the Chapter	122
CHAPTER 6 REAL-WORLD APPLICATIONS		124

6.1	Case Study I: Cylinder Head Water Jacket Optimization	124
6.1.1	Introduction	124
6.1.2	Proposed Proximity-Based Surrogate-Assisted Optimization Method	126
6.1.2.1	Selection of Initial High-Fidelity Solutions	126
6.1.2.2	Parallel Infill Strategy	127
6.1.2.3	Management of the Surrogate Model	130
6.1.2.4	Optimization of the Approximate Function(s)	132
6.1.2.5	Flowchart	133
6.1.3	Descriptive Experiments	133
6.1.3.1	Effect of R_{prox}	135
6.1.3.2	Effect of N_{cycle}	137
6.1.3.3	Effect of r_{R2R}	138
6.1.4	Numerical Comparison	139
6.1.5	Application: Cylinder Head Water Jacket Design	144
6.1.5.1	Problem Description	144
6.1.5.2	Results	144
6.1.6	Summary of Section 6.1	148
6.2	Case Study II: Electric Machine Design	148
6.2.1	Introduction	148
6.2.2	Electric Machine Design and Optimization Problem Formulation	151
6.2.2.1	Selection of Machine Topology, Objective Functions, and Eval- uation Method	152
6.2.2.2	Definition of Feasible Search Space	153
6.2.2.3	Selection of Operating Point for Optimization	153
6.2.3	Methodology	155
6.2.3.1	Repair Operator	157
6.2.3.2	Surrogate Incorporation	157
6.2.3.3	NSGA-II-WR-SA	158
6.2.4	Results and Discussion	160
6.2.4.1	Analysis of Constraints	160
6.2.4.2	Impact of Repair Operator	161
6.2.4.3	Parameter Study for Surrogate-Assisted Optimization	162
6.2.4.4	Convergence Analysis with and without Surrogates	166
6.2.4.5	Analysis of Pareto-optimal Solutions	168
6.2.4.6	Selection of Preferred Solutions	168
6.2.5	Summary of Section 6.2	173
6.3	Summary of the Chapter	174
CHAPTER 7 OPTIMIZATION IN PRACTICE		175
7.1	Collaborative Optimization	175
7.1.1	Introduction	175
7.1.2	Related Work	177
7.1.3	SOLVeR: Collaborative Optimization	178
7.1.4	Case Studies	185
7.1.5	Summary of Section 7.1	187

7.2	pymoo: Multi-Objective Optimization in Python	188
7.2.1	Introduction	189
7.2.2	Related Work	190
7.2.3	Architecture	192
7.2.4	Problems	194
7.2.4.1	Implementations	194
7.2.4.2	Parallelization	194
7.2.5	Optimization Module	197
7.2.5.1	Algorithms	197
7.2.5.2	Operators	197
7.2.5.3	Termination Criterion	199
7.2.5.4	Decomposition	200
7.2.6	Analytics	201
7.2.6.1	Performance Indicators	201
7.2.6.2	Visualization	202
7.2.6.3	Decision Making	204
7.2.7	Summary of Section 7.2	206
7.3	Summary of the Chapter	207
CHAPTER 8 CONCLUSIONS AND FUTURE WORK		208
8.1	Summary of Contributions	208
8.2	Future Directions	209
APPENDIX		212
BIBLIOGRAPHY		214

LIST OF TABLES

Table 1.1: Terminology of different aspects using surrogates.	7
Table 3.1: Overview of surrogate-assisted optimization being used in different kind of applications.	41
Table 4.1: Categorization regarding the surrogate’s role in an algorithm.	50
Table 4.2: Rankings of best performing hyper-parameters.	63
Table 4.3: Ranking with adaptive ρ	63
Table 4.4: A comparison of DE, GA, PSO, and CMAES with their GPSAF variants on unconstrained single-objective problems with four other surrogate-assisted algorithms. The rank of the best performing algorithm in each group is shown in bold. The overall best performing algorithm for each problem is highlighted with a gray shade.	75
Table 4.5: A comparison of DE, GA, PSO, and ISRES with their GPSAF variants on constrained single-objective problems with SACOBRA – the current state-of-art algorithms for constrained optimization.	76
Table 4.6: A comparison of NSGA-II, SMS-EMOA, and SPEA2 with their GPSAF variants with four surrogate-assisted algorithms on bi-objective optimization problems.	78
Table 4.7: A comparison of NSGA-III, SMS-EMOA, and SPEA2 with their GPSAF variants with four surrogate-assisted algorithms on three-objective optimization problems.	78
Table 4.8: A comparison of NSGA-III, SMS-EMOA, and SPEA2 with their GPSAF variants with four surrogate-assisted algorithms on constrained multi-objective optimization problems.	80
Table 5.1: The median normalized Inverted Generational Distance (IGD) values out of 11 runs for NSGA-II, SA-NSGA-II and IC-SA-NSGA-II on constrained bi-objective optimization problems. The best performing method and other statistically similar methods are marked in bold.	101
Table 5.2: Average IGD values for unconstrained bi-objective problems from the ZDT test suite. NSGA-II does not use heterogeneous evaluation time information, hence produce identical IGD value for all different evaluation time combinations.	117

Table 5.3: Average IGD values for the constrained bi-objective problem TNK.	118
Table 5.4: Average IGD values for the constrained bi-objective problem Welded Beam. . . .	119
Table 5.5: Average IGD values for the three-objective problem DTLZ2.	120
Table 6.1: Median of the performance indicator ($f_{\text{best}} - f^*$ for single-objective problems and HVR for multi-objective problems) and the outcome of the statistical test for the performance of the tested methods for $D = 5$. α_{ref} defines the selected reference point for calculation of HRV.	142
Table 6.2: Median of the performance indicator ($f_{\text{best}} - f^*$ for single-objective problems and HVR for multi-objective problems) and the outcome of the statistical test for the performance of the tested methods for $D = 10$. α_{ref} defines the selected reference point for calculation of HRV.	142
Table 6.3: Parameters of IPM machine used for optimization.	151
Table 6.4: Values of geometric variables used for optimization.	154
Table 6.5: The constraint violation of each constraint value from g_1 to g_{10}	161
Table 6.6: Optimization setup and results. Evaluations (Evals) correspond to total functional evaluations performed in five runs. The reported hypervolume is calculated after normalization of objective functions.	162
Table 6.7: Complete setup for analyzing impact of hyperparameters on performance of surrogate-assisted optimization. For all cases, number of functional evaluations is limited to 200 in a single run. Each case is repeated 5 times, thus, making total evaluations 1,000 for each case.	164
Table 6.8: Results for Setups A, B, and C defined for analyzing impact of hyperparameters on performance of surrogates. Hypervolume (HV) is calculated after normalization of objective functions.	164
Table 6.9: Performance comparison of five preferred solutions found using domain specific a posteriori MCDM method and trade-off analysis. Preferred values are highlighted in bold for the five solutions.	171
Table 7.1: Multi-objective optimization frameworks in Python.	192
Table 7.2: Multi-objective optimization test problems.	195
Table 7.3: Multi-objective optimization algorithms.	198

LIST OF FIGURES

Figure 1.1: Characteristics of optimization problems.	5
Figure 1.2: The relation between optimization and machine learning.	8
Figure 2.1: The design and objective spaces during optimization.	16
Figure 2.2: Flowchart of a genetic algorithm.	18
Figure 3.1: An interdisciplinary research area with different terminologies based on the perspective: goal, method, and problem.	26
Figure 3.2: Overview of the overall percentage of each viewpoint.	27
Figure 3.3: Analysis of the literature regarding publications related to surrogate-assisted optimization from 1995 to 2021.	28
Figure 3.4: Overview of surrogate-assisted and related publications at <i>The Genetic and Evolutionary Computation Conference (GECCO)</i> from 2005 to 2020.	29
Figure 4.1: Robustly adding surrogate-assistance to population-based algorithms (illus- tration inspired by [1]).	46
Figure 4.2: Different roles of surrogates in the design of an algorithm.	49
Figure 4.3: Tournament selection with α competitors to create a surrogate-influenced infill solutions.	56
Figure 4.4: Continuation of the algorithm's run for β iteration on the surrogate.	57
Figure 4.5: Hyper-parameter Analysis for PSAF-CMA-ES with varying α , β and ρ . Shown are the baseline algorithm CMA-ES (blue), the 2nd to 10th best (orange), and the best (red). The performance $f^{(any)}$ is normalized with respect to the baseline algorithm.	62
Figure 4.6: Comparison of the average performance of PSAF with the original algorithms and other surrogate-based algorithms.	65
Figure 5.1: Strategies for the evaluation procedure considering a set of solutions X and target values V to be calculated.	90

Figure 5.2: A comparison of \cdot/B and \cdot/E strategies assuming parallel processing of all jobs, requiring different average times for evaluation.	92
Figure 5.3: Job schedule using a queue.	93
Figure 5.4: The two steps in each iteration: exploitation and exploration.	97
Figure 5.5: Sampling the design of experiments only in the feasible space using Rejection-Based Sampling (RBS), Niching Genetic Algorithm (NGA) and Energy Method.	100
Figure 5.6: Solutions in the objective space of representative runs for CTP2, CTP4, CTP8, C3DTLZ4, TNK, and OSY.	102
Figure 5.7: One iteration of HE-EA consisting of an ordered target evaluation (f_1, g_1, f_2) and offspring eliminations.	114
Figure 5.8: An illustration of the objective space for different types of unconstrained and constrained multi-objective problems. The results are based on representative run of the median performance for each problem. The different expensiveness of target functions and termination criteria are set analogously to the other experiments. The visibly better red-colored points are obtained using the proposed HE-NSGA-III procedure.	121
Figure 6.1: Adaptive trust region approach restricts the search within the cross-patterned region allowing the optimization algorithm to focus near high fidelity solutions.	129
Figure 6.2: Selection Error Probability: Pairwise comparison between high-fidelity and prediction values.	132
Figure 6.3: Flowchart of PSA-EA.	134
Figure 6.4: Effect of r_{ini} and τ_R on the performance of PSA-EA on each test problem when $FE_{max} = 100$ and $N_{cycle} = 100$	136
Figure 6.5: Effect of N_{cycle} on the performance of PSA-EA ($\tau_R = 2$, $r_{ini} = 0.4$) for the corresponding test problem.	138
Figure 6.6: Effect of r_{R2R} on the performance of PSA-ES when $\tau_R = 2$, $N_{cycle} = 20$, and $FE_{max} = 100$	140
Figure 6.7: Predicted values of new solutions and their true values after CFD simulation for both methods, both problems, and both objectives.	146
Figure 6.8: All generated solutions by PSA-EA and CS for problems $B34$ and $B38$	147

Figure 6.9: Generated solutions in the vicinity of the interest region for problems <i>B34</i> and <i>B34</i> . The base and the selected designs for fabrication are demonstrated by arrows.	147
Figure 6.10: IPM machine used for optimization.	152
Figure 6.11: Torque profile of reference design at rated operating conditions.	152
Figure 6.12: Geometric variables used for optimization.	154
Figure 6.13: 2010 Prius motor efficiency contours for 650 Vdc [2].	155
Figure 6.14: Ranking selection of solutions obtained by optimizing the surrogate-based optimization problem.	158
Figure 6.15: Objective space illustrating dominated and non-dominated solutions for optimization runs completed using NSGA-II and NSGA-II-WR.	162
Figure 6.16: Objective space illustrating Pareto-optimal fronts for cases of Setups A, B, and C.	165
Figure 6.17: Comparison of objective space with Pareto-optimal fronts and normalized design space of Pareto optimal sets. Pareto-optimal sets are obtained from two optimization methods, NSGA-II-WR and NSGA-II-WR-SA.	166
Figure 6.18: Exploration of objective space and MSE in prediction of objective functions, for each generation using NSGA-II-WR-SA.	167
Figure 6.19: Objective space highlighting the selected solutions using the a posteriori MCDM method.	170
Figure 6.20: Five selected Pareto-optimal solutions highlighted in normalized design space.	173
Figure 6.21: Magnetic flux density plots of Solutions 1, 2, 3 and reference design at rated operation.	173
Figure 7.1: Collaborative optimization practice using SOLVeR.	179
Figure 7.2: Phases and responsibilities.	181
Figure 7.3: Software architecture of pymoo.	193
Figure 7.4: Illustration of some crossover operators for different variables types.	199
Figure 7.5: Different visualization methods coded in pymoo.	203

LIST OF ALGORITHMS

Algorithm 4.1: Infill-And-Advance Interface	52
Algorithm 4.2: PSAF: A Probabilistic Surrogate-Assisted Framework	54
Algorithm 4.3: GPSAF: Generalized Probabilistic Surrogate-Assisted Framework	68
Algorithm 4.4: Probabilistic Knockout Tournament (PKT)	70
Algorithm 5.1: IC-SA-NSGA-II: Inexpensive Constrained Surrogate-Assisted NSGA-II . . .	96
Algorithm 5.2: Probabilistic Survival: Subset Selection under Uncertainty	108
Algorithm 5.3: Probabilistic Surrogate-Guided Mating	109
Algorithm 5.4: Heterogeneously Expensive Evolutionary Algorithm (HE-EA)	111
Algorithm 6.1: NSGA-II-WR-SA: NSGA-II with Repair and Surrogate Assistance.	159
Algorithm A.1: Scopus Query: Problem	213
Algorithm A.2: Scopus Query: Method	213
Algorithm A.3: Scopus Query: Goal	213

KEY TO ABBREVIATIONS

ASE	Approximate Solution Evaluation
BO	Bayesian Optimization
CEP	Computationally Expensive Problem
CFD	Computational Fluid Dynamics
DE	Differential Evolution
EA	Evolutionary Algorithm
EGO	Efficient Global Optimization
EI	Expected Improvement
EOP	Expensive Optimization Problem
ESE	Exact Solution Evaluation
ET	Evaluation Times
FEA	Finite Element Analysis
GA	Genetic Algorithm
GECCO	Genetic And Evolutionary Computation Conference
GPSAF	Generalized Probabilistic Surrogate-Assisted Framework
HE-EA	Heterogeneously Expensive Evolutionary Algorithm
MOP	Multi-Objective Optimization Problem
MTPA	Maximum Torque Per Ampere
PF	Pareto Front

PI	Probability Of Improvement
PKT	Probabilistic Knockout Tournament
PMSM	Permanent Magnet Synchronous Machine
PR	Polynomial Regression
PS	Pareto Set
PSAF	Probabilistic Surrogate-Assisted Framework
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
SE	Solution Evaluation
SOO	Single-Objective Optimization
TPE	Tree-Structured Parzen Estimator

CHAPTER 1

INTRODUCTION

1.1 Motivation

Optimization is omnipresent and has become an inherent part of our lives. The products we use and the services we consume result from a development process where all kinds of decisions regarding design and functionality had to be made. Such decisions are usually based on criteria to be met and measures to be minimized or maximized. For instance, the development of a washing machine requires implementing a controller that decides the amount of soap and water to add given the extent of dirt and grease with the ultimate goal of cleaning the laundry as effectively and efficiently as possible; smartphones are designed to maximize the user's experience considering functionality, usability, and aesthetics; or music streaming services aim to maximize the availability of songs and streaming quality while minimizing the consumer's bandwidth usage. For all these products and services, optimization plays a vital role during development and allows companies to keep a competitor's advantage through steady improvements. Being aware of the importance of optimization, one might want to know what optimization is and how it is defined. Indeed, this is an open question with different answers to it. On the one hand, mathematicians might state that the optimization's goal is finding inputs that satisfy the mathematical definition of optimality given a function to be either minimized or maximized. On the other hand, practitioners might view optimization generally as a tool to provide insights and support for decision-making when facing an application problem. A well-known scientist, George Dantzig, who made significant contributions to industrial engineering, once said:

True optimization is the revolutionary contribution of modern research to decision processes.

— George Dantzig

The usage of the term *true optimization* implies that optimization seems to be more than its mathematical definition and emphasizes that optimization as a process shall have a purpose connected to an overall goal: contributing to modern research. Such an emphasis on applicability shows the importance of the fusion of theory and practice. When facing a real-world optimization problem, identifying characteristics and possible challenges is always a good starting point.

1.1.1 Problem Characteristics and Challenges

Facing a real-world optimization problem often starts by investigating the problem's characteristics. For most applications, such an analysis will be manifold, because each trait must either implicitly or explicitly be considered by the optimization method. In the following, important characteristics of optimization problems are briefly discussed.

Variable Types: The search space of an optimization problem is based on variables and their types. For instance, common variable types are continuous, discrete, binary, or permutation. Where some problems have variables of the same type, others might be of a mixed nature. For example, mixed-integer programming refers to a problem type with continuous and discrete variables to be optimized at the same time [3], or a real-world tour planning problem, the traveling thief problem, consists of a permutation and binary variables [4]. Moreover, it is worth noting that the type of variables also indirectly impact the search space size. While the search space spanned by real-valued variables is infinite and uncountable, for a discrete search space, the cardinality is known however, in practice, intractable to iterate over.

Number of Variables: Not only the type but also the number of variables is critical. Different algorithms are proposed explicitly for single-variable, small-scale, or large-scale optimization problems. For example, a relatively simple optimization method such as the golden section method is known to be effective for optimizing a single variable [5]. However, the golden section search does not apply to problems with more than one variable. On the other extreme, algorithms have been customized to work efficiently on a larger scale. For instance, a genetic algorithm has been

customized to solve one *billion* binary variable problem [6], and stochastic gradient descend based methods with backpropagation [7] are used to optimize a couple of thousands or millions of variables in (deep) neural networks [8].

Number of Objectives: In the early stage of optimization, researchers have focused on optimization problems with a *single* objective. However, many optimization problems consist of multiple conflicting objectives and thus are multi-objective in nature [9]. Consequently, researchers have started to propose algorithms to simultaneously optimize multiple objectives with the goal of obtaining a so-called Pareto-optimal set of solutions. Because not a single solution but a whole set of solutions shall be found, population-based algorithms are predominantly used [10, 11]. It is worth mentioning that a multi-dimensional objective space has its own challenges, such as normalization [12], visualization [13], and decision making [14].

Constraints: Criteria that a solution needs to satisfy are referred to as constraints [15]. From a user's perspective, constraints have priority over objective values. No matter how good the solution's objectives are, a solution is considered infeasible if it violates one or more constraints [16]. Researchers distinguish between equality and inequality constraints when defining an optimization problem. Equality constraints are rather strict and may or may not be supported depending on the type of algorithm. Inequality constraints can have less or more impact depending on their definition. In some (rare) cases, the optimum with and without constraints might remain the same; in other cases, the optimum is shifted to the boundary of one or multiple constraints. Generally, constraints can have a non-neglectable impact on the optimization problem's complexity.

Differentiability: The availability of derivatives can be beneficial and they can be directly used by an optimization method [17]. Primarily, algorithms use the first or second-order derivatives to improve the convergence. However, for many real-world optimization problems, obtaining derivatives is impractical or impossible. Thus, gradient-free optimization must be applied, which is, in its more general form, also known as black-box optimization where no problem-dependent information is utilized [18].

Fitness Landscape: Most of the previously discussed problem characteristics are known before the optimization is applied. However, the nature of the fitness landscape is usually less obvious yet critical. In contrast to relatively simple uni-modal functions, a multi-modal function with a few or many local optima increases the problem complexity significantly. Many suboptimal regions require balancing exploitation and exploration during optimization. Whereas global optimization focuses on finding the single best optimal solution of a function with possibly many local optima, multi-modal optimization attempts to obtain all suboptimal solutions simultaneously [19]. When multiple local or global optima have been found, a common post-processing task is deciding what solution to choose, which often involves comparing the robustness or sensitivity of solutions.

Uncertainty: Many optimization algorithms assume that the objective and constraint functions are deterministic. However, especially in applications based on simulations, some underlying randomness may exist, and thus an evaluation with the same input produces different results. In other words, this type of problem consists of a non-deterministic function introducing uncertainty during evaluation. Some researchers address the uncertainty by converting the problem to be deterministic. The evaluation is repeated with different random seeds in such approaches, and the average and variance can then serve as objectives or constraints and be optimized. Even though this might be suitable for some problems, algorithms directly addressing the functions' stochasticity are preferred. Different kinds of optimization methods have been proposed in the research field of stochastic optimization [20].

Evaluation Time: For practitioners, the time spent for optimization stands in trade-off with the solution's quality. The time spent in optimization can be divided into two categories: first, time for evaluating solutions and, second, the algorithmic overhead. In real-world optimization, the evaluation of a solution often requires utilizing domain-specific third-party software to carry out simulations, for instance, Computational Fluid Dynamic (CFD) [21] or Finite Element Analysis (FEA) [22]. The optimization of problems with computationally expensive evaluations justifies more algorithmic overhead with the goal of selecting solutions to be evaluated more carefully.



Figure 1.1: Characteristics of optimization problems.

Certainly, this shall not be an exhaustive list of problem characteristics but give the reader of this dissertation an idea of how fundamentally different optimization problems are. In Figure 1.1, some of the key challenges are exemplified in a word cloud forming a light bulb. The illustration figuratively demonstrates the number of possible combinations of difficulties one might face in practice. Since they all need to be addressed, this also shows the challenge of finding a suitable optimization method for a specific application problem.

1.1.2 Facing a Computationally Expensive Problem (CEP)

The optimization of computational expensiveness objective and constraint functions has already been briefly discussed. However, since the computational expense is one of the biggest challenges practitioners face in practices, some more details are provided next.

One may raise the question, what kind of problems are computationally expensive? Technically, computational expensiveness is directly related to the number of necessary calculations until the evaluation has terminated. The number of calculations, in turn, is correlated to the problem's complexity and definition. Computationally expensive problems (CEPs) occur in many different

research areas and applications, and the origin of the computational expense is frequently caused by a simulation needed to be run. For instance, discrete-time simulations are employed to model a complex scenario. Let us consider the manufacturing process in a medium-sized enterprise where optimization aims to maximize productivity. The manufacturing process itself is rather complex and includes, amongst other things, arranging a production line or planning the production schedule. The complexity of the process makes it nearly impossible to reduce the objective to a few equations. Therefore, a discrete-time simulation is a common approach to forecast performance. However, to conclude with enough certainty, the simulation needs to be run for a large number of time steps, and thus the performance evaluation becomes computationally expensive. Besides discrete-time simulations, other simulations commonly faced in practice are computational fluid dynamics (CFD) or finite element analysis (FEA). Both computationally intense models are used to forecast fluid flows or other physical phenomena using partial differential equations. Especially in engineering, they are popularly utilized to make design decisions during development. Apart from simulations, time-consuming functions are an inherent part of analyzing a large amount of data. Thus, modeling or predicting based on a large data set requires a longer processing time since it is often unclear what model with what hyper-parameters to choose.

Instead of addressing the complex problem directly, one might suggest attempting to simplify the problem to reduce its complexity and, thus, the evaluation time. Simplifications, however, result in a different optimization problem to be solved, and obtained solutions are unlikely to be optimal regarding the more complicated model. Thus, we argue simplifications can be misleading and oversimplify the application to be investigated. For this reason, directly optimizing the CEP is recommended. However, the optimization method needs to consider the computational expense for a function call and deal with a limited evaluation budget.

1.1.3 The Role of Surrogate Model in Optimization and Some Terminology

When optimizing a CEP, typically the overall goal is not converging to the true optimum with many but to obtain a near-optimal solution with only a few solution evaluations. The execution of

a time-consuming evaluation dominates the algorithm’s computational overhead for finding new solutions in each iteration. Thus, an algorithm has more time for carefully selecting new designs than traditional optimization algorithms; however, the evaluation budget is usually only limited to a few hundred evaluations instead of a few thousand. A standard method to speed up the convergence of existing methods is using a *surrogate* model (also called metamodel, approximation model, simulation model, data-driven model, response surface), which approximates the time-consuming function. Incorporating a surrogate into the optimization process is indicated by adding "surrogate-assisted" or "metamodel-based" to the algorithm’s name or description.

Some more terminology used in this thesis is listed in Table 1.1. We distinguish between the time for a solution evaluation by referring to a computationally inexpensive evaluation using a surrogate and a time-consuming or computationally expensive evaluation running the application’s evaluation function. Moreover, in some cases, we emphasize the accuracy levels by differentiating between the low-fidelity evaluation of the surrogate and the high-fidelity of the application evaluation procedure. As an abbreviation, we denote the surrogate evaluation by approximate solution evaluation (ASE), and the application-based assessment simply by solution evaluation (SE) or in some cases to highlight that no approximation error exists by exact solution evaluation (ESE).

Figure 1.2 illustrates the relation between the time-consuming simulation, the surrogate model, and the optimization method. The optimization method uses the surrogate model to obtain predictions of the simulation’s outcome. The surrogate is generated based on data from the time-consuming simulations in the past. Inevitably, such an approximation model comes along with

Table 1.1: Terminology of different aspects using surrogates.

Aspect	Model / Simulation	Surrogate
Time	computationally expensive time-consuming	computationally inexpensive
Accuracy	high-fidelity	low-fidelity
Evaluation	Exact Solution Evaluation (ESE)	Approximate Solution Evaluation (ASE)

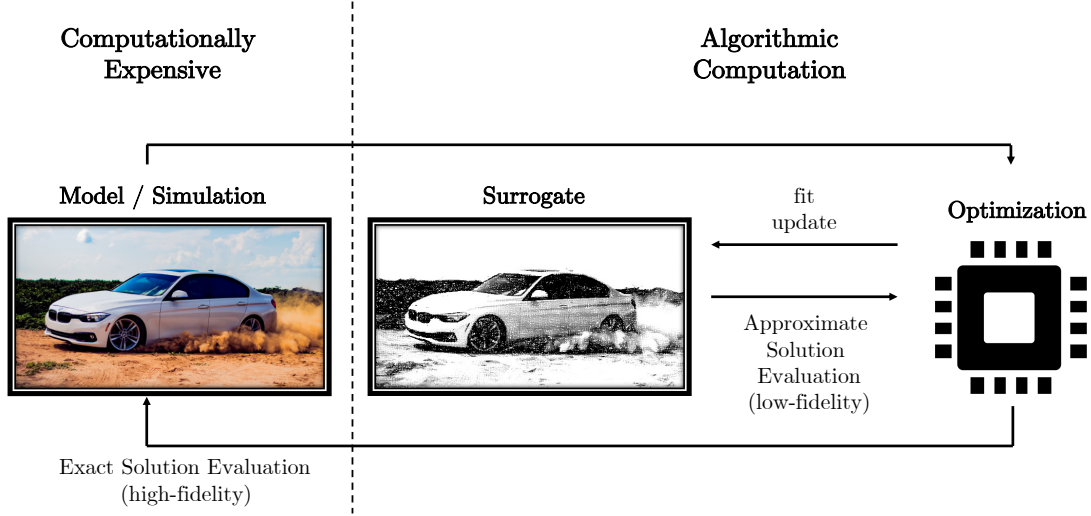


Figure 1.2: The relation between optimization and machine learning.

an inaccuracy. The most fundamental questions in terms of the design of an algorithm using a surrogate are:

What? Before thinking about how the surrogate is used, one might ask what should be modeled at all? Since optimization problems might not have only a single target function, objectives and constraints must be considered during optimization. Thus, for different optimization problems and algorithms, a different granularity of data exists. For instance, should the constraints be aggregated to the constraint violation before modeling, or should each constraint value be modeled separately? Similarly, should the objective values be aggregated with a decomposition function, or each objective be considered separately? These fundamental questions are interwoven with the design of an algorithm but need to be answered to propose any type of surrogate-assisted algorithm. For more information, we encourage the interested reader to look at the taxonomy that describes all possible combinations to model objectives and constraints [23].

With What? After determining what values a surrogate model should predict, one has to decide what type of model to use. Numerous models have been proposed in literature, each having up and downsides. Apart from deciding on the surrogate type beforehand, researchers have investigated selecting the most suitable surrogate dynamically to improve the robustness of the

surrogate assistance. A dynamic surrogate selection includes finding the surrogate type and the surrogate's hyper-parameters, which rapidly increases the time for modeling. For such a selection, the suitability of a surrogate needs to be assessed, which requires choosing training and test and defining an error metric. Both are critical and impact the overall performance of a surrogate-assisted algorithm.

How? Lastly, the question of how an optimization algorithm utilizes the surrogate is of importance. The usage of a surrogate is a more open question and could entirely vary from algorithm to algorithm. Whereas in some algorithms, the surrogates might serve as an assistant, the whole design is based on it in others. Moreover, the role of the surrogate can differ from being used in the vicinity of a solution providing local approximations or attempting to forecast the overall trend of the fitness landscape. Aside from usage, the updating procedure (also referred to as surrogate management) needs to be addressed and is another critical component for the performance of a surrogate-assisted algorithm.

These three fundamental questions must be answered when proposing a surrogate-assisted algorithm. We answer the *What* question throughout this thesis always by modeling each target function (objective and constraint) independently. A downside of such an approach is the increase in the modeling effort for problems with many target objectives and constraints. However, we argue that the additional computational burden is negligible due to the time-consuming functions. Moreover, it avoids prediction error accumulation through function aggregation, leading to higher accuracy. The answer to the *With what* question follows the recent development of selecting the best type and parameters of a surrogate dynamically. Finding the most suitable model for each target function has shown to be a robust solution in surrogate-assisted optimization. More details and what metric for surrogate selection is used is discussed in each of the chapters. Lastly, the *How* question. Different ways of answering *How* surrogates can be utilized during optimization shall be discussed in this thesis. Besides discussing how surrogates have been used throughout literature, we answer this question on a very generic level with only a few assumptions. This keeps

the proposed ideas applicable to many different types of optimization algorithms. Moreover, the *How* question is approached from different viewpoints, such as generalizability, asynchronicity, and practicability.

1.2 Research Goals and Contributions

This dissertation’s research goals and contributions are stated after introducing the optimization of computationally expensive functions using surrogate-assisted algorithms.

(i) Analysis and Classification of Surrogate-Based Algorithms: Optimization problems with expensive evaluation functions have become a vital research direction over the last few years. This dissertation analyzes relevant publications and categorizes them regarding their viewpoints. Moreover, it provides a thematic overview of surrogate-assisted optimization. It covers essential topics such as what type of surrogates have been used, how constraints or multiple objectives have been handled in the past, and other recent research trends. Moreover, it presents a large number of applications demonstrating the importance and practicability of surrogate-assisted optimization.

(ii) Proposal of a Generalizable Surrogate-Assisted Framework: In the last two decades, significant effort has been made to solve computationally expensive optimization problems using surrogate models. Regardless of whether surrogates are the primary drivers of an algorithm or improve the convergence of an existing method, most proposed concepts are rather specific and not very generic. Some important considerations are selecting a baseline optimization algorithm, a suitable surrogate methodology, and the surrogate’s involvement in the overall algorithm design. This dissertation proposes a probabilistic surrogate-assisted framework, demonstrating its applicability to a broad category of optimization algorithms [24]. The framework injects knowledge from a surrogate into an existing algorithm through a tournament-based procedure and continues the optimization run on the surrogate’s predictions. The surrogate’s involvement is determined by updating a replacement probability based on the accuracy from past iterations. The proposed framework enables the incorporation of surrogates into an existing optimization algorithm and, thus, paves

the way for new surrogate-assisted algorithms dealing with challenges in less-frequently addressed computationally expensive functions, such as different variable types, a large number of variables, multiple objectives, and constraints.

(iii) Proposal of a Methodology for Heterogeneous Expensive Optimization: A significant amount of research has been done in optimizing computationally expensive functions using different kinds of surrogate modeling approaches. Most studies, however, assume that the evaluation of objective and constraint functions are non-separable, and their values are available at the same time. However, in practice, the target functions can often be evaluated independently and are differently time-consuming or, in other words, are heterogeneously expensive. In this dissertation, we first investigate problems with separately computable computationally inexpensive constraint functions, while the objectives may still be time-consuming [25]. This scenario probably makes the simplest case of handling heterogeneous and multi-scale surrogate modeling in the presence of constraints. Second, we generalize the proposed concept to be applied to any kind of heterogeneously expensive problem [26]. The proposed method sends batches to an evaluator asking only a subset of objectives or constraints to be evaluated. This also requires dealing with partial information during optimization. Investigating heterogeneously expensive problems is vital for optimizing real-world optimization problems.

(iv) A Blueprint for Collaborative Optimization: One of the related issues for surrogate-based optimization is the need for executing the optimization task in collaboration between the domain experts and the optimization experts. Collaboration among different stakeholders in achieving a problem-solving task is increasingly recognized as a vital component of applied research today. For instance, in various research areas in engineering, economics, medicine, and society, optimization methods are used to find efficient solutions. Such a problem-solving task involves at least two types of collaborators – optimization experts and domain experts. Each collaborator cannot solve a problem most efficiently and meaningfully alone, but a systematic collaborative effort utilizing each other’s expert knowledge plays a critical and essential role. While many articles on the outcome of such collaborations have been published, and the justification of domain-specific information

within an optimization has been established, systematic approaches to collaborative optimization have not been proposed yet. In this dissertation, methodical descriptions and challenges of collaborative optimization in practice are provided, and a blueprint illustrating the essential phases of the collaborative process is proposed [27]. Moreover, collaborative optimization is illustrated by case studies of previous optimization projects with several industries. The study should encourage and pave the way for optimization researchers and practitioners to come together and embrace each other’s expertise as they solve complex problems of the twenty-first century.

(v) *Demonstrating Optimization of Computationally Expensive Functions in Practice:* Many real-world problems are associated with computationally expensive and time-consuming simulations for evaluation. Each candidate design should be selected carefully in such problems, even though it means extra algorithmic complexity. In this dissertation, two real-world engineering problems are investigated. First, we demonstrate the optimization of an engine cylinder head design, which has eight design parameters and two conflicting objectives [28]. Each design evaluation requires a detailed CFD simulation which takes about one hour using 32 CPUs. The optimization budget is limited to 61 design evaluations in total. Second, we present the optimization of an electric machine design optimization of a Toyota Prius 2010 motor. The problem consists of 10 design variables, two computationally expensive objectives, and 10 geometric computationally inexpensive constraints. The proposed method exploits the independently computable objective and constraint functions and their difference in evaluation time. The case studies demonstrate the relevance of optimizing computationally expensive functions in practice.

(vi) *Insights on the Design and Usage of an Optimization Framework:* Developing an optimization method entirely from scratch can be tedious and very time-consuming. Thus, choosing an optimization framework or customizing existing algorithms is wise and time-saving. As an example, this dissertation discusses the software design and features of pymoo, a multi-objective optimization framework in Python [29]. As the leading developers of the framework, we like to give insights into the class structure and organization and the meaning and responsibility of being in charge of a framework. We cover the architecture’s high-level overview to show its capabilities, followed by

explaining each module and its corresponding sub-modules. The algorithm implementations are customizable, and the source code can be easily modified/extended by supplying custom operators. The latter is crucial for rapid prototyping, often employed in practice.

1.3 Structure of the Thesis

This thesis is structured in the following way. Chapter 2 provides the necessary background information for the reader to follow along with this dissertation. On the one hand, this includes the basics of single and multi-objective optimization and, on the other hand, some state-of-the-art surrogate models. A brief history and a review of surrogate-assisted optimization are presented in Chapter 3. Moreover, we provide a categorization of existing surrogate-based optimization methods and give an overview of applications with computationally expensive functions across research fields. In Chapter 4, a generic surrogate-assisted method is proposed, first for single-objective optimization and then for constrained multi-objective optimization. In contrast to most existing approaches, the proposed method can be applied to a variety of metaheuristics. So far, most assume the evaluation procedure evaluates objectives and constraints all at once. However, since this is often not the case in practice, we look into the optimization of independently computable functions with varying expenses in Chapter 5. The proposed method exploits more inexpensive objectives and defines an information gain metric to find a suitable order for the evaluation process. In Chapter 6, we present the optimization of two engineering optimization problems with time-consuming evaluations and insights into collaborative optimization in practice, and the design of an optimization framework is given in Chapter 7. Finally, conclusions and future research directions are discussed in Chapter 8.

CHAPTER 2

FUNDAMENTALS

This chapter lays the foundation for understanding the components of surrogate-assisted optimization. First, the goal of (single-objective) optimization is verbally and mathematically defined. Second, multiple-objective optimization is introduced by explaining fundamental concepts such as Pareto-dominance and Pareto-optimality. Moreover, we discuss the history and design of genetic algorithms as they have emerged as the predominant choice for solving multi-objective optimization problems. Apart from the optimization, three widely used types of surrogate models are presented, and their benefits and drawbacks are discussed. Lastly, one of the most well-known surrogate-based algorithms called efficient global optimization (EGO) is described.

2.1 Single-Objective Optimization (SOO)

Before exploring the usage of surrogates in optimization, optimization itself needs to be defined. Nocedal and Wright describe optimization as "the minimization or maximization of a function subject to constraints on its variables"[30]. Mathematically, an optimization problem is given by

$$\begin{aligned}
 &\text{Minimize} && f(\mathbf{x}), \\
 &\text{subject to} && g_j(\mathbf{x}) \leq 0, && \forall j \in (1, \dots, J), \\
 & && h_k(\mathbf{x}) = 0, && \forall k \in (1, \dots, K), \\
 & && x_d^{(L)} \leq x_d \leq x_d^{(U)}, && \forall d \in (1, \dots, D),
 \end{aligned} \tag{2.1}$$

where \mathbf{x} denotes a vector in the search space $\mathbf{x} \in \Omega$ of length D , $f(\mathbf{x})$ the objective function mapping to a real number, $g_j(\mathbf{x})$ the j -th inequality constraints and $h_k(\mathbf{x})$ the k -th equality constraint. For each dimension d of the variable vector \mathbf{x} , a box constraints is given by $x_d^{(L)} \leq x_d \leq x_d^{(U)}$. The definition only considers minimization as it is known that

$$x^* = \operatorname{argmax} f(\mathbf{x}) = \operatorname{argmin} -f(\mathbf{x}), \quad (2.2)$$

and thus, any maximization can be converted to a minimization problem. In the remainder of this thesis, we will explicitly refer to single-objective optimization (SOO) if it shall be emphasized the optimization problem consists of a *single* objective function.

2.2 Multi-Objective Optimization (MOO)

In practice, optimization problems often do not consist of only one but multiple conflicting objectives. For some readers, this seems to be a relatively minor change in the problem definition; however, it significantly impacts the complexity of the optimization problem. A multi-objective optimization problem (MOP) is mathematically defined by

$$\begin{aligned} \text{Minimize} \quad & f_m(\mathbf{x}), & \forall m \in (1, \dots, M), \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq 0, & \forall j \in (1, \dots, J), \\ & h_k(\mathbf{x}) = 0, & \forall k \in (1, \dots, K), \\ & x_d^{(L)} \leq x_d \leq x_d^{(U)}, & \forall d \in (1, \dots, D). \end{aligned} \quad (2.3)$$

In contrast to single-objective optimization problems (see Equation 2.1), multiple objectives f_m where $m \in (1, \dots, M)$ are minimized which results in a multi-dimensional objective space of \mathbb{R}^M . The existence of a multi-dimensional design and objective space is illustrated in Figure 2.1. In this example, the design space consists of three variables ($D = 3$), and the objective space has a dimensionality of two ($M = 2$). Each three-dimensional point \mathbf{x} in the design space maps to a two-dimensional point in the objective space $f(\mathbf{x})$. It is interesting to observe that the two-dimensional objective space does not allow us to conclude if one solution \mathbf{p} is better than another solution \mathbf{q} by simply comparing two scalar values. Because of an additional dimension in the objective space, two vectors instead of two scalars have to be compared, and a more general

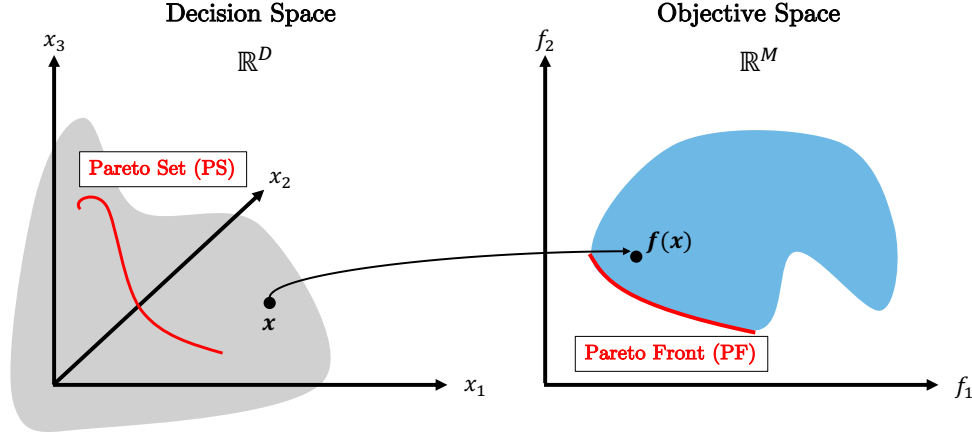


Figure 2.1: The design and objective spaces during optimization.

concept is necessary. One of the most important relations between two solutions in multi-objective optimization is Pareto-dominance:

Definition 2.2.1 (Pareto-dominance). A solution \mathbf{p} dominates another solution \mathbf{q} with the Pareto-dominance relation if $f_i(\mathbf{p}) \leq f_i(\mathbf{q})$ holds $\forall i \in \{1, \dots, M\}$ and $\exists j \in \{1, \dots, M\}$ such that $f_j(\mathbf{p}) < f_j(\mathbf{q})$.

In other words, a solution \mathbf{p} dominates another solution \mathbf{q} if (i) \mathbf{p} is not worse in any objective and (ii) is at least strictly better in one objective. Having defined the relation between two solutions, one can decide if a solution is optimal in a multi-objective context:

Definition 2.2.2 (Pareto-optimal). A solution \mathbf{x}^* is Pareto-optimal if there does not exist any solution \mathbf{p} which Pareto-dominates \mathbf{x}^* .

First, one can note that claiming that one solution is Pareto-optimal requires considering its relation to all other solutions in the search space. Second, the definition allows more than one solution to be Pareto-optimal. Commonly, researchers refer to all Pareto-optimal solutions in the design space as the Pareto set (PS). The PS maps to the objective space where it is called Pareto front (PF) defined by $\mathbf{PF} = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in \mathbf{PS}\}$. The goal of multi-objective optimization is to obtain \mathbf{PS} and thus \mathbf{PF} as quickly as possible.

Even though the definition of multi-objective problems (see Equation 2.3) includes any value for the number of objectives M , it is worth noting that researchers often refer to optimization problems with four or more objectives ($M \geq 4$) to *many*-objective optimization problems. One reason why $M \geq 4$ has been chosen is because of the increasing difficulty of visualizing results in more than three dimensions and the requirement of a different search methodology to handle many dimensions. For more details about multi- and many-objective optimization, we encourage interested readers to have a look at Kalyanmoy Deb’s book [9], one of the most formative and cited books in this research area.

2.3 Genetic Algorithms (GAs)

In the late 1950s and early 1960s, John Holland proposed to mimic sexually reproducing organisms in computer science [31]. Because it did not address recombination but only the mutation, researchers did not pay much attention to this initial study. Nevertheless, Holland continued his research and published his seminal book *Adaptation in Natural and Artificial Systems* [32] which is together with David Goldberg’s book *Genetic Algorithms in Search, Optimization and Machine Learning* [1] is considered as the birth of genetic algorithms. From there on, researchers and practitioners started to apply evolutionary computation to problems in various research fields, and it quickly gained popularity. Most other existing optimization methods during that time were based on a single solution attempting to be improved in each iteration. The major drawback of such *point-by-point* search algorithms is their lack of a global view of the fitness landscape increasing the likelihood of getting stuck in a local optimum significantly. In contrast, genetic algorithms (GAs) are based on a set of solutions and implicitly explore the search space but still exploit the knowledge of well-performing solutions using evolutionary operators.

Before describing the overall procedure of GAs, some terms frequently used in evolutionary computing will be discussed. Researchers commonly refer to the set of solutions as *population* and a single solution as *individual*. Moreover, each iteration of a genetic algorithm is called *generation*. An important hyper-parameter of genetic algorithms is the *population size* which determines the

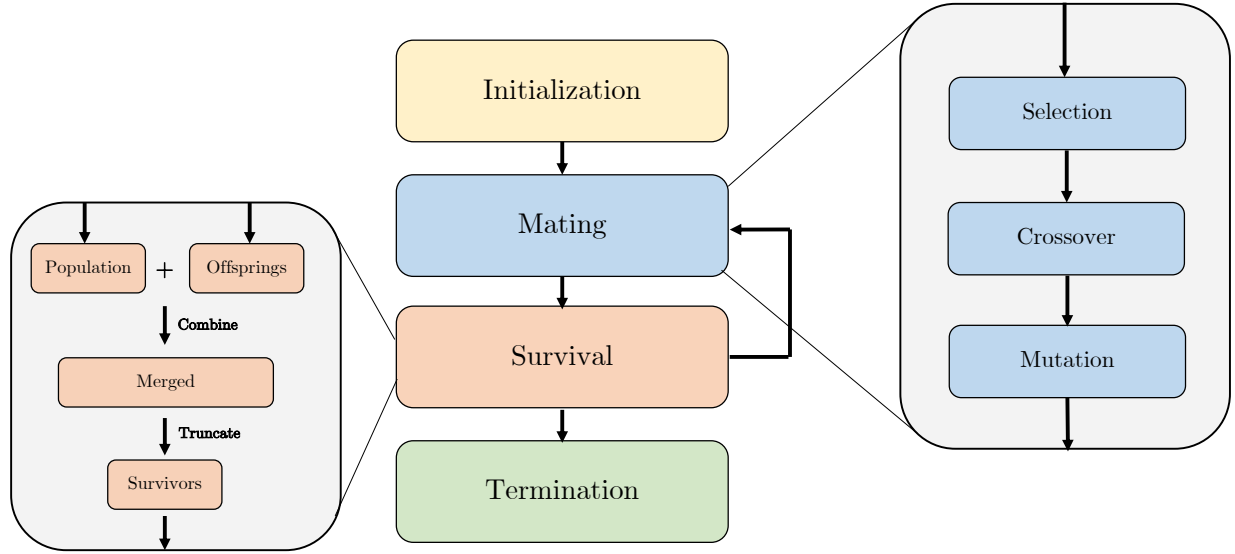


Figure 2.2: Flowchart of a genetic algorithm.

number of individuals kept in each generation. Figure 2.2 illustrates the procedure of genetic algorithms (GAs). First, the initial population is (usually randomly) created and evaluated. Then, the current population is used to create offsprings by executing a procedure called mating (also known as recombination). Mating itself consists of three steps: selection, crossover, and mutation. The selection creates a pool of individuals serving as parents. The crossover operator recombines the parents to generate one or multiple offspring solutions. The crossover’s design has the challenging task of incorporating information of all parents to create one or multiple new solutions. Then, the mutation is applied to each offspring by perturbing solutions with a specific pre-defined probability. The mating has been completed, and the offsprings are merged with the current population. This results in a population twice as large as initially. The (environmental) survival applies a natural selection to the new merged population and reduces it back to its original size. This reduction mimics the survival of the fittest in biology and is an essential aspect of evolutionary computation. After the survival, the algorithm either proceeds with the next generation or terminates.

GAs are known as a meta-heuristic because the realization of the evolutionary operators allows incorporating domain-specific knowledge into the algorithm. The usage of such custom operators in genetic algorithms is also referred to as *customization* [33]. One might have realized that processing

a set of solutions in each iteration is ideal for obtaining not a single but multiple optimal solutions in the end. Thus, GAs are a suitable candidate for finding a set of optimal solutions due to their search behavior.

2.4 Data Modeling and Predictors

Data modeling is a well-studied research direction with broad applicability in all kinds of disciplines. Thus, numerous approximation and interpolation models have been proposed over the last decades. A model aims to approximate the relation between input \mathbf{X} and output \mathbf{y} . Let the i -th data point of the input be denoted by $\mathbf{X}^{(i)}$ with $i \in (1, \dots, N)$ and be a vector of length $\mathbf{X}^{(i)} \in \mathcal{R}^D$. We refer to the d -th value of the i -th data point by $x_d^{(i)}$ and to the corresponding function value of $\mathbf{X}^{(i)}$ by $\mathbf{y}^{(i)} = f(\mathbf{X}^{(i)})$. Given another set of data points \mathbf{X}' and \mathbf{y}' originating from the same source, a predictor uses a model based on \mathbf{X} and \mathbf{y} to provide predictions $\hat{\mathbf{y}}'$ given \mathbf{X}' . Note that in this dissertation, we generally denote predicted values from models by $\hat{\cdot}$ (known as the *hat* symbol). Generally, the deviation between the predictions $\hat{\mathbf{y}}'$ and the true values \mathbf{y}' should be minimized. Some popularly used error metrics for measuring the deviation are the mean squared error, mean absolute error, or the coefficient of determination.

Since our purpose of using models is their utilization as surrogates during optimization, three common choices for surrogates – polynomial regression (PR), radial basis function (RBF), and Kriging – are discussed next.

2.4.1 Polynomial Regression (PR)

One type of surrogate model which has been used very early on in optimization is polynomial regression (PR). An important parameter of PR is the degree of the polynomial, which determines the complexity of the model. Let $P(\mathbf{X})$ be a function mapping an input \mathbf{X} to its polynomial representation. Then, one can formulate a system of linear equations

$$\mathbf{y} = P(\mathbf{X})\beta + \epsilon, \tag{2.4}$$

where \mathbf{y} is the output, β is coefficients to be found, and ϵ is the approximation error. Minimizing the approximation error ϵ results in

$$\beta = \left(P(\mathbf{X})^T P(\mathbf{X}) \right)^{-1} P(\mathbf{X})^T \mathbf{y}. \quad (2.5)$$

The shape and values of $P(\mathbf{X})$ differ depending on the polynomial degree. The most simple form of polynomial regression is *constant* where $P(\mathbf{X})$ is given by

$$P(\mathbf{X}) = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad (2.6)$$

which plays only a minor role in practice. More interesting, however, is a *linear*

$$P(\mathbf{X}) = \begin{pmatrix} 1 & \mathbf{X}_1^{(1)} & \dots & \mathbf{X}_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbf{X}_1^{(n)} & \dots & \mathbf{X}_d^{(n)} \end{pmatrix}, \quad (2.7)$$

or, *quadratic* regression

$$P(\mathbf{X}) = \begin{pmatrix} 1 & \mathbf{X}_1^{(1)} & \dots & \mathbf{X}_d^{(1)} & \mathbf{X}_1^{(1)2} & \dots & \mathbf{X}_i^{(1)} \mathbf{X}_j^{(1)} & \dots & \mathbf{X}_d^{(1)2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \ddots & & \vdots \\ 1 & \mathbf{X}_1^{(n)} & \dots & \mathbf{X}_d^{(n)} & \mathbf{X}_1^{(n)2} & \dots & \mathbf{X}_i^{(n)} \mathbf{X}_j^{(n)} & \dots & \mathbf{X}_d^{(n)2} \end{pmatrix}, \quad (2.8)$$

which is employed to estimate the first or second order derivative. It is worth noting that a smaller degree might not capture the function entirely but indicate trends. In contrast, a larger degree can model more complex functions but often suffers from overfitting. Moreover, in comparison to other models, PR does not necessarily fit through all data points exactly.

2.4.2 Radial Basis Functions (RBFs)

Radial basis functions (RBFs) [34] are real-valued functions used to predict new points through interpolation. The interpolation is based on the relation between pairs of solutions $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$. Given a distance $r = \|\mathbf{X}^{(i)} - \mathbf{X}^{(j)}\|$ between two points, their relation is expressed by a kernel function $k(r)$. Some well-known kernel functions are:

$$\begin{aligned}
 k(r) &= r && \text{(linear),} \\
 k(r) &= r^3 && \text{(cubic),} \\
 k(r) &= \sqrt{r^2 + \sigma^2} && \text{(multiquadratic),} \\
 k(r) &= e^{-r^2/\sigma^2} && \text{(gaussian),} \\
 k(r) &= r^2 \log(r) && \text{(tps).}
 \end{aligned} \tag{2.9}$$

The kernel function determines the type of relation between points. Given two sets of points A and B , the kernel function K defines the kernel matrix between $K(\mathbf{A}, \mathbf{B})$ by applying K to all possible point pairs between $\mathbf{a}^{(i)} \in \mathbf{A}$ and $\mathbf{b}^{(j)} \in \mathbf{B}$:

$$K(\mathbf{A}, \mathbf{B}) = \begin{pmatrix} k(\mathbf{a}^{(1)}, \mathbf{b}^{(1)}) & \dots & k(\mathbf{a}^{(1)}, \mathbf{b}^{(|B|)}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{a}^{(|A|)}, \mathbf{b}^{(1)}) & \dots & k(\mathbf{a}^{(|A|)}, \mathbf{b}^{(|B|)}) \end{pmatrix}. \tag{2.10}$$

To fit the RBF model, $\mathbf{A} = \mathbf{B}$ and thus the kernel matrix $K(,)$ and the tail $P()$, form a linear system of equations

$$\begin{bmatrix} K(\mathbf{X}, \mathbf{X}) & P(\mathbf{X}) \\ P(\mathbf{X})^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ c \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}, \tag{2.11}$$

to be solved for λ and c . The predictions for an unknown set of points \mathbf{X}' is then given by

$$\hat{\mathbf{y}}' = K(\mathbf{X}', \mathbf{X})\lambda + c^T P(\mathbf{X}'). \tag{2.12}$$

RBFs have the advantage of fitting through each data point and a prediction based on distances to other solutions in their neighborhood. However, one open question is what distance functions (most researchers use the Euclidean distance) and what kernel functions are the most suitable for modeling a specific data set.

2.4.3 Kriging

Kriging [35] (also known as Gaussian Process) is a powerful tool frequently used in Geostatistics and Machine Learning. Similar to RBFs, it uses a kernel function to make inferences [36]. Nevertheless, the motivation of Kriging is entirely different and based on a normal distribution of functions. The input \mathbf{X} and output \mathbf{y} are both assumed to be standardized, thus having a mean of zero and variance of one. Therefore, some data pre-processing might be necessary to apply Kriging if this is not the case. Predictions are based on a joint distribution between known inputs \mathbf{X} and \mathbf{y} and points to predict \mathbf{X}' with their true function values of \mathbf{y}' :

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}' \end{bmatrix} \sim \mathcal{N} \left(0, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & K(\mathbf{X}, \mathbf{X}') \\ K(\mathbf{X}', \mathbf{X}) & K(\mathbf{X}', \mathbf{X}') \end{bmatrix} \right). \quad (2.13)$$

Thus, we can derive the conditional distribution

$$\mathbf{y}' | \mathbf{X}, \mathbf{y}, \mathbf{X}' \sim \mathcal{N}(\mu_{y'}, \sigma_{y'}), \quad (2.14)$$

where

$$\mu_{y'} = K(\mathbf{X}', \mathbf{X}) \left(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} \right)^{-1} \mathbf{y}, \quad (2.15)$$

and

$$\sigma_{y'} = K(\mathbf{X}', \mathbf{X}') - K(\mathbf{X}', \mathbf{X}) \left(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} \right)^{-1} K(\mathbf{X}, \mathbf{X}'). \quad (2.16)$$

The values of $\mu_{y'}$ serve as predictions \hat{y}' , and the diagonal of the covariance matrix $\sigma_{y'}$ as the estimated prediction error. Not only providing a prediction but also an uncertainty measure is one of the major benefits of Kriging. Moreover, an interesting feature of Kriging is automatic relevance detection which optimizes over the influence of a specific dimension (or feature). This is done by adding D additional parameters to the kernel function, each representing the influence of a particular dimension. The relevance parameters are usually optimized using a maximum likelihood estimation. Nevertheless, another layer of optimization can further increase the computational burden, which is already known to be cubic with respect to the number of data points to fit the model.

2.5 Efficient Global Optimization (EGO)

One of the most popular surrogate-based algorithms is efficient global optimization (EGO) [37] (also known as Bayesian optimization (BO)) which shall be briefly explained. Efficient global optimization (EGO) is based on Kriging and has essentially three phases: (i) fit a Kriging model given all previously evaluated solutions, (ii) define an optimization problem using the predictions and uncertainty measure, (iii) solve the optimization problem to obtain a new infill point to be evaluated. This *fit-define-optimize* procedure has been widely used over the last years, and many different variants and extensions have been proposed. The definition of the optimization problem in (ii) is also known as the acquisition function or infill criterion. Based on the prediction of Kriging, a common acquisition function is the probability of improvement (PI) defined by

$$\text{PI}(\mathbf{x}) = P(f(\mathbf{x}) \geq f(\mathbf{x}^+)) = \Phi\left(\frac{\mu_{y'} - f(\mathbf{x}^+)}{\sigma_{y'}}\right), \quad (2.17)$$

where $\mu_{y'}$ and $\sigma_{y'}$ represent the output of Kriging, and \mathbf{x}^+ the current best solution. One downside of the probability of improvement is it measures only the likelihood of a solution being improved but not the amount of improvement. Thus, researchers have proposed expected improvement (EI) defined by

$$\text{EI}(\mathbf{x}) = (\mu_{y'} - f(\mathbf{x}^+)) \Phi(Z) + \sigma_{y'} \phi(Z), \quad (2.18)$$

where

$$Z = \begin{cases} \frac{(\mu_{y'} - f(\mathbf{x}^+))}{\sigma_{y'}} & \text{if } \sigma_{y'} > 0 \\ 0 & \text{if } \sigma_{y'} = 0 \end{cases}, \quad (2.19)$$

to encounter this drawback. More details about the history of EGO and its extension will be provided in Chapter 3.3. It is worth noting that with an increasing amount of data points, the Kriging model itself becomes computationally expensive. Also, studies have shown that with increasing dimensionality, the algorithm's performance deteriorates significantly.

2.6 Summary of the Chapter

In this chapter, we have introduced some fundamental principles and algorithms referred to throughout this thesis. First, we have presented the basics of single and multi-objective optimization. Second, genetic algorithms have been explained, and some basic (surrogate) models were introduced. Finally, a well-known surrogate-based optimization method called EGO was presented. Altogether, a brief introduction of fundamentals shall help the reader to follow along with this thesis.

CHAPTER 3

LITERATURE REVIEW

Even though handling computationally expensive function evaluations seems to be a rather practical manner, many relevant papers have been published in recent years. In this comprehensive literature review, we seek to identify the most relevant publications and keywords to get an idea of the recent developments. It is worth noting that optimizing computationally expensive functions is a task performed in many different research areas and thus rather interdisciplinary. The interdisciplinarity causes the authors' focus of studies and viewpoints to be diverse. Therefore, this chapter aims to keep the big picture in mind and consider different viewpoints of optimization computationally expensive functions.

3.1 Different Viewpoints on Computationally Expensive Optimization

Existing publications can be roughly put into three categories based on their focus: goal, method, or problem. Each viewpoint is reasonable and often determined by the authors' personal preference. An overview of each category and its keywords is illustrated in Figure 3.1. Moreover, each type is discussed in detail next.

Goal: The metadata of a publication, such as a title or keywords, often focuses on the goal or purpose. It is critical to converge to the (global) optimum with a limited number of function evaluations when optimizing time-consuming functions. Thus, relevant articles have been published under the umbrella of *efficient global optimization*. Moreover, it is worth noting that this is related to anytime optimization. As indicated by the name, the optimization's goal is the proposal of methods that can be interrupted *anytime*, having achieved the best result possible. Thus, anytime optimization focuses on developing algorithms considering the whole convergence curves and not only the final result. Comparing the convergence generally favors quickly converging algorithms to near-optimal solutions over slower converging algorithms finding the exact optimum.

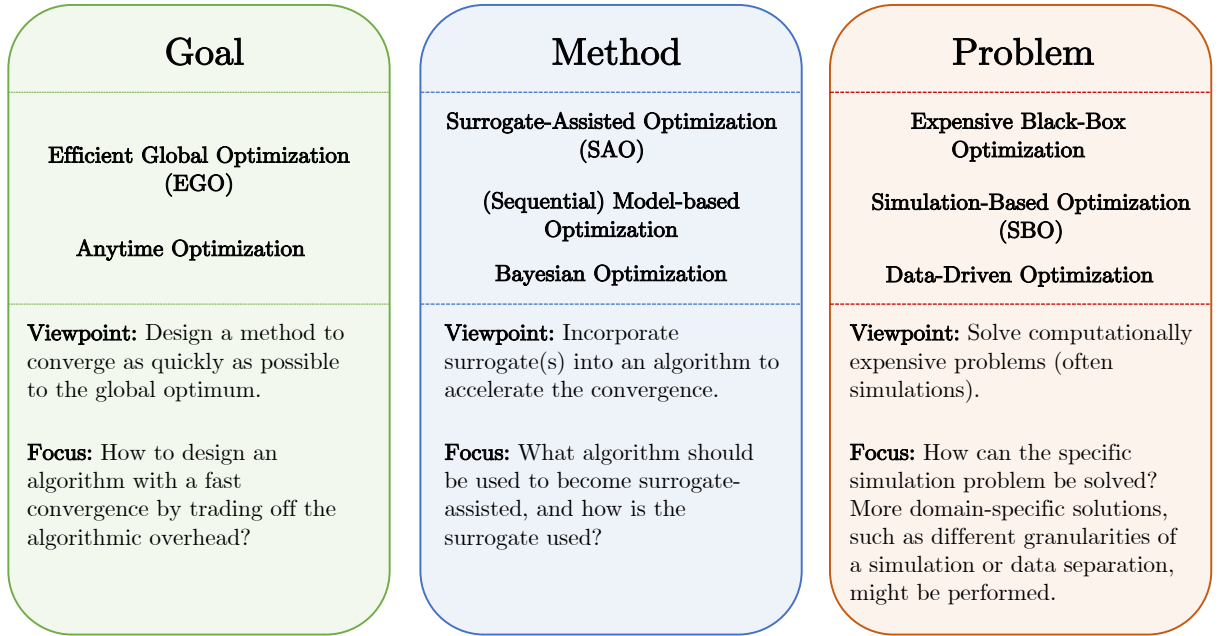


Figure 3.1: An interdisciplinary research area with different terminologies based on the perspective: goal, method, and problem.

Method: In contrast to the goal of optimization, the methodology to achieve the goal is frequently focused on. Thus, the authors emphasize the usage of *surrogates* during optimization as it is an inherent part of the proposed algorithm. It is worth noting that, nevertheless, researchers refer to a model commonly as surrogate; however, other terms such as metamodel, response surface model, approximation model, or simulation model have also been used in the past [38]. Although some terms might have (slightly) different meanings, one might find relevant publications by refining the search considering a different terminology.

Problem: Apart from emphasizing the goal or method, some researchers focus on the computationally expensive application itself. Problem-related literature requires a search regarding different types of applications that are either computationally expensive or supposed to be optimized with a minimal budget of function evaluations. Typically, simulation-based problems are of an expensive-nature and are thus worth searching for. An evaluation time of a couple of hours or even days is quite common. Moreover, one can think of optimization problems where a large amount of data needs to be processed. Data-intensive studies frequently address the use of distributed systems to

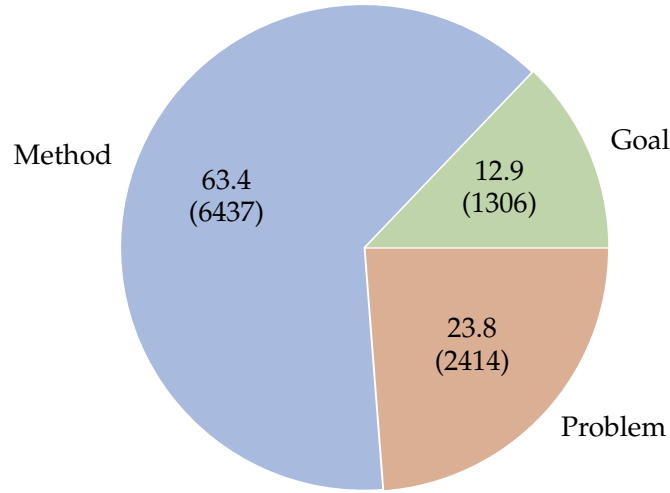


Figure 3.2: Overview of the overall percentage of each viewpoint.

parallelize and speed up function evaluation. So far, less attention has been paid to the usage of surrogates in such fields.

Identifying categories and keywords is essential for a comprehensive literature review. Next, we provide some more context and statistics about the viewpoints on optimizing computationally expensive problems described above.

3.2 Surrogate-Based Optimization: History and Recent Trends

Next, the importance and frequency of each of the viewpoints are evaluated. To visualize the interest and activity in the whole field and show the authors' usage of different viewpoints, we have performed a keyword analysis using Scopus [39]. The keywords and queries of the analyses can be found in Appendix 13. In Figure 3.2, the distribution of keywords mentioned in the publication's title is shown. It is apparent that most publications (63.4%) focus on the method itself and directly mention the surrogate or synonyms somewhere in the title or abstract. Nevertheless, a significant amount of research has also been conducted targeting the problem (23.8%) or the goal (12.9%). The distribution of the different viewpoints demonstrates the importance of looking at all facets of optimizing computationally expensive functions.

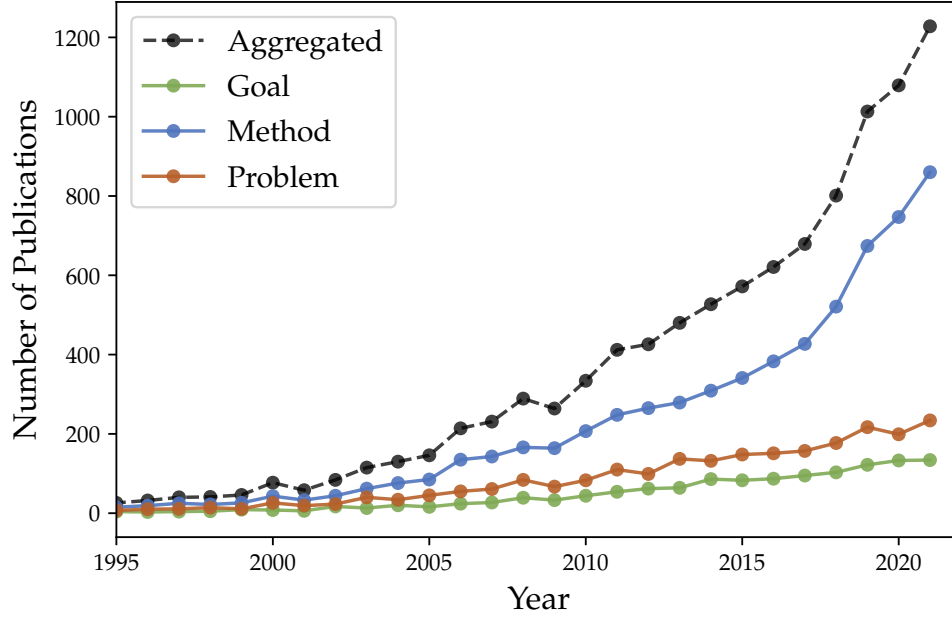


Figure 3.3: Analysis of the literature regarding publications related to surrogate-assisted optimization from 1995 to 2021.

Besides the overall distribution of viewpoints, the development over the last years shall be discussed. Figure 3.3 shows the number of publications in each year categorized by goal, method, and problem. The x-axis represents each year starting from 1990 to 2021, and the y-axis is the absolute number of publications. Each line is based on the number of publications in the specific year if the metadata includes at least one representative keyword of the category. Please note that one publication can be assigned to more than one category in this case. The aggregated curve, however, represents the sum of all publications, counting each publication exactly once. First, one can observe an increasing trend throughout all categories. This increase is not least caused by the relevancy of addressing the computational expensiveness in practice. Second, for publications focusing on the method, an almost exponential trend can be observed, whereas the goal and problem seem to behave more linearly. One possible reason for this trend is the usage of the term *surrogate* which has become the first choice by many researchers.

One venue where the usage of surrogates found a place is the genetic and evolutionary computation conference (GECCO) [40]. GECCO is one of the largest peer-reviewed conferences in

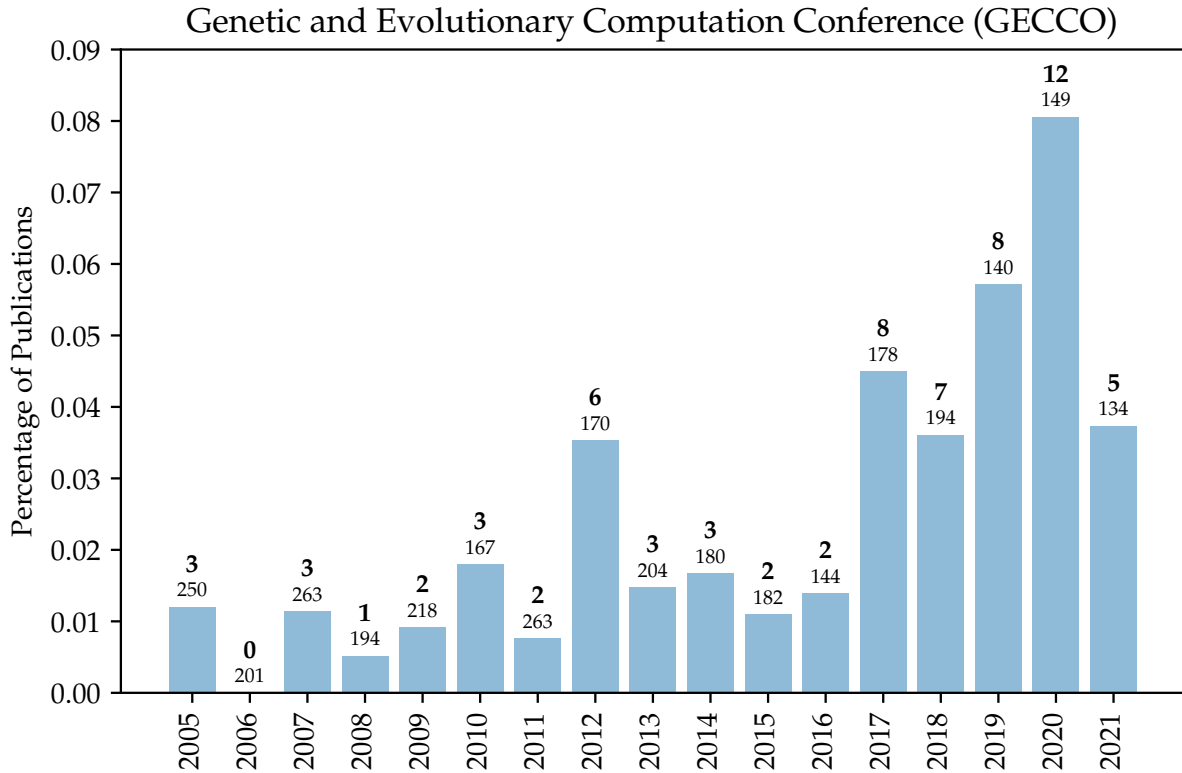


Figure 3.4: Overview of surrogate-assisted and related publications at *The Genetic and Evolutionary Computation Conference (GECCO)* from 2005 to 2020.

the field of Evolutionary Computation and the leading conference of the Special Interest Group on Genetic and Evolutionary Computation (SIGEVO) of the Association for Computing Machinery (ACM). In addition to evolutionary methods, researchers started to discuss approaches aiming to solve computationally expensive problems. To draw some conclusions about the development in the field, we have analyzed the number of publications addressing computationally expensive problems from 2005 to 2021. We have used the *BibTeX* files provided by ACM for this analysis and have filtered out all publications with two pages or less (poster publications). We have identified publications related to surrogate-assisted optimization by a keyword search¹ in the titles since the conference’s focus is methods and applications related to optimization, no other precautions to avoid finding irrelevant studies needed to be taken.

¹Keywords: surrogate, metamodel, model-based, computationally expensive, Kriging, Gaussian process, radial basis, response surface, EGO, efficient global optimization

In Figure 3.4, the years are shown on the x-axis and the percentage of publications matching the criteria for being related to surrogate-assisted optimization on the y-axis. The number of publications is located on top of each bar, and the number of each year’s accepted publications is printed below. One can observe an increasing trend, especially from 2017 onward. In 2020 more than eight percent of all publications are related to surrogate-assisted optimization and address the optimization computationally expensive optimization problems, which is encouraging.

So far, this chapter has introduced different viewpoints and provided some insights into the research field’s development. Next, a thematic literature review of state-of-the-art methods for optimizing computationally expensive problems is provided, and recent trends are discussed.

3.3 Relevant Literature and Applications

All algorithms utilizing surrogates during optimization have to find an answer to the *What?*, *With what?*, and *How?* questions discussed in Chapter 1.1.3. These questions are directly related to important topics in surrogate-assisted optimization, such as what type of surrogate or what optimization algorithm is used.

Type of Surrogate: The type of surrogate is a critical answer to the *With what?* question. The surrogate is generated based on information obtained through time-consuming evaluations from the past. Fitting a model based on data is a common task for Machine Learning methods [41] and thus a research field where many ideas originate or are borrowed from, especially in data-driven optimization [42]. Frequently, surrogates are used to predict the objective and constraint values directly. In general, for such a prediction, one can distinguish between approximation and interpolation models. Approximation models are not necessarily returning the exact values of the training data during prediction. For instance, PR [43, 44, 45] with a predefined degree is used to approximate the objective function. In [45], a linear regression model has been utilized, for which the prediction variance and inference quality can be determined. Many optimization algorithms use a replacement strategy and keep the newly evaluated solution if it outperforms the current one. For

such decision-based optimization algorithms, Support Vector Machines (SVMs) [46, 47] have been employed to predict the outcome of the replacement decision. The predictions are based on binary classification (will the new solution outperform the current one) by drawing a decision boundary in the design space [48]. By predicting the outcome of decisions, surrogate-assisted algorithms attempt to look at one or multiple iterations into the future using a surrogate [46] and thus improve their convergence behavior. It is worth noting that approximation models, such as polynomial regression with a smaller degree and SVMs, minimize the training error but will often not reduce it to zero due to the models' simplicity. In contrast, interpolation models attempt to fit each point of the data set exactly. A widely used interpolation model for surrogate-based optimization is Kriging (also known as Gaussian Process) [37, 49, 50, 51, 52, 53]. Kriging is known to be a good choice for problems with a low number of decision variables. One challenge of incorporating Kriging into an algorithm is the choice of a suitable hyperparameter configuration and the numerical instability for imbalanced data sets [54]. Numerous implementations of Kriging in all kinds of programming languages exist, but the Matlab DACEfit toolbox [55] made available by Lophaven et al. is one of the most commonly used by researchers. For optimization problems with a larger number of variables, researchers have frequently chosen RBFs as surrogates [54, 56, 57, 58, 59, 60]. In general, RBFs have been not only be used to model the function globally but also locally [57]. Similar to other surrogates, RBF uses kernel functions of a specific type (for instance, linear, multi-quadratic, gaussian), which is an important hyper-parameter to be considered [58]. In contrast to Kriging, RBFs do not provide any information about the estimated error or uncertainty; however, distances to existing high-fidelity evaluated points have been used instead [59]. Moreover, Neural Networks [61, 62, 63, 64, 65] have been utilized to model data sets. Even though neural networks are very effective in fitting data accurately, they have only been sporadically used for surrogate-assisted optimization. The main reason for that is the challenge of using neural networks with a limited amount of data [66].

Most of the surrogate types mentioned above are designed for real-valued input data spanning a continuous search space. For discrete variables, however, Random Forests have shown promising

results [67, 68, 69, 70, 71]. For instance, Hutter et al. [67] have proposed a sequential model-based algorithm configuration (SMAC) method, which is well-known for finding optimal hyperparameters without any manual tuning [67]. Also, some researchers have investigated surrogates for mixed variables types, for instance, a combination of continuous and discrete variables [72, 73]. Furthermore, Walsh functions [74] have recently been discovered to serve as surrogates for discrete optimization problems [75, 76]. Walsh functions can decompose any function of the Hilbert space and be naturally used as a basis for the space of pseudo-boolean problems [75]. Compared to other surrogates, Walsh functions are comparably computationally inexpensive and can be extendable for other variable types such as permutations [76]. They have been used for mutation in evolutionary algorithms [77] and might be one possible promising research direction for surrogate-assisted optimization. Furthermore, the tree-structured parzen estimator (TPE) [78, 79] has been utilized by surrogates for continuous, discrete as well as mixed variable problems. Whereas Kriging directly models $p(y|x)$, TPE models $p(x|y)$ and $p(y)$ (where x is defined by the variable and y by the objectives or constraints) [78]. Among other things, TPEs have been used for hyper-parameter optimization for neural networks and computationally expensive multi-objective optimization [79].

When having to choose from several surrogate types, it can be challenging to commit to a specific type during algorithm design. Also, intuitively the suitability of a surrogate depends on the fitness landscape [80]. Thus, the performance of different surrogates for different types of problems has been compared [81]. Moreover, methods using an online surrogate selection, also known as surrogate ensembles, have been proposed [54, 82, 83, 84, 85]. The ensemble of surrogates can be incorporated in different ways. For instance, a set of solutions can be obtained where each solution corresponds to an optimum found on a specific surrogate. Alternatively, one may combine all surrogates by taking the average of the predictions [82]. In [83], the authors have addressed the so-called *curse of uncertainty* by employing a weighted average of surrogate’s prediction where the weight is proportional to the surrogate’s root mean squared error. In contrast, the variance of predictions from a surrogate ensemble has been utilized as a measure for robustness in [56]. The amount of publications addressing different kinds of surrogates is quite large. Each surrogate

has its benefits and drawbacks, which is indicated by the variety of surrogates used in different studies. However, a general recommendation for what type of surrogate is suitable for what type of optimization problem is still an open question [38, 86].

Type of Algorithm: Early on, researchers have realized that response surfaces – a synonym for surrogates – are helpful to be used for optimization. [87, 88, 89]. A response surface has been fitted through a data set referred to as the design of experiments created in a space-filling manner. Then, an optimization algorithm is executed on the response surface, and an optimal solution is obtained [87]. This procedure became especially popular in engineering and is also known as offline or non-adaptive surrogate-assisted optimization. However, such a non-adaptive approach assumes the model to be relatively accurate and does not account for any model error. Because for most applications, an underlying modeling error is inevitable, a sequential surrogate update has emerged as one of the crucial components of surrogate-based optimization methods [90].

Sequentially updating surrogates opens up many different possibilities of making use of an approximation model during optimization. One type of algorithm where the surrogate is a substantial part of the optimization procedure is Bayesian optimization [91], also known as EGO [37, 92]. In EGO, the surrogate – typically Kriging [35] – provides a prediction and an uncertainty measure used to find a trade-off between exploitation and exploration. Both aspects are addressed by so-called acquisition functions (or infill criteria) such as expected improvement [37] or probability of improvement [92]. Especially in lower-dimensional search spaces, the exploration aspect in the acquisition function is important [93]. Other types of surrogate and scalarization techniques have been studied, such as radial basis functions with distance-based uncertainty measure [94] or linear spline function with a customized probability function [95]. All EGO approaches have a common embedded (global) optimization algorithm using the surrogate’s predictions. The limitation of finding only a single new solution in each iteration has been investigated thoroughly, and multi-point EGO approaches have been proposed [96, 97, 98, 99, 100]. Some known challenges of EGO are dealing with a large number of variables and numerical instabilities introduced by a biased solution distribution in the search space [54].

Instead of letting surrogates be the most substantial part of an algorithm's design, they have also been used to guide the search of existing algorithms. For instance, for evolutionary algorithms, a prescreening with surrogates has been commonly employed to improve the convergence [101, 102, 103, 104]. For prescreening, a larger number of offsprings is created through mating, and the surrogate predicts their fitness. Then an environmental selection is applied to reduce the offspring to a few being evaluated on the time-consuming function [101]. Such incorporation of surrogates is also known as generation-wise evolutionary control [105] and introduces naturally bias towards solutions being predicted to be more promising. Moreover, surrogates are incorporated into memetic algorithms [106] – evolutionary algorithms with an embedded local search – where they assist the local search in becoming more efficient [83].

Another well-known metaheuristic and type of evolutionary algorithm is differential evolution (DE) [107] known to be especially effective for the global optimization of continuous variables [108]. In DE, the mating is based on a crossover with a mutated individual. The mutation is based on the addition of a weighted differential vector. An offspring replaces an individual of the current population if its performance is superior. In [109] the surrogate serves as a classifier predicting whether a solution will be replaced or not; thus, if an offspring outperforms the current solution. This is in contrast to [110], where the authors propose only evaluating the most promising offspring based on the surrogate's prediction. Furthermore, the algorithm's global search has been complemented by a local search assisted with a surrogate [111]. The combination of global and local search provides an indirect impact through biased recombination and direct impact through a local refinement. Another approach has been proposed in [60], where the surrogate filters out only the most promising solutions analogously to the prescreening for genetic algorithms.

Besides DE, variants of the well-known model-based evolutionary algorithm covariance matrix adaptation evolution strategy (CMA-ES) [112, 113] with assisting surrogates have been proposed [46, 48, 114, 115, 116, 117]. In CMA-ES new candidate solutions are sampled according to a multivariate normal distribution which is continuously updated based on the best performing individuals determined by their rank. In [115], locally weighted (quadratic) regression [114]

approximates the ranking before the expensive evaluation takes place. One drawback of such a relatively simple approximation model is its simplicity and the lack of suitable fit for more complex fitness landscapes. Moreover, a full quadratic model requires a large number of solutions with increasing search space dimensionality, which has been addressed in [46]. Later on, the authors have proposed to replace the quadratic regression with a rank-based Support Vector Machine [48] with a Gaussian kernel. As a kernel matrix, the covariance matrix adapted from CMA-ES itself is used. Although the proposed method has shown to be more efficient, the results also indicate premature convergence on problems with a multi-modal landscape. These issues have been addressed, and the method has been further improved by the authors in [116], which only exploits the surrogate if it is sufficiently accurate.

Moreover, surrogates have been incorporated into particle swarm optimization (PSO) [118] in various ways. Similar to evolutionary algorithms, a generation can be simulated on the surrogate before being evaluated on the computationally expensive evaluation [119]. Another interesting way of using surrogate knowledge is the modification of the algorithm’s social component. For instance, in [120], attractors are derived from an embedded optimization on the ensemble of local and global surrogates and, thus, improve the algorithm’s convergence.

Multiple objectives: Optimizing multiple conflicting objectives at a time has been extensively studied in the last decades. However, improving the convergence by incorporating surrogates requires rethinking current approaches. Evolutionary algorithms are commonly employed to solve multi-objective problems because their population-based search fits the desire of obtaining a non-dominated set of solutions. A comprehensive overview of evolutionary algorithms assisted by surrogates can be found in [121]. The authors discuss 45 different algorithms proposed in the literature and categorize the approaches regarding their type and the number of surrogates. Moreover, we encourage interested readers to look at [122] for a more general overview of methods, challenges, applications, and recent developments in the research field. Commonly, multi-objective optimization algorithms are either based on dominance, decomposition, or indicators to address the existence of larger dimensional objective spaces [121]. For each approach incorporating a surrogate

has to be implemented differently. For instance, in [64, 123] a classifier has been learned to determine whether an offspring is dominated. The decision boundary drawn by the classifier to make the decisions regarding dominance is used to filter out non-promising offsprings without evaluating them expensively. Moreover, the approximations of objectives and constraints can be utilized to prescreen a set of solutions. For instance, Chugh et al. proposed KRVEA [124] – an extension of RVEA [125] with Kriging as a surrogate – performing a survival with the predictions of a surrogate before evaluating solutions expensively. In 2006, Joshua Knowles proposed ParEGO [126], where an iteration of the well-known EGO [37] algorithm is performed on scalarized objective values. The decomposition is based on the augmented Tchebycheff aggregation function [127] with a weight vector uniformly drawn from a set of reference directions. The rather expensive surrogate implementation has been further improved later on to handle a solution set up to a size of 500 [128]. In [129], Li et al. proposed a Kriging Metamodel Assisted Multi-Objective Genetic Algorithm (K-MOGA), which keeps expensively evaluated and predictions by the surrogate in the same population. The algorithm determines dynamically whether the predicted values need to be evaluated by the time-consuming function based on their domination status. The domination status considers the distances between non-dominated and dominated solutions and aims to represent the prediction error of the surrogate. Despite predicting each individual separately, the surrogate has been utilized to predict a multi-objective performance indicator. For instance, in [130], the authors proposed a Selection-based Efficient Global Optimization (SMS-EGO) algorithm, which uses the S-metric [131] as an infill criterion. The infill criterion was further adapted to consider a step-wise uncertainty reduction in [132] and a computationally cheaper infill criterion with similar performance [133]. In 2010, Zhang et al. proposed MOEA/D-EGO [134], which is a combination of the EGO idea and the MOEA/D [11] in general and was able to outperform ParEGO. MOEA/D-EGO fits local surrogates in the neighborhood determined by fuzzy clusters [135] and can evaluate more than one solution in each generation. Moreover, Habib et al. proposed Hybrid Surrogate-Assisted Multi-objective Algorithm (HSMEA) addressing computationally expensive many-objective optimization problems [136]. The proposed method is capable of handling constraints and keeps two

archives to improve the performance for irregular Pareto fronts. Also, it incorporates a local search subject to the angle of reference directions to improve the equality of infill solutions. The results on a wide range of test problems indicate that HSMEA performs significantly better than CSEA and K-RVEA. Recently, an adaptive Bayesian approach has been proposed by Wang et al. [137], which is based on the EGO principle but tunes the hyperparameter of the acquisition function according to search dynamics. The proposed method balances the exploration and exploitation by switching between an angle-based distance and an angle-penalized distance throughout the optimization. The method showed promising results on test problems and one real-world problem with a budget of 300 function evaluations.

Constraints: Efficient constraint handling plays a vital role in computationally expensive problems [59, 60, 94, 111, 138]. In the early phase of surrogate-assisted optimization, rather simple approaches have been used to address constraints. For instance, in [94] mostly box constraints have been considered for which it was not necessary to fit a surrogate because of their simplicity. Later on, methods have been proposed to handle more realistic computationally expensive problems with constraints using the feasibility first principle. In [59] the algorithm's performance has been shown on a real-world problem from the car industry with 68 inequality constraints. The proposed method uses the predictions of surrogates to approximate the constraint functions to find candidate solutions with the least constraint violation. A few solutions are selected from these candidates by a weighted ranking considering the predicted objective values and their distance to existing already evaluated solutions. Similarly, a feasibility prescreening has been proposed in [60], where the surrogate is utilized to improve the probability of evaluating a feasible solution in each generation. A more indirect way of surrogate-assisted constrained handling using a tournament selection based on predictions has been investigated in [111]. Based on predictions, two individuals compete with each other intending to choose the better one. If both solutions are feasible, the least infeasible of the two; if both are infeasible, the more feasible of the two. This selection pressure naturally introduces bias to evaluate feasible solutions in a genetic algorithm. Also, a common technique to deal with constraints by penalizing the objective function has been investigated using surro-

gates. As known for computationally inexpensive problems, the constraint violation's weighting is important to balance the impact of feasibility and infeasibility. In [139] the authors proposed to weigh the penalty in proportion to the number of feasible solutions. Results have shown that such an approach outperforms a constant weighting throughout the run of an algorithm. In [138], the proposed method minimizes the constraint violation until a feasible solution has been found and, second, aggregates the objective and constraint using a modified expected improvement function to proceed. Altogether, constraint handling is an essential aspect of optimization in practice, and it should be considered during the algorithm design. However, existing studies do not allow us to derive a clear tendency toward a surrogate-based constraint handling approach performing better than others.

Heterogeneous Computation of Functions: Less attention has been paid to problems with objective and constraint functions with varying expensiveness. Studies are limited to bi-objective problems where one objective is computational expensive and the other computational inexpensive (cheap) [140, 141, 142, 143, 144]. In 2013, Allmendinger et al. [140] has laid the foundation for investigating heterogeneously expensive optimization problems and proposed baseline methods, such as waiting for the slow objective to be evaluated or using the nearest neighbor approximation with Gaussian noise as a prediction. The authors extended their initial study in [141] in which they proposed evolutionary algorithms where the cheap objective is used as a look-ahead function for one or more generations. In 2018, Chugh et al. have proposed HK-RVEA [142], which uses Kriging as a surrogate to approximate the expensive objective. Moreover, a trust-region-based algorithm with quadratic approximations for objectives has been investigated later on in [143]. Wang et al. proposed the usage of transfer learning to make inferences about the more expensive objective from the cheap one [144]. It is worth noting that the difference in expensiveness between the objectives is critical. In existing studies, it has been assumed that one objective is two, five, 10, or 15 times slower than another [142, 144].

Miscellaneous: Different characteristics and challenges of optimization problems have been already

discussed in Chapter 1. A few of them shall now be reviewed with respect to surrogate-assisted optimization. The optimization problem might be computationally expensive but also have a large number of variables that need to be paid extra attention. To deal with such problems, either the algorithm handles the large-scaled decision space directly [145], or a feature selection is performed [146, 147]. Some real-world problems do not contain only a single optimization problem but multiple, which can be put in a hierarchy. These problems are especially challenging due to the embedded optimization which needs to be performed. For a bi-level optimization problem, the lower-level can be approximated by a surrogate [148]. Moreover, it is worth mentioning that simulations are often not deterministic and, thus, stochastic optimization shall be used [149]. Non-determinism implies that two uncertainties, one from the simulation and one from the surrogate, accumulate. Some simulations allow defining different levels of accuracy. For instance, running a simulation for one hour provides function values with a confidence level of 95%, whereas running it for two hours increases it to 99%. Optimizing such problems is commonly referred to as multi-fidelity optimization [150]. For example, the existence of multiple fidelity levels have been addressed with linear regression models in [45], and a test problem suite was proposed in [151]. Furthermore, the usage of surrogates in dynamic optimization has to be investigated. For instance, in [152] the concept drift during optimization is addressed using a sliding window approach for the surrogate data to be fitted. All these challenges of real-world optimization problems show what type of problems need to be looked at in connection with expensive functions.

Applications: Many practical optimization problems consist of expensive function evaluation. Thus, surrogate-assisted optimization has been widely applied in all kinds of interdisciplinary research areas. In the following, we highlight applications and research fields that have shown to be of importance and have had a significant impact. Table 3.1 provides an overview of research fields and concrete applications addressed in the literature.

Some problems directly related to Computer Science have an expensive evaluation function [67, 71, 153, 154, 155]. Most existing algorithms in the literature have parameters, also referred to as hyperparameters, which shall be tuned to maximize the performance. Hyperparameter-tuning

is especially expensive for non-deterministic algorithms where performance assessment requires running multiple runs [67]. One such example is finding a well-performing architecture of a neural network. Each network design requires optimizing the corresponding weights, which usually requires lots of computational resources and time [71, 153]. Moreover, learning decision policies, as commonly done in reinforcement learning, is a time-consuming task and requires many iterations. Thus, instead of interacting in the live environment, surrogates have been used to speed up the convergence time [154, 155].

Besides Computer Science, many other multidisciplinary research fields require the optimization of computationally expensive functions. For instance, in environmental optimization, various simulations of different kinds of optimization problems have been applied. One way of addressing risk is by simulation of possible scenarios in the future. For instance, time-consuming simulations have been optimized to forecast wildfire behavior [156], the seismic risk based on stochastic ground motion [157], or the spread of a pandemic [69]. Apart from assessing the risk or predicting the future outcome, the performance evaluation of configurations or policies is frequently based on simulations. For instance, surrogates have been used to obtain an optimal robotic milking barn facility allocation and to investigate the design's relation to herd size, feeding routine, and management practices [158]. Also, surrogates have frequently been employed to improve the convergence on problems related to water distribution systems in different ways, such as data-driven, projection, and hierarchical-based approaches [159, 160, 161, 162]. Furthermore, other environmental problems with expensive simulations related to the wind farm layout [163], reservoir management [164, 165], aquifer systems [166], sustainable transportation [167], wood-based composite materials [168], nuclear power plants [169], wind waves [170] have been investigated.

In general, the goodness of an engineering design often requires simulations for evaluation purposes. One research field is the design of aircraft structural components where CFD simulations are performed to predict aerodynamic forces and aerodynamic efficiency [173, 174, 175, 176]. Despite optimization related to the exterior shape of an airplane, surrogates have been employed to model turbulent reacting flows of aeronautical combustion chambers [178]. Also, simulations

Table 3.1: Overview of surrogate-assisted optimization being used in different kind of applications.

Research Field	Topic
Computer Science	Algorithm Configuration [67], Neural Architecture Search [153, 71], Reinforcement Learning [154, 155]
Environment Agriculture	Seismic Risk Assessment [157], Wild Fire Spread [156], Pandemic Forecasting [69], Crop Yield [171, 172], Aquifer Systems [166], Water Management [159, 160, 161, 162], Reservoir Management [164, 165], Robotic Milking Barn [158], Sustainable Transportation [167], Wind Farm Layout [163], Wood-based Composite Materials [168], Nuclear Power Plant [169], Wind Wave [170]
Engineering	Aircraft Structural Components [173, 174, 175, 176], Helicopter Rotor Design [177], Aeronautical Combustion Chambers [178], Vehicle Crashworthiness Design [179, 180], Vehicle Body Lightweight [181] DC Motor [182], Building Performance Simulation [183, 184, 185, 186], Fused magnesium furnaces [187, 188], Antenna [189, 190], Microwave Structure [191, 192, 193], Circuit Design Centering [194], Modular Flowsheet Optimization [195],
Medicine Biology	Health Care Operation Management [196, 197] Resource Planning Emergency Department [198], Trauma System [70], Breast Cancer [199, 200], Medical Image Registration [201], Protein Engineering [202]
Business Operations	Production Planing [203], Supply Chain [204], Traffic [205, 206, 207], Inventory Management [208, 209], Job Scheduling [210, 211], Enterprise Architecture [212], Manufacturing [213], Recommender Systems [214], Portfolio Optimization, Maintenance [215]

are not limited to airplanes but also have been applied to design the helicopter rotor in [177]. Another critical research direction is simulations related to automobiles. For instance, studies to maximize the vehicle's crashworthiness [179, 180], minimize the vehicle body weight [181] or to control the direct speed of a DC Motor have been conducted [182]. Moreover, the design of buildings has been investigated [183, 184, 185, 186]. Optimization problems consist mostly of a discrete search space and multiple objectives such as environmental quality or building energy consumption. Moreover, expensive simulations are performed in electrical and chemical engineering. For instance, surrogates have been utilized to optimize the antenna's design regards to maximum

gain, maximum front-to-back ratio, and minimal ground plane area [189, 190], the microwave design [191, 192, 193], circuit design centering [194], or a chemical modular flowsheets [195].

Apart from engineering, expensive simulation problems frequently occur in the medical and biological research field. For instance, studies about simulations of schedules of operations in a hospital in general [196, 197] or resource planning in an emergency department [198] have been conducted. Moreover, the design of a Trauma System [70] in Colorado with regards to the waiting time until a patient is being treated and the suitability and quality of treatment. The problems were driven by a data set with 100,000 emergency records of 72 hospitals with five different capability levels. Furthermore, more medical-related tasks such as the prediction of the growth of breast cancer [199, 200] or image registration (a process of transforming different sets of data into one coordinate system) [201] have been assisted by surrogates.

Also, in operation research, simulations are necessary to evaluate the outcome. Expensive simulations can take place anywhere along the value chain, such as manufacturing [213], production planning [203], supply chain management [204], or inventory management [208, 209]. For each task, the management has to choose one implementation out of many possible combinations by optimizing the company's objectives. For companies where logistics play a more critical role, simulations have been employed to optimize the (freight) traffic [205, 206, 207]. For marketing, so-called recommender systems suggest products to customers based on the current shopping cart or purchase history. Obtaining such suggestions can be computationally expensive because they are often based on a large amount of data, and thus surrogates have been used [214].

The variety of applications requiring expensive simulations demonstrates the need for suitable algorithms. For most such applications and simulations, surrogates have been employed to address the time-consuming simulations and make approximates available to the algorithm.

3.4 Summary of the Chapter and Open Issues

This chapter has provided an overview of surrogate-assisted optimization from different viewpoints. Moreover, we have presented a number of applications with time-consuming simulations demonstrating the relevance of research investigating the optimization of computationally expensive functions.

Despite all achievements in surrogate-assisted optimization over the last years, a few research directions have been paid only a little attention [38]. Many algorithms that have been proposed are rather specific and not very generalizable. Thus, a lot of surrogate-assisted methods address only a specific problem class. This results in numerous surrogate-assisted variants of algorithms where it is unclear what method is the most appropriate for an application problem. Thus, there is a need for a more general methodology to incorporate surrogates into an existing algorithm. Moreover, most studies assume that the objective and constraints are not separably evaluable and have the same evaluation time. However, for many real-world problems, the evaluation consists of calling different functions or third-party software products. Therefore, the independence with possibly varying evaluation times shall be directly exploited by an optimization method. Such practicable aspects have mostly been neglected so far in literature and are worth investigating. Besides aspects directly related to the optimization of time-consuming functions, optimization often is interdisciplinary and requires collaborations. Rarely, the characteristics and realizations of such collaborations have been focused on. However, in practice, collaboration is essential for the overall success of the project.

CHAPTER 4

A GENERALIZED PROBABILISTIC SURROGATE-ASSISTED FRAMEWORK

This chapter proposes a methodology for incorporating surrogates into metaheuristics and optimization in general. After showing the importance of considering the computational expense during optimization, different surrogate-based approaches are categorized regarding the role of the surrogate during optimization and the interdependency of the algorithm's design. First, we focus on computationally expensive *single-objective* optimization problems and propose a framework to incorporate surrogates into the optimization procedure. The incorporation is based on a probabilistic selection from a search pattern created by optimizing the surrogate. Afterward, we generalize this concept to solve constrained multi-objective problems by incorporating the Pareto-dominance principle and constraint violation of solutions into the probabilistic selection scheme. The majority of this chapter is based on [24] and [216] with some minor modifications to ensure consistency throughout this thesis.

4.1 Introduction

Many optimization problems are computationally expensive and require the execution of one or multiple time-consuming functions to evaluate a solution. Expensive optimization problems (EOPs) are especially important in practice and are omnipresent in all kinds of research and application areas, for instance Agriculture [172], Engineering [179], Health Care [196], or Computer Science [153]. Often the expensiveness of the evaluation is caused by the requirement of running a simulation, such as Computational Fluid Dynamic (CFD) [21], Finite Element Analysis (FEA) [22], or processing a large amount of data [217, 70]. The majority of simulation-based data-intensive problems is black-box in nature [218] and gradient information is not available or even more time-consuming to derive. Thus, it is vital to address the time-consuming objective and/or constraint functions as an inherent part of the optimization problem and limit the overall evaluation budget significantly.

The computational expensiveness is most commonly addressed by the usage of so-called surrogate or metamodels [104]. Substantial effort has been made based on a well-known approach known as efficient global optimization (EGO) [37]. The solutions being evaluated in each iteration are based on the optimum of a utility optimization problem – also known as infill criterion – commonly defined by the surrogate’s value and error predictions from Kriging [35]. Original limitations such as the evaluation of a single point per iteration, the lack of constraint handling, or dealing with multiple objectives have been investigated, and extensions have been proposed [126, 219]. Overall, the algorithm can be reduced to a *fit-define-optimize* procedure where the utility problem definition becomes more challenging when new problem complexities need to be handled. Moreover, the surrogate model is the *core* of all EGO approaches, and its accuracy is inevitably more critical for the algorithm’s performance.

Another category of surrogate-assisted algorithms uses an existing optimization method but incorporates one or multiple surrogates more organically [105]. Such approaches aim to improve the convergence behavior of the baseline algorithm and, thus, the anytime performance. Researchers have explored different ways of incorporating surrogates into population-based algorithms, such as genetic algorithms (GA) [1], differential evolution (DE) [220], or particle swarm optimization (PSO) [118] over the last years. All surrogate-assisted algorithms must find a reasonable trade-off between exploiting the knowledge of the surrogate and exploring the search space. On the one hand, researchers have looked into methods adding a surrogate with lighter influence on the original algorithm, for instance, using surrogate-based pre-selection in evolutionary strategy [221] or a predictor for the individual replacement in DE [109]. On the other hand, an existing algorithm’s behavior can be entirely biased by a surrogate by guiding the search more significantly. A global and local surrogate have been incorporated into PSO to solve expensive large-scale optimization problems [111] or into DE for expensive constrained optimization problems [222]. Numerous variants of surrogate-assisted algorithms indicates that many different ways of incorporating surrogates into an optimization method exist, but also that no best practice procedure has been established yet [38].

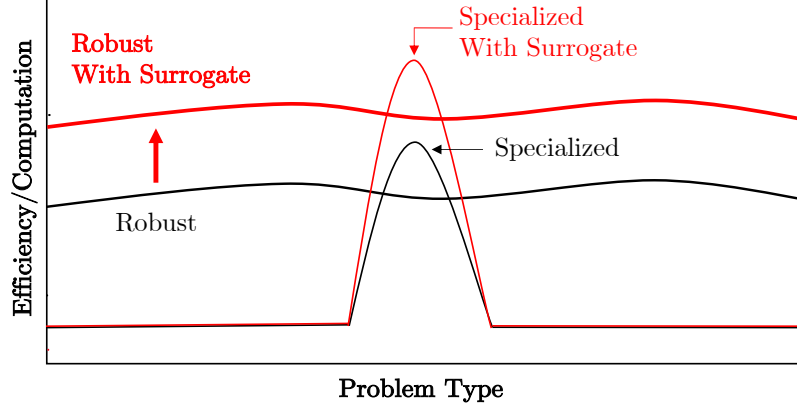


Figure 4.1: Robustly adding surrogate-assistance to population-based algorithms (illustration inspired by [1]).

The need for more generalizable concepts in surrogate-assisted optimization has been recognized, and frameworks aiming to solve a broad category of optimization problems have been proposed [83, 223]. These frameworks provide a generic method for solving unconstrained and constrained, single- and multi-objective optimization problems using the fit-define-optimize strategy. Despite these frameworks being applicable for numerous optimization problems, their design is rather challenging to incorporate and apply research conducted on computationally more inexpensive problems. Thus, a framework generalizing different algorithms is desired. In this chapter, we propose a novel surrogate-assisted framework that enables the ability to add surrogate assistance to population-based algorithms. Whereas most other surrogate-assisted algorithms aim to incorporate surrogates into specialized algorithms, the goal of this study is to provide a scheme to add surrogate assistance to a whole algorithm class (see Figure 4.1). Even though specialized surrogate-assisted algorithms are likely to outperform a generic concept on a specific problem type, the merit of this study is its broad applicability to different algorithms being proposed in literature. Because of the variety of existing algorithms for all kinds of problems, our framework is also generalized to numerous problem classes, such as unconstrained or constrained and single or multi-objective problems. The main contributions of this chapter can be summarized as follows:

- (i) We provide a categorization of existing surrogate-assisted algorithms regarding their surrogate usage. We distinguish methods regarding their surrogate’s impact and importance

during optimization and identify what has had less attention in the past.

- (ii) We propose a framework that uses an existing population-based algorithm’s search pattern and improves the convergence behavior incorporating surrogate assistance. In contrast to existing surrogate-assisted methods, we are using the entire search pattern of the search of an algorithm on the surrogate and not only using final solutions. This allows transferring features of existing algorithms to expensive optimization problems.
- (iii) The exploitation of the surrogate and the search space exploration is addressed using a probabilistic tournament selection based on points suggested by the algorithm. The surrogate prediction error is incorporated into the tournament selection and reliably balances the exploitation-exploration trade-off based on the surrogate’s accuracy.
- (iv) Our proposed method has truly surrogate-assisted characteristics. The surrogate guides the search depending on its accuracy and can have more or less impact on the baseline algorithm. Moreover, the maximum amount of impact can be regulated by setting a hyperparameter. In an extreme case, if the surrogate turns out to be a disadvantage during the search, it might even be disabled temporarily, and the method falls back to the default algorithm.

4.2 Background

In this section, the short overview of existing methods in the previous section will be enriched with details, and existing surrogate-assisted algorithms will be categorized regarding their surrogate incorporation.

Existing surrogate-assisted methods can be roughly put into one of the following categories based on the surrogate’s involvement: aided, customized, dependent, or once (see Figure 4.2). The latter describes the early development of optimization using the approximation model, fitted exactly once during optimization and never updated (once). Algorithms that perform an update of the surrogate can mostly depend on its predictions (dependent) or use it as an assistant in an existing method to improve the convergence behavior (aided, customized). The surrogate’s role and

dependency on the algorithm's design are vital for generalizing and, thus, shall be given special attention in this study. A thematic overview of these different types of surrogate involvement in an algorithm is given next. Especially in the early phase of surrogate-based optimization, the surrogate was fitted *only once* and optimized. Thus, the optimization's outcome entirely depends on the accuracy of the surrogate model.

The limitation of fitting a surrogate only once has soon been overcome by a more adaptive approach known as EGO (Efficient Global Optimization) [37]. The surrogate guiding the search is Kriging [35], which provides predictions and a measure of uncertainty for each point. The prediction and uncertainty together define the so-called acquisition function (or infill criterion), such as the expected improvement [37] or probability of improvement [92] aiming to balance exploitation and exploration simultaneously. The optimization of the acquisition function results in an infill solution, which is first evaluated and then added to the model. The procedure is repeated until a termination criterion is met. The limitation of finding only a single new solution in each iteration has been investigated thoroughly, and multi-point EGO approaches have been proposed [96, 99, 100]. Moreover, the concept has been generalized to solve multi-objective optimization problems by using decomposition [126, 134] or replacing the objective with a performance indicator based metric [130]. The idea has also been extended to handle constraints, which is especially important for solving real-world optimization problems [138]. Instead of using acquisition functions to address the surrogate's uncertainty, algorithms based on trust regions have been proposed. Inevitably, updating the trust-region radii becomes vital for the algorithm's performance [224]. Whereas original studies were limited to unconstrained single-objective optimization, the surrogate-assisted trust-region concept has been generalized to constrained and multi-objective optimization [83, 223].

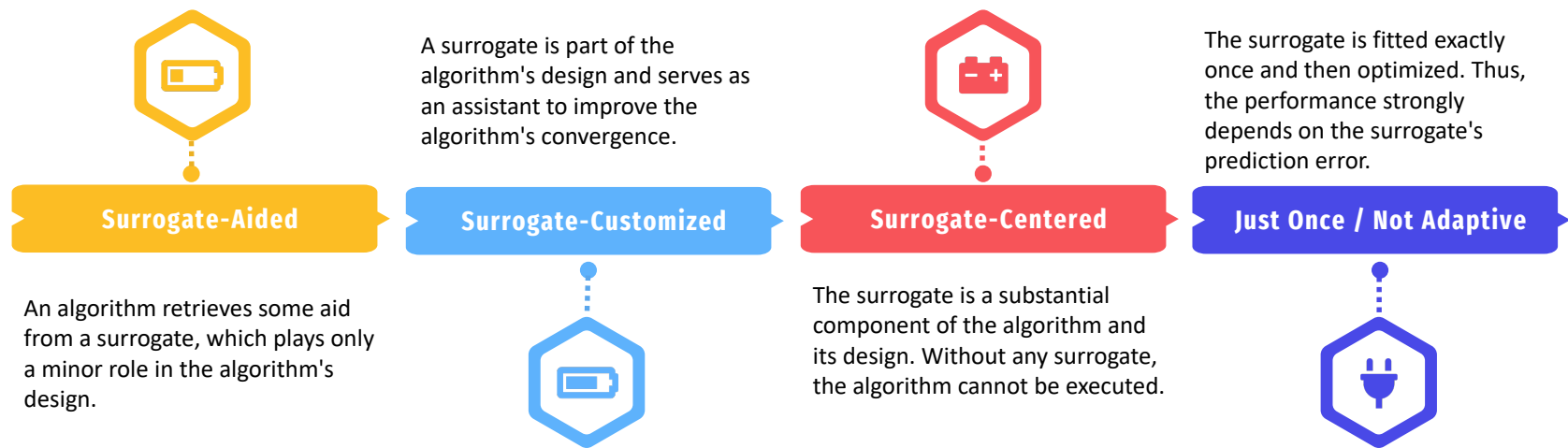


Figure 4.2: Different roles of surrogates in the design of an algorithm.

Apart from the approaches discussed above, the direct usage of surrogates in an algorithm has been explored in various areas, for instance, bi-level optimization [56, 227] or mixed-integer optimization [228]. All these approaches have one thing in common – the algorithm has been designed based on an approximating model and thus has a significant surrogate dependency. Therefore, the surrogate’s suitability and accuracy are critical for the optimization’s success. Inaccurate surrogate predictions and error estimations, inevitably occurring in large-scale optimization problems, are known to be problematic [38].

In contrast to algorithms being design-based on surrogates, researchers have investigated surrogates’ incorporation into existing optimization methods. Such approaches are also known as *surrogate-assisted* algorithms, emphasizing the surrogate’s role as an assistant during optimization. In our categorization, surrogate-assisted algorithms are split up into two categories. On the one hand, algorithms being aided by a surrogate where only minor changes of the original algorithm design are made; on the other hand, surrogate-customized methods where the algorithm has a significant impact on the algorithm’s design. Because the judgment of *impact* is subjective, the transition between both categories is somewhat fluent.

The benefit of surrogate-aided algorithms is that with relatively minor modifications, a surrogate has been incorporated, and the performance has been improved [104]. One well-known approach is a pre-selection (or pre-filtering), which uses a surrogate to select a subset of solutions that usually

Table 4.1: Categorization regarding the surrogate’s role in an algorithm.

Category	Algorithm / Study
Aided	MAES [221]
Customized	MOEAD-EGO [134], K-RVEA [225], HSMEA [136], CSEA [64], PAL-SAPSO [119], CAL-SAPSO [120]
Centered	EGO [37], ParEGO [126], SMS-EGO [130], Max-Min SAEA [56], SACOBRA [226], SABLA [227], GS-MOMA [83], GSGA [223], MISO [228], GOSAC [229]
Just Once	[230], [87], [231]

would be evaluated on the expensive problem [221]. Moreover, instead of changing the behavior in a generation, surrogates have also been used across generations by switching between the expensive evaluation and surrogate predictions entirely for some iterations [102, 105]. Another example for a surrogate-influenced algorithm is modifying a memetic algorithm (genetic algorithm with local search) by executing an evaluation-intensive local search on the surrogate [51].

Apart from surrogate-assisted methods with relatively minor modifications of existing algorithms, substantial customization based on well-known algorithms has been proposed by researchers. This has resulted in surrogate-assisted variants of well-known algorithms, such as lqC-MAES [232] derived from CMAES [112], KRVEA [225] and HSMEA [136] based on RVEA [125], MOEAD-EGO [134] as an improvement of MOEAD [11], or CAL-PSO [120] based on PSO [118], to name a few. Each surrogate-assisted variant is in principle based on an algorithm originally developed for computationally more inexpensive optimization problems but customizes the default behavior, for instance, by one or multiple local or global surrogates, implementing a sophisticated selection after surrogate-based optimization.

The increasing number of surrogate-assisted algorithms shows the importance and relevance of optimizing computationally expensive functions in practice. Indisputably, approaches and ideas directly designed for and dependent on one or multiple surrogates have their legitimacy but are rather difficult to use for newly proposed algorithms. The increasing number of surrogate-customized algorithms indicates the need for a best practice procedure and more generalizable methods. Thus, this study aims to investigate a surrogate-aided framework of algorithms applicable to a broad category of optimization methods.

4.3 Interfacing Metaheuristics

One of the major challenges when proposing a generalized optimization framework is the number and strictness of assumptions being made. On the one hand, too many assumptions restrict the applicability; on the other hand, too few assumptions limit the usage of existing elements in algorithms. In this study, we target any type of population-based algorithm with two phases in an

Algorithm 4.1: Infill-And-Advance Interface

Input : Algorithm Φ

```
1 while  $\Phi$  has not terminated do  
2    $X \leftarrow \Phi.\text{infill}()$   
3    $F, G \leftarrow \text{evaluate}(X)$   
4    $\Phi.\text{advance}(X, F, G)$   
5 end
```

iteration: the process of generating new solutions to be evaluated (infill) and a method processing evaluated infill solutions (advance). With these two methods, running an algorithm can be summarized by the pseudo-code shown in Algorithm 4.1. Until the algorithm Φ has been terminated, the *infill* method returns a set of new designs X to be evaluated. After obtaining the objective F and constraint G values for design, the algorithm is advanced by providing the evaluated solutions $\{X, F, G\}$. By looking at this interface, we further make two (weak) assumptions. First, we do not assume that X needs to be identical with the suggested designs from *infill* (Line 2 and 4), but can also be modified. Second, the *infill* method is *non-deterministic*, resulting in different designs X whenever called. Both assumptions can be considered weak because most population-based algorithms already fulfill them. So, how can existing optimization methods be described into *infill* and *advance* phases? Genetic algorithms (GAs) generate new solutions using evolutionary recombination-mutation operators and then process them using an environmental survival selection [1] operator; PSO methods create new solutions based on a particles' current velocity, personal best, and global best, and process the solutions using a replacement strategy [118]; CMAES samples new solutions from a normal distribution, which is then updated in each iteration [112]. Shown by well-known state-of-the-art algorithms following or being suitable to be implemented in this optimization method design pattern, this seems to be a reasonable assumption to be made for a generic framework. Moreover, it is worth noting that some researchers and practitioners also refer to the pattern as *ask-and-tell* interface.

However, how should this interface be utilized, and what role can surrogates play in improving the algorithm's performance? Precisely this is the subject of this work. Nevertheless, before moving on to the proposed framework, some more specifications of the surrogate usage are to be

defined. First, the surrogate shall only be used as an assistant (in contrast to other methods where everything is developed centered around the surrogate). Second, the proposed method should be adaptive, allowing to decrease and increase the impact of surrogate usage and, if desired, even falling back to the original pseudo-code shown in Algorithm 4.1. Third, the surrogate prediction error needs to be addressed to ensure both exploitation and exploration. Altogether, the design goals are formulated to make the optimization framework and surrogate incorporation flexible.

4.4 Probabilistic Surrogate-Assisted Framework (PSAF)

In the following, we propose probabilistic surrogate-assisted framework (PSAF), a framework for solving computationally expensive *single-objective* optimization problems.

4.4.1 Methodology

In contrast to most existing surrogate-assisted algorithms, PSAF uses not only the final solution(s) obtained by optimizing the surrogate but the whole *search pattern*. By making use of the search pattern, the exploration-exploitation balance is found by taking the surrogate’s accuracy into account. To allow even more flexible exploitation of the surrogate, we propose two phases. First, derive a solution set that is influenced by the surrogate, and second, introduce surrogate bias by optimizing the surrogate for a number of iterations. Both procedures are important to incorporate surrogates into existing methods effectively.

The overall outline of the algorithm is shown in Algorithm 4.2. The PSAF concept requires a baseline algorithm Φ which implements two methods – `infill()` and `advance(X, F)`. Moreover, three hyper-parameters, α , β , and $\rho^{(\max)}$ are passed to balance the surrogate’s influence on the baseline algorithm. First, the design of experiments $X^{(\text{doe})}$ are generated in a space-filling manner and evaluated $F^{(\text{doe})}$ using the time-consuming evaluation function (Line 1). An iterative procedure introducing surrogate bias to the baseline algorithm continues until the user-defined termination criterion is met. Each iteration begins with asking the baseline algorithm for new infill solutions. All steps from there on until the evaluation of X and the advancement of the algorithm (Line 29

Algorithm 4.2: PSAF: A Probabilistic Surrogate-Assisted Framework

Input : Algorithm Φ with `infill()` and `advance(X, F)`, Surrogate Tournament Pressure α (≥ 1),
Number of Simulated Iterations β (≥ 0), Maximum Surrogate-Bias $\rho^{(\max)}$ (≥ 0.0)

```
/* Sample Design of Experiments (DOE) */
1  $X^{(\text{doe})} \leftarrow \text{doe}()$ ;  $F^{(\text{doe})} \leftarrow \text{evaluate}(X^{(\text{doe})})$ 
2  $\Phi.\text{advance}(X^{(\text{doe})}, F^{(\text{doe})})$ 
3 while not terminated do
    /* Default infill solutions from baseline algorithm */
    4  $X \leftarrow \Phi.\text{infill}()$ 
    /* Fit a surrogate and predict values for infills */
    5  $S \leftarrow \text{fit}(X^{(\text{doe})}, F^{(\text{doe})})$ 
    6  $\hat{F} \leftarrow S.\text{predict}(X)$ 
    /* Surrogate-assisted Tournament Pressure ( $\alpha$ ) */
    7 foreach  $k \leftarrow 2$  to  $\alpha$  do
    8      $X^{\alpha_k} \leftarrow \Phi.\text{infill}()$ 
    9      $\hat{F}^{\alpha_k} \leftarrow S.\text{predict}(X^{\alpha_k})$ 
    10    foreach  $j \leftarrow 1$  to  $\text{size}(F^{\alpha_k})$  do
    11        if  $\hat{F}_j^{\alpha_k} < \hat{F}_j$  then  $X_j \leftarrow X_j^{\alpha_k}$ ;  $\hat{F}_j \leftarrow \hat{F}_j^{\alpha_k}$ ;
    12    end
    13 end
    /* Bias by Continuing the Algorithm  $\Phi'$  on Surrogate ( $\beta$ ) */
    14  $\Phi' \leftarrow \text{copy}(\Phi)$ 
    15  $X^\beta \leftarrow X$ ;  $\hat{F}^\beta \leftarrow \hat{F}$ 
    16 foreach  $k \leftarrow 1$  to  $\beta$  do
    17      $X^{\beta_k} \leftarrow \Phi'.\text{infill}()$ 
    18      $\hat{F}^{\beta_k} \leftarrow S.\text{predict}(X^{\beta_k})$ 
    19     foreach  $j \leftarrow 1$  to  $\text{size}(\hat{F}^{\beta_k})$  do
    20          $i \leftarrow \text{closest}(X_j^{\beta_k}, X^\beta)$ 
    21         if  $\hat{F}_j^{\beta_k} < \hat{F}_i^\beta$  then  $X_i^\beta \leftarrow X_j^{\beta_k}$ ;  $\hat{F}_i^\beta \leftarrow \hat{F}_j^{\beta_k}$ ;
    22     end
    23      $\Phi'.\text{advance}(X^{\beta_k}, \hat{F}^{\beta_k})$ 
    24 end
    25  $\rho \leftarrow \min(\text{estm\_surr\_bias}(S), \rho^{(\max)})$ 
    26 foreach  $j \leftarrow 1$  to  $\text{size}(X^\beta)$  do
    27     if  $\text{random}() < \rho$  then  $X_j \leftarrow X_j^\beta$ ;
    28 end
    /* Next iteration of the overall algorithm */
    29  $F \leftarrow \text{evaluate}(X)$ 
    30  $A.\text{advance}(X, F)$ 
    31  $X^{(\text{doe})} \leftarrow X^{(\text{doe})} \cup X$ 
    32  $F^{(\text{doe})} \leftarrow F^{(\text{doe})} \cup F$ 
33 end
```

and 30) introduce surrogate bias. The surrogate bias consists of two phases: the α -phase with a light influence using a tournament selection based on surrogate predictions (Line 7 to 13); and

the β -phase simulating the algorithm for a number of generations on the surrogate and accepting solutions with the probability ρ derived from the surrogate's accuracy (Line 14 to 28).

4.4.1.1 Influence of Surrogate through Tournament Selection Pressure (α)

A well-known concept in evolutionary computation to introduce a bias toward more promising solutions is *tournament selection*. An individual from the population has to win a tournament to contribute to the mating process. The number of competitors (α) balances how greedy the selection procedure will be. On the one hand, a larger value of α allows only elitist solutions to participate in mating, while a smaller value introduces less selection pressure. For genetic algorithms, the most frequently used tournament mode is the binary tournament ($\alpha = 2$), which compares a pair of solutions regarding one or multiple metrics. A standard binary tournament implementation for constrained single-objective optimization declares the less infeasible solution as the winner if one or both solutions are infeasible or otherwise the solution with the smaller function value.

In the context of surrogate assistance, the tournament selection introduces surrogate bias during the generation of new infill solutions. Whereas in genetic algorithms, evaluated solutions (using ESE) compete with each other during mating selection, in PSAF solutions evaluated on the surrogate (ASE) are compared. Figuratively, the surrogate serves as a referee in a tournament by providing predictions before the evaluation. In Figure 4.3, surrogate-assisted tournament selection with three competitors ($\alpha = 3$) for four infill solutions is shown. Initially, the algorithm's `infill` function has been called three times to generate X^{α_1} , X^{α_2} , and X^{α_3} . Then, a tournament takes place where the i -th solutions of the j -th infill solution set $X_i^{\alpha_j}$ compete with each other. For instance, for the first tournament the winner of $X_1^{\alpha_1}$, $X_2^{\alpha_1}$, and $X_3^{\alpha_1}$ has to be declared. As a comparison function, the surrogate's approximation function $\hat{F}_i^{\alpha_j}$ is used. In general, setting $\alpha = 1$ disables the tournament selection and serves as a fallback. By involving the surrogate in the tournament selection ($\alpha > 1$), the infill solutions X get a smaller or larger influence based on the number of competitors.

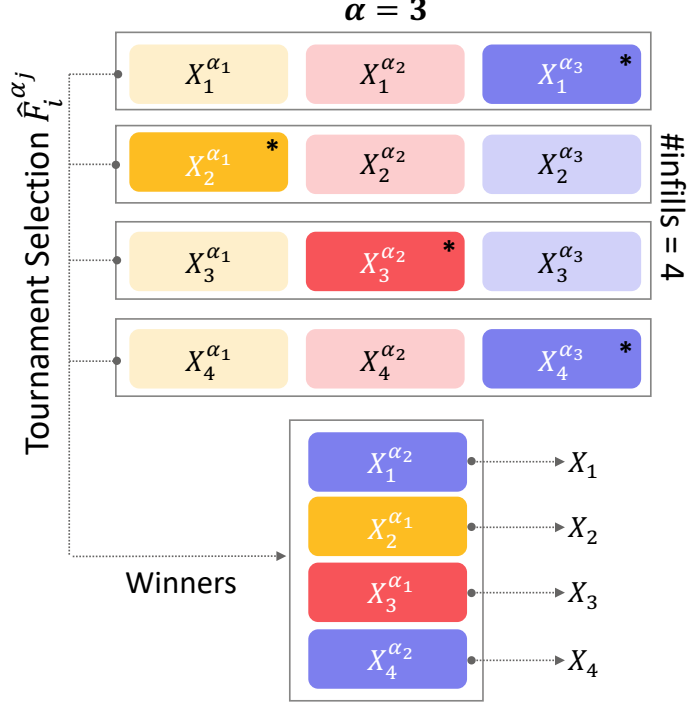


Figure 4.3: Tournament selection with α competitors to create a surrogate-influenced infill solutions.

4.4.1.2 Continue Optimization on Surrogate (β)

While the tournament is an effective concept to incorporate the surrogate's approximation, it is limited by looking only a *single* iteration into the future. To further increase the surrogate's impact, the baseline algorithm is continued to run for β more consecutive iterations on the surrogate's approximations. Inevitably, the question of how many iterations are suitable arises and indicates the importance of tuning β . Nevertheless, even more critical, how should the algorithm profit from simulating the algorithm on the surrogate? An inappropriate choice of β will cause the surrogate's optimum to be repeatedly found and will entirely discard the baseline algorithm's default infill procedure. This also causes a diversity loss of infill solutions and does not account for the surrogate's approximation error. Thus, we propose a probabilistic surrogate-assisted approach that balances the surrogate's impact on the baseline algorithm to address these issues.

The probabilistic procedure is described in Algorithm 4.2 from Line 14 to 28. Because the iterations are only simulated on the surrogate, the original algorithm object Φ must be copied to

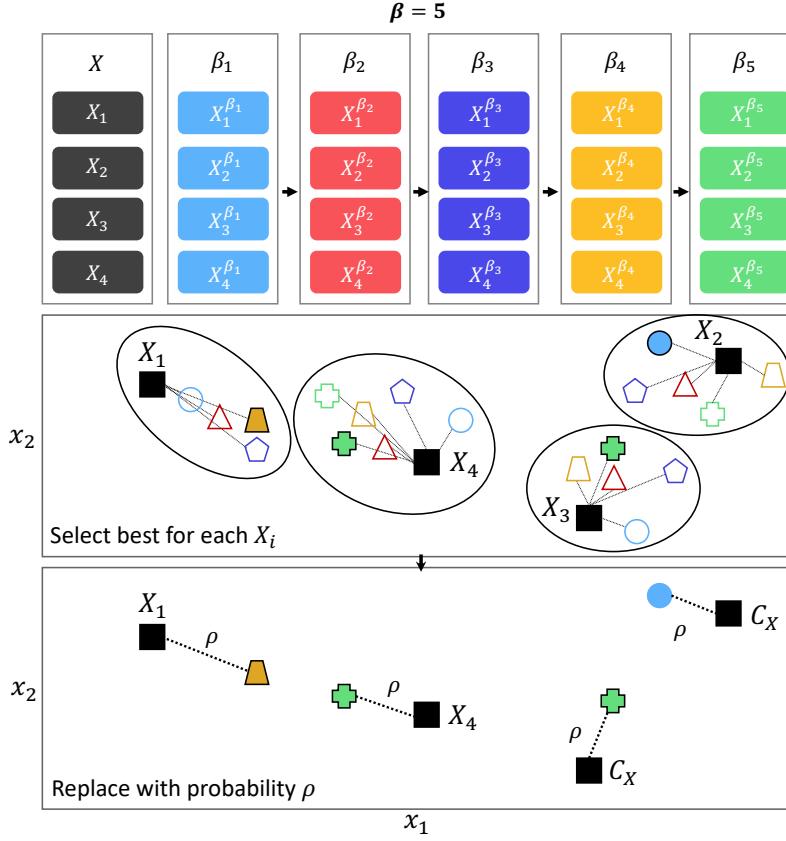


Figure 4.4: Continuation of the algorithm's run for β iteration on the surrogate.

Φ' to avoid any modifications of the current algorithm's state (Line 14). Then, the algorithm's run (of Φ') is continued on the surrogate model for β iterations, by calling in each iteration k the `infill` method returning X^{β_k} and feeding back to the algorithm the approximations \hat{F}^{β_k} by calling `advance` (Line 17, 18, and 23). The goal of these iterations is to introduce more surrogate-bias into X . Therefore, a surrogate-biased population X^β is obtained by initializing $X^\beta = X$ and $\hat{F}^\beta = \hat{F}$ (Line 15) and assigning in each iteration (k) every solution $X_j^{\beta_k}$ to its closest solution i in X^β . The closest solution is determined based on the smallest (normalized) Euclidean distance in the design space. The infill X_i^β and the corresponding prediction \hat{F}_i^β is replaced if the newly found solution performs better considering the surrogate's prediction. Finally, a biased candidate solution X_j^β replaces X_j with probability ρ bounded by $\rho^{(\max)}$. Clearly, the value of ρ determines the impact of the β -phase on the baseline algorithm.

An example with five iterations ($\beta = 5$) and four infill solutions X_1 , X_2 , X_3 , and X_4 is also

illustrated in Figure 4.4. Calling the infill function of the baseline algorithm results in five solution sets with four solutions each. When running the algorithm, the assignment takes place, and for instance, X_1 has four solutions being the closest to, and X_4 has six. The assignment of the closest solution will show cluster-like arrangements and preserve diversity.

In general, optimizing the surrogate model is a common technique used in surrogate-assisted algorithms. However, a crucial aspect is addressing the utilization of knowledge from these iterations. An assignment-based and probabilistic approach keeps the balance between the default algorithm's behavior and surrogate bias. The strategy to determine the "bottleneck" variable ρ of the β -phase is described next.

4.4.1.3 Balancing the Utilization of Surrogate (ρ)

The number of infill solutions being finally biased by β iterations on the surrogate is critical and balances the whole surrogate's involvement. In industry projects finding a suitable surrogate can be rather challenging and is often done manually. Comparing different types of surrogates and selecting the most suitable one is usually based on a metric that judges a model's trustworthiness. A well-known metric to estimate the accuracy is the coefficient of determination (also known as R^2):

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{f}_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{\text{MSE}(y, \hat{f})}{\text{MSE}(y, \hat{f}^{\bar{y}})}, \quad (4.1)$$

where y_i represents the output and \hat{f}_i the prediction of the i -th value, and \bar{y} the arithmetic mean of all output values. The denominator $\sum_i (y_i - \bar{y})^2 = \text{MSE}(y, \hat{f}^{\bar{y}})$ represents the Mean Squared Error (MSE) of a surrogate $\hat{f}^{\bar{y}}$ always predicting the average of all output values. This error serves as the normalization constant for coefficient of determination. For a surrogate performing worse than $\hat{f}^{\bar{y}}$, the right-hand side of the equation results in a value greater than 1 and, thus, R^2 becomes negative. If the surrogate performs equally good, the value is zero and otherwise positive. The upper bound of the R^2 metrics is 1, which could theoretically be reached with an MSE of zero. These characteristics turn out to be very suitable for defining a probability and thus have been the inspiration for balancing the surrogate bias.

The replacement probability ρ is given by bounding R^2 on the lower end to zero

$$\rho = \max \left(0, 1 - \frac{\text{MSE}(y, \hat{f})}{\text{MSE}(y, \hat{f}^b)} \right), \quad (4.2)$$

where \hat{f}^b represents a *baseline* predictor. The formula of ρ generalizes the definition of R^2 by considering an arbitrary baseline predictor \hat{f}^b instead of $\hat{f}^{\bar{y}}$. Given that R^2 has an upper bound of one and is at least zero, $\rho \in (0, 1)$ holds and thus is a valid probability.

But what does ρ as a metric defining the surrogate bias imply in the context of an algorithm's iteration? If the surrogate performs worse than \hat{f}^b ($\rho = 0$), no solution will be replaced. On the opposite, if the surrogate has no prediction error, all solutions will be surrogate biased. Even more importantly, if none of these two extreme cases occur, the value of ρ , and therefore the surrogate bias, will be adjusted proportionally to the accuracy of the model normalized by the performance of \hat{f}^b . In our implementation, we chose a k -nearest neighbor model ($k = n + 1$ where n represents the number of variables) as a baseline predictor \hat{f}^b and average ρ over the last five iterations (sliding window). For estimating the model's accuracy, we use all data points observed until the last generation as training and newly evaluated infill solutions as a validation set. In the initial iteration, where only the design of experiments and no infill solutions exist, a k -fold cross-validation is performed. Our experiments have shown that the surrogate prediction error assessment is essential and, thus, directly incorporating it into the algorithm design recommended.

4.4.1.4 Surrogate Management

The algorithm's outline has already shown that the surrogate model has to be fitted through data points and is used as a predictor for new infill solutions. In general, all matters related to fitting or updating a surrogate are referred to as surrogate management. It is noteworthy that in practice, not only single but multiple surrogates are recommended to provide a more robust model with less approximation error. With multiple surrogates, we refer not only to the type of surrogate but also concrete hyper-parameters. In our implementation, in total 15 surrogates, consisting of the model types RBF [34] and Kriging [35], are validated. The hyper-parameters instantiate models with different mean functions, kernel, and noise. Finally, the model with the highest ρ value is chosen.

On the one hand, an increasing number of points from optimization increase the time spent for surrogate management and, on the other hand, can lead to precision issues. The precision issues are caused by solutions very close to each other in the design space. Thus, we employ an ϵ -clearing approach, which always selects the solution with the smallest function value and then clears all solutions with less than ϵ distance to it ($\epsilon = 0.005$). We reduce the overall amount of points by only considering the 200 best solutions from the ϵ -cleared solution set.

4.4.2 Experimental Results

This study focuses on computationally expensive functions by limiting the evaluation budget for test problems. This is a commonly used principle in surrogate-assisted research, especially for more general approaches that need to be tested on several optimization problems. In this study, we have limited the function evaluations to 200-300 and considered problems with up to 10 variables. For comparison, we use well-known test problems, such as Sphere, Ackley, Rosenbrock, and others from the single-objective BBOB test suite [233]. To demonstrate the generalizability of PSAF, we (i) conduct an experiment focusing on the most suitable hyper-parameter combination, (ii) investigate the impact of dynamically determining the surrogate-bias by ρ and (iii), lastly, compare the proposed framework of surrogate-assisted algorithms with other recently proposed methods for computationally expensive problems.

4.4.2.1 What are suitable values for $(\alpha, \beta$ and $\rho)$?

In our first experiment, ρ is kept fixed and not updated. This shall give insights if ρ is a problem-dependent variable and, in fact, benefits from being updated based on the surrogate’s prediction error. Moreover, the impact of α and β on an algorithm’s performance shall be of interest.

For this hyper-parameter study, we select CMA-ES [112] as a baseline algorithm. We normalize each variable between zero and one to avoid any scaling irregularities and initialize the algorithm with a standard deviation of $\sigma = 0.15$. The algorithm’s initial starting point is determined by the best solution found by generating 20 points using Latin Hypercube Sampling [234]. We employ grid-

based optimization by setting the hyper-parameters $\alpha \in (1, 2, 3, 5, 10)$, $\beta \in (0, 5, 10, 20, 30, 40, 50)$ and $\rho \in (0.1, 0.2, \dots, 0.9, 1.0)$ for PSAF-CMA-ES. Because $\beta = 0$ makes the value of ρ irrelevant, there is no need to consider any run with $\rho = 0$ in addition. Because of the stochastic nature of the algorithm, we execute each parameter combination 11 times. This resulted in 60,588 runs in total for all test problems. As a performance criterion, we address the so-called anytime performance $f^{(\text{any})}$ of the algorithm and calculate the *integral* of the convergence curve based on the gap to the optimum $f^{(\text{gap})} = f - f^{(\text{opt})}$. Measuring the convergence and not only the final function value addresses the desire of a surrogate-assisted algorithm converging as quickly as possible with a very limited function evaluation budget.

In Figure 4.5 results of the hyper-parameter experiment for three exemplary optimization problems are shown in the form of a parallel coordinate plot [235]. The first three vertical lines represent the parameters α , β , and ρ , and the last the performance metric $f^{(\text{any})}$, relative to the baseline algorithm CMA-ES. Thus, the baseline algorithm's performance (blue) always ends up being 1.0, and the resulting values indicate the proportional improvement/deterioration. Moreover, the best performing parameter combination (red) and the second to tenth best (yellow) are highlighted.

(i) One can observe that adding surrogate bias has successfully improved the baseline algorithm's performance. PSAF achieved values less than one for almost all parameter combinations and improved the baseline algorithm's performance. Moreover, for the most suitable hyper-parameter values, PSAF showed a remarkable improvement by reducing the convergence integral to 30%.

(ii) One might think that introducing a strong bias in the β -phase makes the α -phase irrelevant. However, results indicate that it is beneficial to employ pre-filtering. The surrogate-influence in the α phase is applied no matter how good the surrogate performs but only provides surrogate influence to a specific extent. Nevertheless, more experiments need to be conducted to determine the most suitable value for α across all problems.

(iii) It becomes evident that for sphere and for rosenbrock rather large values of β and ρ and thus a stronger surrogate-bias are a better choice. This is in contrast to bbob-f07-1 where less surrogate involvement has turned out to be more effective. Moreover, even for a relatively simple

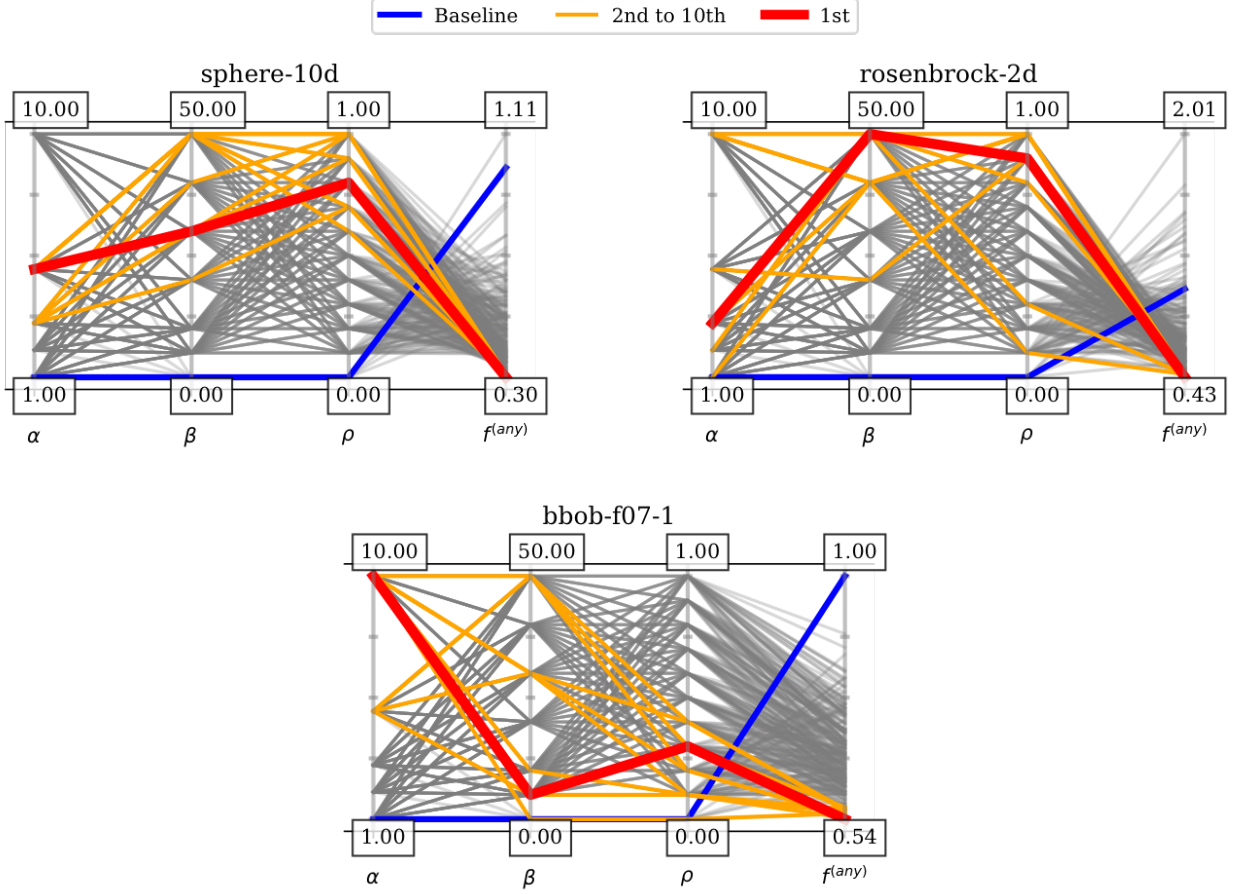


Figure 4.5: Hyper-parameter Analysis for PSAF-CMA-ES with varying α , β and ρ . Shown are the baseline algorithm CMA-ES (blue), the 2nd to 10th best (orange), and the best (red). The performance $f^{(any)}$ is normalized with respect to the baseline algorithm.

10-dimensional quadratic function, the surrogate may not be used 100% ($\rho = 1.0$) of the time. Analysis has shown this can be attributed to the limited number of points initially. Even for problems with almost no complexity, a small initial number of designs of experiments (here 20) requires the baseline algorithm to do some exploration until the surrogate starts to recognize the characteristics of the function and has a suitable accuracy.

(iv) Besides visualization, we have also performed a ranking-based analysis to find suitable parameter combinations. Thus, we have averaged the ranking in *percentage* across all problems. For instance, rank 30 out of 307 results in a value of ≈ 0.0977 . The average percentage ranks with their standard deviations are shown in Table 4.2. Results indicate that an α value between 5 to 10, a β value between 30 to 50 and a value of ρ between 0.3 to 0.5 perform best.

Table 4.2: Rankings of best performing hyper-parameters.

α	β	ρ	norm. rank	
			mean	std
10	40	0.4	0.2485	0.1556
5	40	0.5	0.2485	0.1949
10	50	0.3	0.2586	0.1664
10	30	0.3	0.2595	0.1914
5	40	0.3	0.2606	0.1728

4.4.2.2 Is it beneficial to update ρ each iteration?

Our next study addresses the impact of updating ρ in each iteration. The relatively small ρ values found to perform best might indicate that trusting the surrogate too much slows down the overall convergence. Thus, the effect of updating ρ in each iteration based on the surrogate’s prediction error should be investigated (Algorithm 4.2 and Section 4.4.1.3). Considering the insights gained from the hyper-parameter study, we define an upper bound for ρ , determining the maximum influence of the β -phase to a reasonable value of $\rho^{(\max)} = 0.7$. Moreover, we have used good-performing parameter combinations from the previous hyper-parameter study. The experiment reveals that an update of ρ performs significantly better than the best parameter combination from before and is, thus, recommended (see Table 4.3).

Table 4.3: Ranking with adaptive ρ .

α	β	ρ	norm. rank	
			mean	std
10	40	adaptive	0.1375	0.1831
10	40	0.4	0.2485	0.1556

4.4.2.3 How does PSAF perform compared to other surrogate-based algorithms?

For the remainder of this study, we fix the hyper-parameters to a $\alpha = 10$, $\beta = 40$ and perform a dynamic update of ρ with $\rho^{(\max)} = 0.7$. So far, our experiments have been based on CMA-ES to avoid an immense amount of runs for drawing conclusions about suitable hyper-parameters. However, for a comparison with other methods we have applied PSAF to the following well-known

population-based algorithms besides CMA-ES [112]: DE [220], PSO [118] with adaptive $c = 1$ and $c = 2$ [236] and a standard genetic algorithm [1]. For all algorithms, the population size and number of infills solutions (or, depending on the algorithm, called particles or offsprings) have been set to 10. We have used the standard implementations of the algorithms mentioned above available in pymoo [29]. All other hyperparameters are set to each algorithm’s default settings.

First, we like to confirm that PSAF improves the convergence of the considered baseline algorithms on various test problems. Figure 4.6 shows the convergence plots (averaged over 11 runs) of a variety of single-objective optimization problems. The PSAF variants are plotted using straight and the baseline algorithms with dashed lines. The convergence curves demonstrate the superiority of surrogate-assisted approaches across all test problems except for `bbob-f04-1-10d`. We attribute the superiority of PSO to the problem complexity and the fact that no algorithm can converge with the limited evaluation budget of 300.

Second, the performance compared to other surrogate-based algorithms shall be demonstrated. As a comparison we have chosen a standard EGO implementation from GPyOpt [237] (with 10 infill points in each iteration), a recently proposed method called ϵ -shotgun [238] (with a batch size of 10 and $\epsilon = 0.1$), and lqCMAES [232]. First, one can observe that lqCMA-ES, based on a quadratic model approximation, converges closer to the optimum *if* a solution near the optimum is found. Nevertheless, for problems where this is not the case, PSAF variants show superior performance. Extending PSAF to perform a local search using a quadratic model might show similar convergence behavior near an optimum. Moreover, EGO and ϵ -shotgun are outperformed by almost all PSAF variants except for `bbob-f05-1-10d` where a solution close to the optimum is found right away. Comparing algorithms of the PSAF framework with itself does allow to declare no clear winner. Whereas PSAF-PSO seems to perform well for most problems, PSAF-CMA-ES converges faster for the problems with only two variables. Altogether, considering an algorithm with a gap to the optimum of less than 10^{-6} as converged, at least one PSAF variant was better 50% (6/12) and equally good 30% (4/12) of the time. This can be considered a remarkable achievement for a generalizable framework.

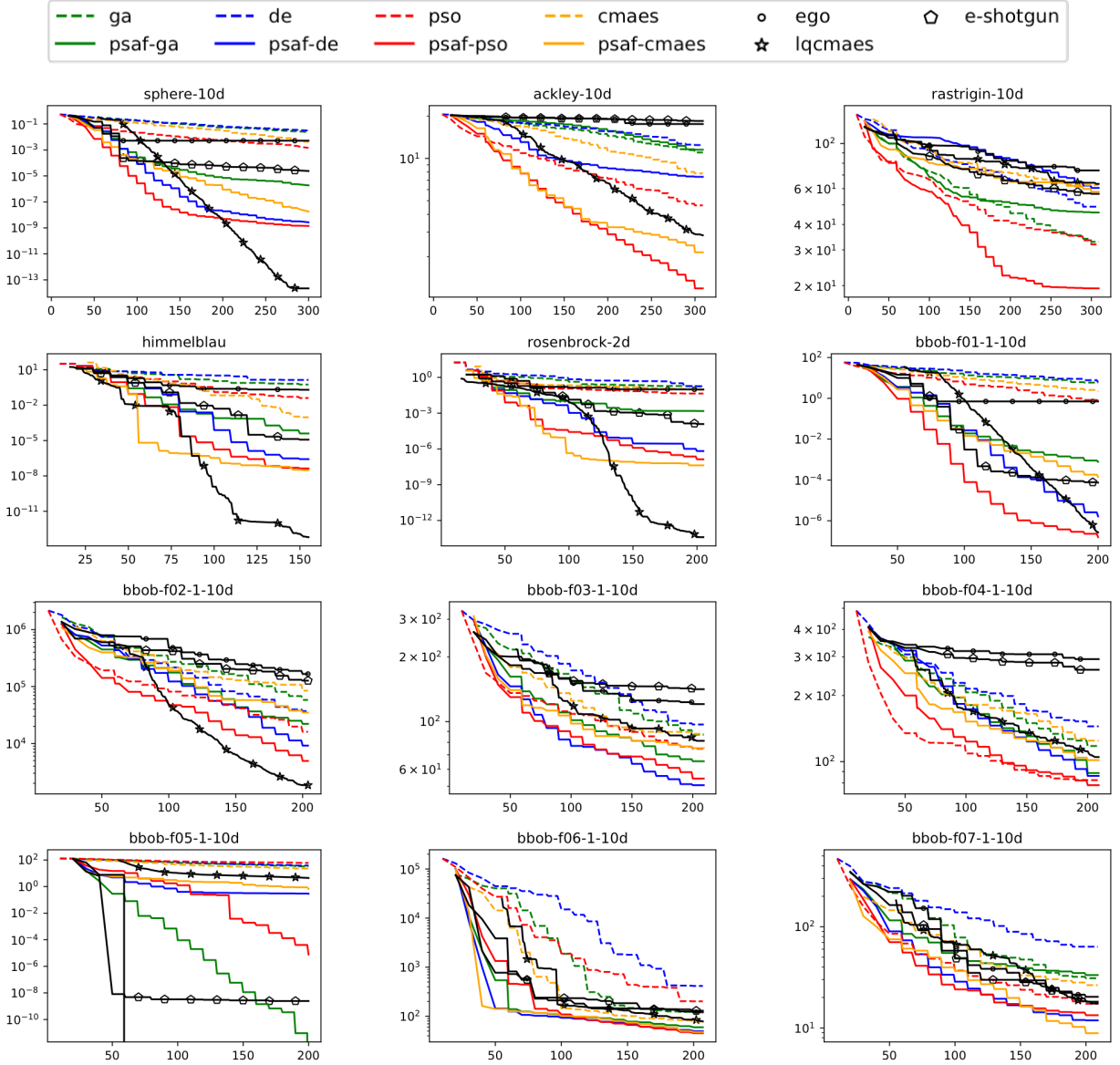


Figure 4.6: Comparison of the average performance of PSAF with the original algorithms and other surrogate-based algorithms.

4.4.3 Summary of Section 4.4

In this section, we have proposed a framework of generalized probabilistic surrogate-assisted algorithms. The idea is based on improving the convergence of an existing algorithm by incorporating a surrogate's knowledge. The concept consists of a surrogate's influence through a tournament-based procedure with α competitors and a stronger surrogate's bias by using solutions with probability ρ derived from continuing the optimizing for β iterations on the surrogate. Experiments with a

parametric study on α , β , and ρ have shown that the proposed approach effectively improves the convergence behavior on a variety of problems. While $\alpha = 10$ and $\beta = 40$ worked the best overall, we have presented an adaptive procedure of updating ρ depending on the surrogate’s prediction error inspired by the well-known R^2 metric. PSAF variants of CMA-ES, DE, GA, and PSO have shown competitive performance compared to other surrogate-based algorithms. Applying PSAF to other variable types to further demonstrate the approach’s capabilities will be part of future work. Additionally, the effect of a local search to improve the convergence behavior near the optimum is worth investigating. Other interesting future studies for PSAF are extensions to handle constraints and multiple objectives. This will require a suitable baseline algorithm and a modification of ρ estimation based on more than one surrogate. Altogether, the proposed probabilistic surrogate-assisted concept shall pave the way for new algorithms. PSAF allows making use of existing algorithms’ benefits to solve computationally expensive problems efficiently using a surrogate. Thus, this shall be an alternative to the widely-used *fit-and-optimize* method used in EGO and other algorithms.

4.5 Generalized Probabilistic Surrogate-Assisted Framework (GPSAF)

Next, PSAF shall be extended to be suitable to handle multiple objectives and constraints. Generalized probabilistic surrogate-assisted framework (GPSAF) follows the two-phase concept as PSAF. However, the α -phase and the β -phase now have to consider multiple criteria when comparing solutions.

4.5.1 Methodology

Before describing the responsibilities and modifications of each of the phases, the outline of the algorithm is discussed (see Algorithm 4.3). Before any surrogate can be fit, a solution archive A is initialized by some design of experiments $A.X$ are generated in a space-filling manner. A good spread of solutions is recommended to allow surrogates to capture the overall fitness landscape as accurately as possible. $A.X$ is evaluated using exact solution evaluation (ESE) resulting in $A.F$ and $A.G$ (Line 2). Then, while the number of evaluations is less than the maximum solution

evaluation budget $ESE^{(\max)}$, infill solutions $P.X$ are generated by calling the non-deterministic *infill* method of the baseline algorithm Φ . The default execution of algorithm Φ would immediately evaluate $P.X$ using ESE and directly feed the solutions back to the algorithm by executing $\Phi.advance(P.X, P.F, P.G)$ (Line 35 and 36). However, instead of doing so, GPSAF modifies $P.X$ in a way to be influenced and biased by surrogates (Line 6 to 30) and advances the algorithm in the end of the iteration (Line 35 to 36). After having estimated the surrogate error and fitted the surrogates for objective and constraint functions, the α -phase adds surrogate influence to $P.X$ by replacing solutions being predicted to be better (Line 8 to 14). Thereafter, the β -phase runs algorithm Φ for multiple generations (evaluations only on ASE) and assigns each solution to its closest $P.X$. For each of the resulting candidate solution pools $U[j]$ assigned $P[j]$ a probabilistic tournament determines the winning candidate (Line 15 to 30). Afterward, the replacement phase takes place where either the solution originating from the α -phase $P[j]$ is kept or replaced with $U[j]$ from the β -phase (Line 31 to 34). The solutions set to $P.X$ are evaluated, and the algorithm Φ is advanced (Line 35 and 36). Finally, the prediction error is updated before starting the next iteration, and the newly evaluated solutions are added to archive A .

4.5.1.1 Generalized α and β -phase

For single-objective optimization, the α -phase has already been described (see Section 4.4.1.1). There, the comparison of the two solutions is based only on one single objective value. For the more generic version with constraints and objectives, the winner of each solution pool is determined as follows: if *all* solutions are infeasible, select the least infeasible solution; otherwise, select a non-dominated solution (break ties randomly). For both the constraint and objective values, only ASEs are used. Otherwise, the α -phase remains the same, including its responsibilities and mechanics.

Analogously to PSAF, GPSAF further increases the surrogate's impact by looking β iterations into the future through calling *infill* and *advance* of the baseline algorithm repetitively. To obtain the β -solution for constrained multi-objective problems, we use a so-called probabilistic knockout tournament (PKT) to select solutions from each cluster with the goal of self-adaptively exploiting

Algorithm 4.3: GPSAF: Generalized Probabilistic Surrogate-Assisted Framework

Input : Algorithm Φ , Surrogate Tournament Pressure α (≥ 1), Number of Simulated Iterations β (≥ 0), Replacement Probability Exponent γ , Maximum Number of Solution Evaluations $ESE^{(\max)}$

```
/* Sample Design of Experiments (DOE) */
1 A  $\leftarrow \emptyset$ ; P  $\leftarrow \emptyset$ ; Q  $\leftarrow \emptyset$ ; U  $\leftarrow \emptyset$ ; e  $\leftarrow \emptyset$ 
2 A.X  $\leftarrow$  doe(); A.F, A.G  $\leftarrow$  evaluate(A.X)
3 while size(A) < ESE(max) do
    /* Infill sols. from baseline algorithm */
    4 P.X  $\leftarrow \Phi$ .infill()
    /* Estimate error - only initially */
    5 if e =  $\emptyset$  then e  $\leftarrow$  estm_error(A.X, A.F, A.G);
    /* Surrogates for each obj. and constr. */
    6 S  $\leftarrow$  fit(A.X, A.F, A.G)
    7 P. $\hat{F}$ , P. $\hat{G}$   $\leftarrow$  S.predict(P.X)
    /* Surrogate Influence ( $\alpha$ ) */
    8 foreach k  $\leftarrow$  2 to  $\alpha$  do
    9     Q.X  $\leftarrow \Phi$ .infill()
    10    Q. $\hat{F}$ , Q. $\hat{G}$   $\leftarrow$  S.predict(Q.X)
    11    foreach j  $\leftarrow$  1 to size(Q) do
    12        if not dominates(P[j], Q[j]) then P[j] = Q[j];
    13    end
    14 end
    /* Surrogate Bias ( $\beta$ ) */
    15  $\Phi'$   $\leftarrow$  copy( $\Phi$ )
    16 U  $\leftarrow \emptyset$ 
    17 foreach k  $\leftarrow$  1 to  $\beta$  do
    18     Q.k  $\leftarrow$  k
    19     Q.X  $\leftarrow \Phi'$ .infill()
    20     Q. $\hat{F}$ , Q. $\hat{G}$   $\leftarrow$  S.predict(Q.X)
    21     foreach j  $\leftarrow$  1 to size(Q) do
    22         i  $\leftarrow$  closest(P.X, Q[j].X)
    23         U[i]  $\leftarrow$  U[i]  $\cup$  Q[j]
    24     end
    25      $\Phi'$ .advance(Q.X, Q. $\hat{F}$ , Q. $\hat{G}$ )
    26 end
    27 V  $\leftarrow$  list()
    28 foreach j  $\leftarrow$  1 to size(U) do
    29     V  $\leftarrow$  V  $\cup$  prob_knockout_tourn(U[j])
    30 end
    /* Replacement ( $\gamma$ ) */
    31 foreach j  $\leftarrow$  1 to size(P) do
    32      $\rho$   $\leftarrow$  repl_prob(U[j], U,  $\gamma$ )
    33     if rand() <  $\rho$  then P[j]  $\leftarrow$  V[j];
    34 end
    /* Evaluate on ESE */
    35 P.F, P.G  $\leftarrow$  evaluate(P.X)
    /* Prepare next iteration of GPSAF */
    36  $\Phi$ .advance(P.X, P.F, P.G)
    37 e  $\leftarrow$  update_error(P.F, P.G, P. $\hat{F}$ , P. $\hat{G}$ )
    38 A  $\leftarrow$  A  $\cup$  P
39 end
```

surrogates. The goal is to use surrogates more when they provide accurate predictions but use them more carefully when they provide only rough estimations. Necessary for generalization, PKT also applies to problems with multiple objectives and constraints, often with varying complexities and

surrogate errors to be considered.

Generally, we define PKT as a subset selection of k solutions from a set of solutions C by applying *pairwise* comparisons under noise as shown in Algorithm 4.4. Initially, the solution set C to select from is shuffled to randomize the matches (Line 1). If the current number of participants $|C^{(t)}|$ is odd, a random solution is chosen to compete twice (Line 4). Each competition occurs under noise, based on the current prediction error of the surrogates. The noise is added to each objective and constraint independently before comparing the solutions. After adding the noise, the comparison is identical to the subset selection explained in Section 4.5.1.1 (feasibility, dominance, random tie break) with two competitors ($\alpha = 2$). The winner of each round moves on to the next and is added to $C^{(t+1)}$ (Line 8). If too many solutions have been eliminated, randomly choose some losers from the last round (Line 13). This results in a set of solutions of size k being returned as tournament winners under noise. The design of PKT applies to the most general case of constrained multi-objective optimization because the selection procedure can be reduced to a comparison of two solutions.

Back to the cluster-wise selection in the β -phase where PKT is executed with $k = 1$ to obtain a winner for each solution set U_j . An example with five iterations ($\beta = 5$) and four infill solutions X_1, X_2, X_3 , and X_4 is illustrated in Figure 4.4. Calling the *infill* and *advance* function of the baseline algorithm results in five solution sets (β_1 to β_5) with four solutions each. The advancement of multiple iterations is based on ASEs. In each iteration, all solutions are directly assigned to the closest X_i solution from the α -phase forming the cluster U_i . The cluster search pattern division is essential to preserve diversity. For each cluster, a winner V_i is declared by performing the PKT. For instance, in this example, X_1 has four solutions in U_1 where one from the fourth iteration β_4 is finally selected. At the end of the β -phase, each cluster U_i has at most one solution V_i to be assigned to (some clusters may stay empty because no solutions are assigned to it).

Algorithm 4.4: Probabilistic Knockout Tournament (PKT)

Input : Solution Set C , Prediction errors e , Number of winners k

```
1  $C^{(1)} \leftarrow \text{shuffle}(C)$ 
2  $t \leftarrow 1$ 
3 while  $|C^{(t)}| > k$  do
4   if  $|C^{(t)}|$  is odd then  $C^{(t)} \leftarrow C^{(t)} \cup \text{rselect}(C^{(t)}, 1);$ 
5    $C^{(t+1)} \leftarrow \emptyset$ 
6   foreach  $i \leftarrow 1$  to  $|C^{(t)}|/2$  do
7      $w \leftarrow \text{compare\_noisy}(C_{2i}^{(t)}, C_{2i+1}^{(t)}, e)$ 
8      $C^{(t+1)} \leftarrow C^{(t+1)} \cup w$ 
9   end
10   $t \leftarrow t + 1$ 
11 end
12 if  $|C^{(t)}| < k$  then
13    $C^{(t)} \leftarrow C^{(t)} \cup \text{rselect}(C^{(t-1)} \setminus C^{(t)}, |C^{(t)}| - k)$ 
14 end
15 return  $C^{(t)}$ 
```

4.5.1.2 Balancing the Exploration and Exploitation (γ)

PSAF has defined the probability ρ for choosing between either keeping the α -solution or replacing it with the β -solution based on the error of the surrogate model. However, now multiple surrogate models exist, and for instance, the model for the first objective might have almost no prediction error, but the one for the second objective be highly inaccurate. Thus a different logic has to be implemented.

GPSAF uses another particularly useful piece of information for this decision: the distribution of assigned solutions across clusters. The search pattern derived from surrogates with a high-density area indicates a region of interest. Thus, we propose to set the replacement probability to

$$\rho = \left(\frac{|U_j|}{\max_j |U_j|} \right)^\gamma. \quad (4.3)$$

The denominator $\max_j |U_j|$ normalizes the number of assigned points with respect to the points in the current cluster $|U_j|$. The exponent γ can be used to control the importance of the distribution and was kept constant at $\gamma = 0.5$. The cluster with the highest density is always chosen from the β -phase because the nominator and denominator will be equal. This will always be the case for

baseline algorithms returning only *one* infill solution where stronger surrogate bias is generally desirable. After the replacement, the solutions will finally be sent to the time-consuming solution evaluation.

4.5.1.3 Surrogate Management

Besides using surrogates in an algorithmic framework, more needs to be said about the models themselves. First, one should note that only the predictions of data points need to be provided by surrogates and no additional error estimation (the error estimates are kept track of by our method directly). Not requiring an error estimation does not limit the models to a specific type, unlike other surrogate-based algorithms. Second, each of the objective and constraint functions is modeled independently, known as *MI* in the surrogate usage taxonomy in [23]. Even though modeling all functions increases the algorithmic overhead, it prevents larger prediction errors through complexity aggregations of multiple functions. Third, a generic framework for optimizing computationally expensive functions requires a generic surrogate model implementation. Clearly, some model types are more suitable for some problems than others. Thus, to provide a more robust framework, each function is approximated with a set of surrogates, and the best one is chosen to be used. The surrogate types in this section consist of the model types RBF [34] and Kriging [35], both initialized with different hyper-parameters (normalization, regressions, kernel). A pre-normalization step referred to as PLOG [226] is attempted and selected, if well-performing, for constraint functions. Two metrics assess the performance of a model: First, Kendall Tau Distance [239] comparing the ranking of solutions being less sensitive to outliers with a large prediction error; second, the Maximum Absolute Error (MAE) to break any ties. The value of MAE is also used as an error approximation when noise is added to individuals. The error estimation in the first iteration is based on k-fold cross-validation ($k = 5$) to get a rough estimate of how well a surrogate can capture the function type. The performance metrics are updated in each iteration by fitting a surrogate based on all solutions seen so far (training set) and assessing their error on the newly evaluated solutions (test set). Finally, a moving average of five iterations to avoid a smooth and more robust

estimation provides the data for selecting the best surrogate and estimating the prediction error for each objective and constraint.

4.5.2 Experimental Results

In this section, we present the performance of GPSAF applied to various population-based algorithms solving unconstrained and constrained, single- and multi-objective optimization problems. Proposing an optimization framework requires comparing a group of algorithms, which is not a trivial task itself. Benchmarking is further complicated when non-deterministic algorithms are compared, in which case not only a single but multiple runs need to be considered.

For a fair comparison of optimization methods across test problems and to measure the impact of GPSAF on a baseline algorithm, we use the following ranking-based procedure:

- i) *Statistical Domination:* After collecting the data for each test problem and algorithm ($\mathcal{A} \in \Omega$) from multiple runs, we perform a pairwise comparison of performance indicators (PI) between all algorithms using the Wilcoxon Rank Sum Test ($\alpha = 0.05$). The null-hypothesis H_0 is that no significant difference exists, whereas the alternative hypothesis is that the performance indicator of the first algorithm ($\text{PI}(\mathcal{A})$) is smaller than the one of the second ($\text{PI}(\mathcal{B})$). For single-objective optimization, the PI function consists of the gap to the optimum (if known) or the best function value found. For multi-objective optimization IGD [240] (if optimum is known) or Hypervolume [241] is used. The dominance function between two algorithms, \mathcal{A} and \mathcal{B} , is then defined by

$$\phi(\mathcal{A}, \mathcal{B}) = \mathbf{RANKSUM}(\text{PI}(\mathcal{B}), \text{PI}(\mathcal{A}), \text{alt}='less'), \quad (4.4)$$

where the function $\phi(\mathcal{A}, \mathcal{B})$ returns zero if the null hypothesis is accepted or a one if it is rejected.

- ii) *Number of Dominations:* The performance $P(\mathcal{A})$ of algorithm \mathcal{A} is then determined by the

number of methods that are dominating it:

$$P(\mathcal{A}) = \sum_{\substack{\mathcal{B} \in \Omega \\ \mathcal{A} \neq \mathcal{B}}} \phi(\mathcal{B}, \mathcal{A}). \quad (4.5)$$

This results in a domination number $P(\mathcal{A})$ for each method, which is zero if no other algorithm does not outperform it.

- iii) *Ranking*: Finally, we sort the methods by their $P(\mathcal{A})$. This may result in a partial ordering with multiple algorithms with the same $P(\mathcal{A})$ values. In order to keep the overall sum of ranks equal, we assign their average ranks in case of ties. For instance, let us assume five optimizations methods A, B, C, D, and E: algorithm A outperforms all others; between the performances of B, C, and D, no significant difference exists; E performs the worst. In this case, method A gets rank 1, the group of methods B, C, and D, rank $(2 + 3 + 4)/3 = 9/3 = 3$, and E ranks 5. Averaging the ranks for ties penalizes an optimization method for being dominated by the same amount of algorithms as others and keeps the rank sum for each problem the same.

This conveniently provides a ranking for each test problem. To evaluate the performance of a method on a test suite, we finally average the ranks across problems. If an algorithm fails to solve a specific problem for all runs, it gets the maximum rank and becomes the worst performing algorithm. Otherwise, all failing runs will be ignored (this has only rarely happened for a competitor algorithm to compare with). The ranks are used to compare the performances of methods in this manuscript, the values of the performance indicators for the methods on all test problems can be found in the Supplementary Document. Each algorithm has been executed 11 times on each test problem. If not explicitly mentioned in the specific experiment, the total number of solution evaluations has been set to $\text{ESE}^{(\max)} = 300$. For some simpler constrained problems, even fewer evaluations have been used. A relatively limited evaluation budget also means that more complicated problems might not be solved (near) optimally. However, a comparison of how well an algorithm has performed imitates the situation researchers face in practice. If the number of variables is not fixed, the

number of variables is fixed to 10. The results are presented in ranking tables where the overall best performing algorithm(s) are highlighted with a gray cell background for each ranking-based comparison for a test problem. The best-performing ones in a group are shown in bold.

Moreover, some more details about our implementation shall be said. For the baseline algorithms, we use implementations of population-based algorithms available in the well-known multi-objective optimization framework pymoo¹ [29] developed in Python. For all methods, the default parameters provided by the framework are kept unmodified, except the population size (=20) and the number of offsprings (=10) to create a more greedy implementation of the methods. The surrogate implementation of Kriging is based on a Python clone² of DACEFit [55] originally implemented in Matlab. The RBF models are a re-implementation based on [226]. The hyper-parameters of GPSAF were determined through numerous empirical experiments during the algorithm development. A reasonable and well-performing configuration given by $\alpha = 30$, $\beta = 5$, and $\gamma = 0.5$ is fixed throughout all experiments.

4.5.2.1 (Unconstrained) Single-Objective Optimization

The first experiment investigates the capabilities of GPSAF for improving the performance of existing algorithms on unconstrained single-objective problems. We use the BBOB test problems (24 functions in total) available in the COCO-platform [233] which is a widely used test suite with a variety of more and less complex problems. Four well-known population-based optimization methods, DE [107], GA [1], PSO [118], and CMAES [112] serve as baseline optimization algorithms and their GPSAF variants provide a surrogate-assisted version. The results are compared with four other surrogate-assisted algorithms, SACOSO [222], SACC-EAM-II [242], SADESammon [243], SAMSO [244] available in the PlatEMO [245] framework. The rankings from the experiment are shown in Table 4.4. First, one can note that GPSAF outperforms the other four existing surrogate-assisted algorithms. One possible reason for the significant difference could be their development for a different type of test suite (for instance, problems with a larger number of variables). In

¹<http://pymoo.org> (Version 0.5.0)

²<https://pypi.org/project/pydacefit/>

Table 4.4: A comparison of DE, GA, PSO, and CMAES with their GPSAF variants on unconstrained single-objective problems with four other surrogate-assisted algorithms. The rank of the best performing algorithm in each group is shown in bold. The overall best performing algorithm for each problem is highlighted with a gray shade.

Problem	DE	GPSAF-DE	GA	GPSAF-GA	PSO	GPSAF-PSO	CMAES	GPSAF-CMAES	SA COSO	SACC-EAM-II	SADE Sammon	SA MSO
f01	11.0	4.0	7.5	2.5	5.5	1.0	5.5	2.5	7.5	9.5	12.0	9.5
f02	10.0	4.0	4.0	1.0	4.0	2.0	7.5	6.0	12.0	7.5	10.0	10.0
f03	8.5	3.0	5.0	1.5	7.0	1.5	8.5	5.0	10.0	5.0	11.5	11.5
f04	8.5	4.0	4.0	2.0	4.0	1.0	6.5	6.5	10.0	8.5	11.0	12.0
f05	5.5	1.0	7.5	3.0	5.5	2.0	7.5	4.0	11.0	9.5	9.5	12.0
f06	7.0	7.0	2.0	2.0	4.5	4.5	9.0	2.0	10.0	7.0	11.0	12.0
f07	9.5	5.5	5.5	2.0	5.5	2.0	5.5	2.0	9.5	8.0	11.0	12.0
f08	8.0	6.0	8.0	2.0	5.0	2.0	4.0	2.0	10.0	8.0	11.0	12.0
f09	10.0	3.5	8.0	3.5	3.5	3.5	7.0	3.5	3.5	9.0	11.5	11.5
f10	10.5	5.5	5.5	1.5	5.5	1.5	5.5	5.5	10.5	5.5	10.5	10.5
f11	10.5	3.5	7.0	1.5	7.0	3.5	7.0	7.0	10.5	1.5	12.0	7.0
f12	2.5	6.5	6.5	6.5	2.5	2.5	2.5	6.5	10.0	9.0	11.0	12.0
f13	7.5	5.0	7.5	2.5	5.0	1.0	5.0	2.5	10.0	9.0	11.0	12.0
f14	9.5	5.0	7.5	2.0	5.0	2.0	5.0	2.0	9.5	7.5	11.0	12.0
f15	10.0	3.5	6.5	1.5	6.5	1.5	6.5	3.5	9.0	6.5	11.0	12.0
f16	9.0	6.0	6.0	2.0	9.0	2.0	4.0	2.0	11.5	6.0	11.5	9.0
f17	9.5	5.0	7.0	3.0	7.0	3.0	3.0	1.0	11.0	7.0	9.5	12.0
f18	9.5	4.0	6.0	2.0	6.0	2.0	6.0	2.0	9.5	8.0	11.0	12.0
f19	11.0	5.0	10.0	1.0	5.0	5.0	8.5	5.0	5.0	2.0	8.5	12.0
f20	9.5	2.5	7.0	2.5	2.5	2.5	5.5	5.5	9.5	8.0	11.5	11.5
f21	8.5	7.0	3.5	3.5	3.5	3.5	3.5	3.5	10.0	8.5	11.5	11.5
f22	9.5	6.0	6.0	2.5	6.0	2.5	2.5	2.5	9.5	8.0	11.0	12.0
f23	10.0	5.5	5.5	1.5	5.5	5.5	11.0	12.0	9.0	1.5	5.5	5.5
f24	10.0	2.0	8.0	2.0	4.0	2.0	8.0	5.5	8.0	5.5	11.0	12.0
Total	8.958	4.583	6.292	2.292	5.188	2.479	6.021	4.146	9.417	6.896	10.667	11.062

this test suite, some problems are rather complicated and exploiting the surrogate too much will cause the optimizer to be easily trapped in local optima. Also, we contribute the efficiency of GPSAF to the significant effort for finding the most suitable surrogate. The order of relative rank improvement is given by GA ($6.292/2.292 = 2.7452$), PSO (2.0927), DE (1.9546), and for CMAES (1.4522). Besides GPSAF-GA having the biggest relative rank improvement, it also is the overall best performing algorithm in this experiment, closely followed by GPSAF-CMAES. Altogether, a significant and quite remarkable improvement is achieved by applying GPSAF for (unconstrained) single-objective optimization.

4.5.2.2 Constrained Single-Objective Optimization

Rarely are optimization problems unconstrained in practice. Thus, especially for surrogate-assisted methods aiming to solve computationally expensive real-world problems, the capability of dealing

Table 4.5: A comparison of DE, GA, PSO, and ISRES with their GPSAF variants on constrained single-objective problems with SACOBRA – the current state-of-art algorithms for constrained optimization.

Problem	ESE ^(max)	DE	GPSAF-DE	GA	GPSAF-GA	PSO	GPSAF-PSO	GPSAF-ISRES	SACOBRA
G1	75	5.5	3.5	7.5	3.5	7.5	5.5	1.0	2.0
G2	300	3.5	7.0	1.0	3.5	6.0	3.5	8.0	3.5
G4	75	6.5	3.5	6.5	5.0	8.0	3.5	1.5	1.5
G6	75	7.0	4.0	7.0	4.0	7.0	4.0	1.5	1.5
G7	75	7.0	4.0	7.0	4.0	7.0	4.0	2.0	1.0
G8	100	7.0	4.5	7.0	4.5	7.0	3.0	1.0	2.0
G9	300	7.0	4.5	7.0	2.5	4.5	2.5	7.0	1.0
G10	300	8.0	3.5	6.5	3.5	6.5	3.5	3.5	1.0
G11	300	7.0	2.5	5.5	2.5	5.5	2.5	2.5	8.0
G12	300	6.0	4.5	8.0	4.5	7.0	3.0	1.5	1.5
G16	300	5.5	2.5	5.5	8.0	7.0	2.5	2.5	2.5
G18	300	7.0	3.5	7.0	3.5	7.0	3.5	3.5	1.0
G19	300	7.5	2.0	6.0	2.0	7.5	2.0	5.0	4.0
G24	300	7.5	4.5	7.5	4.5	6.0	2.0	2.0	2.0
Total		6.571	3.857	6.357	3.964	6.679	3.214	3.036	2.321

with constraints is essential. The so-called G-problems or G-function benchmark [246, 247] was proposed to develop optimization algorithms dealing with different kinds of constraints regarding the type (equality and inequality), amount, complexity, and result in feasible and infeasible search space. The original 13 test functions were extended in a CEC competition in 2006 [248] to 24 constrained single-objective test problems [249]. In this study, G-problems with only inequality constraints (and no equality constraints) are used. Besides the GPSAF variants of DE and GA, improved stochastic ranking evolutionary strategy (ISRES) [250] is applied to GPSAF. ISRES implements an improved mating strategy using differentials between solutions in contrast to its predecessor SRES [251]. ISRES follows the well-known $1/7$ rule, which means with a population size of μ individuals $7 \cdot \mu$ offsprings are created. For this study, GPSAF creates a steady-state variant of ISRES by using the proposed probabilistic knockout tournament to choose *one* out of the λ solutions. This ensures a fair comparison with SACOBRA [226] which also evaluates one solution per iteration. To the best of our knowledge, SACOBRA implemented in R [252] is currently the best-performing algorithm on the G problem suite. The constrained single-objective results are presented in Table 4.5. First, it is apparent that the GPSAF variants improve

the baseline algorithms. Only for G2, the genetic algorithm outperforms its and other surrogate-assisted variants, which we contribute to the very restricted feasible search space (also, this has shown to be a difficult problem for surrogate-assisted algorithms in [226]). Second, GPSAF-ISRES shows the best results out of all GPSAF variants. This indicates that it is beneficial if the baseline method has been proposed with a specific problem class in mind. Even though DE, GA, and PSO can handle constraints (for instance, naively using the parameter-less approach), there are known to not perform particularly well on complex constrained functions without any modifications. In contrast, ISRES has been tested on the G problems in the original study and proven to be effective. Furthermore, adding surrogate assistance to it has further improved the results. Third, GPSAF-ISRES shows competitive performance to the state-of-the-art algorithm SACOBRA. In this experiment, out of all 14 test problems: GPSAF variants were able to outperform SACOBRA four times and a baseline algorithm (GA) one time; five times the performance of at least one GPSAF variant was similar; four times SACOBRA has shown significantly better results. Altogether, one can say GPSAF has created surrogate-assisted methods competing with the state-of-the-art method for constrained single-objective problems.

4.5.2.3 (Unconstrained) Multi-Objective Optimization

Many applications have not one but multiple conflicting objectives to optimize. For this reason, this experiment focuses specifically on multi-objective optimization problems. As a test suite, we choose ZDT [253], a well-known test suite proposed when multi-objective optimization has gained popularity. Throughout this experiment, we set the number of variables to 10, except for the high multi-modal problem, ZDT4, where the number of variables is limited to 5. The WFG [254] test suit provides even more flexibility by being scalable with respect to the number of objectives. Here, we simply set the objective number to be two to create another bi-objective test suite. Moreover, the number of variables has been set to 10 where four of them are positional. The baseline algorithms NSGA-II [10], SMS-EMOA [255], and SPEA2 [256] are used as baseline algorithms. The results are compared with four other surrogate-assisted algorithms: AB-SAEA [137], KRVEA [225],

Table 4.6: A comparison of NSGA-II, SMS-EMOA, and SPEA2 with their GPSAF variants with four surrogate-assisted algorithms on bi-objective optimization problems.

Problem	NSGA-II	GPSAF-NSGA-II	SMS-EMOA	GPSAF-SMS-EMOA	SPEA2	GPSAF-SPEA2	AB-SAEA	K-RVEA	ParEGO	CSEA
ZDT1	9.0	2.5	9.0	2.5	9.0	2.5	6.0	5.0	2.5	7.0
ZDT2	9.0	1.5	9.0	6.0	9.0	5.0	3.0	4.0	1.5	7.0
ZDT3	9.0	4.5	9.0	4.5	9.0	4.5	4.5	1.0	2.0	7.0
ZDT4	6.5	2.0	6.5	2.0	6.5	2.0	9.0	6.5	10.0	4.0
ZDT6	7.5	3.0	9.5	6.0	9.5	3.0	5.0	3.0	1.0	7.5
WFG1	9.0	6.0	9.0	6.0	9.0	6.0	4.0	2.0	2.0	2.0
WFG2	7.0	1.5	8.0	9.0	4.5	4.5	4.5	1.5	10.0	4.5
WFG3	8.0	3.5	10.0	3.5	8.0	1.5	5.5	1.5	5.5	8.0
WFG4	6.5	1.5	9.0	5.0	6.5	1.5	3.5	3.5	9.0	9.0
WFG5	8.0	2.5	8.0	4.0	10.0	5.0	6.0	2.5	1.0	8.0
WFG6	9.0	2.0	10.0	6.5	4.5	2.0	2.0	4.5	6.5	8.0
WFG7	5.5	4.0	8.5	2.0	8.5	2.0	5.5	7.0	2.0	10.0
WFG8	7.5	2.5	10.0	5.0	9.0	2.5	2.5	6.0	2.5	7.5
WFG9	7.5	3.0	7.5	3.0	6.0	3.0	3.0	10.0	9.0	3.0
Total	7.786	2.857	8.786	4.643	7.786	3.214	4.571	4.143	4.607	6.607

Table 4.7: A comparison of NSGA-III, SMS-EMOA, and SPEA2 with their GPSAF variants with four surrogate-assisted algorithms on three-objective optimization problems.

Problem	NSGA-III	GPSAF-NSGA-III	SMS-EMOA	GPSAF-SMS-EMOA	SPEA2	GPSAF-SPEA2	AB-SAEA	K-RVEA	ParEGO	CSEA
DTLZ1	1.5	3.5	1.5	6.5	5.0	8.0	10.0	9.0	6.5	3.5
DTLZ2	8.5	2.0	8.5	2.0	8.5	2.0	6.0	4.0	5.0	8.5
DTLZ3	2.0	5.5	2.0	2.0	5.5	8.0	10.0	9.0	5.5	5.5
DTLZ4	6.5	6.5	9.0	6.5	6.5	3.5	2.0	1.0	10.0	3.5
DTLZ5	6.0	3.0	9.0	1.0	9.0	2.0	6.0	6.0	4.0	9.0
DTLZ6	7.5	5.5	7.5	3.5	9.0	5.5	1.5	1.5	10.0	3.5
DTLZ7	8.0	3.5	8.0	3.5	8.0	3.5	3.5	1.0	10.0	6.0
Total	5.714	4.214	6.5	3.571	7.357	4.643	5.571	4.5	7.286	5.643

ParEGO [126], CSEA [64] available in PlatEMO [245].

The results on the two multi-objective test suites are shown in Table 4.6. First, one can note that all surrogate-assisted algorithms outperform the ones without. This indicates that surrogate assistance effectively improves the convergence behavior. Second, GPSAF-NSGA-II performs the best with a rank of 2.893 and shows the best performance, followed by GPSAF-SPEA2, GPSAF-SMS-EMOA, and KRVEA. It is worth noting that ParEGO is penalized by being terminated for ZDT4 and WFG2, where the surrogate model was not able to be built.

To show the behavior of three-objective optimization problems, we have replaced NSGA-II with NSGA-III and run all algorithms on the DTLZ problems suite [257] test suite. The results are shown in Table 4.7. Whereas for most problems, the GPSAF variants outperform the baseline

algorithms, for DTLZ1 and DTLZ3, this is not the case. Both problems consist of multi-modal convergence functions, which causes a large amount of surrogate error. Thus, surrogate-assisted algorithms (including the four GPSAF is compared to) are misguided. This seems to be a vital observation deserving to be investigated in more detail in the future. Nevertheless, GPSAF improves the performance of baseline algorithms for the other problems. GPSAF-SMS-EMOA shows overall the best results in this experiment with an average rank of 2.786 followed by GPSAF-NSGA-III.

4.5.2.4 Constrained Multi-Objective Optimization

Lastly, we compare GPSAF on constrained multi-objective optimization problems which often occur in real-world optimization. The challenge in dealing with multiple objectives and constraints in combination with computationally expensive solution evaluations truly mimics the complexity of industrial optimization problems. We have compared our results with HSMEA [136] a recently proposed algorithm for constraint multi-objective optimization. With consultation of the authors, some minor modifications of the publicly available source code had to be made for dealing with computationally expensive constraints – as this is an assumption made in this study. The results on CDTLZ [258], BNH [259], SRN [260], TNK [261], and OSY [262] are shown in Table 4.8. Again, one can observe that the GPSAF variants consistently improve the performance of the baseline optimization methods. The only exception is C1-DTLZ1, where all methods could find no feasible solution, and thus, an equal rank is assigned. We contribute this to the complexity of the test problems given by the constraint violation and the multi-modality of the objective functions. For OSY and TNK, the GPSAF variants show a significantly better performance than HSMEA; for C3-DTLZ4, the performance is similar; and for C2-DTLZ2, BNH, and TNK, it performs better. Altogether, GPSAF-NSGA-II can obtain a better rank than HSMEA, but it is fair to say that for three out of the seven constrained multi-objective optimization problems, HSMEA is the winner. Nevertheless, GPSAF improved the performance of baseline algorithms and showed competitive results to another surrogate-assisted optimization method.

Table 4.8: A comparison of NSGA-III, SMS-EMOA, and SPEA2 with their GPSAF variants with four surrogate-assisted algorithms on constrained multi-objective optimization problems.

Problem	ESE ^(max)	NSGA-II	GPSAF-NSGA-II	SMS-EMOA	GPSAF-SMS-EMOA	SPEA2	GPSAF-SPEA2	HSMEA
C1-DTLZ1	300	4.0	4.0	4.0	4.0	4.0	4.0	4.0
C2-DTLZ2	300	4.5	4.5	4.5	4.5	4.5	4.5	1.0
C3-DTLZ4	300	5.0	1.5	5.0	5.0	5.0	5.0	1.5
BNH	100	5.5	2.5	7.0	4.0	5.5	2.5	1.0
SRN	100	6.0	3.0	6.0	3.0	6.0	3.0	1.0
TNK	100	6.0	2.0	6.0	2.0	6.0	2.0	4.0
OSY	300	5.0	1.5	5.0	3.0	5.0	1.5	7.0
Total		5.143	2.714	5.357	3.643	5.143	3.214	2.786

4.5.3 Summary of Section 4.5

This section has proposed a generalized probabilistic surrogate-assisted framework applicable to any type of population-based algorithm. GSPAF incorporates two different phases to provide surrogate assistance, one considering using the current state of the baseline algorithm and the other looking at multiple iterations into the future. In contrast to other existing surrogate-assisted algorithms, the surrogate search is not reduced to the final solutions on the surrogate, but the whole search pattern is utilized. Solutions are selected using a probabilistic tournament that considers surrogate prediction errors for objectives and constraints from the search pattern. GPSAF has been applied to multiple well-known population-based algorithms proposed for unconstrained and constrained single and multi-objective optimization. We have provided comprehensive results on test problem suites indicating that GPSAF competes and outperforms existing surrogate-assisted methods. The combination of GPSAF creating well-performing surrogate-assisted algorithms with its *simplicity* and *broad* applicability is very promising.

The encouraging results provide scope for further exploring generalized surrogate-assisted algorithms. One main challenge of a generalized approach is the recommendation of hyper-parameter configurations (α , β , ρ , or γ). The parameters have been set through empirical experiments; however, through the broad applicability, different mechanisms of baseline algorithms on very different optimization problems make it difficult to draw generally valid conclusions. A more systemic and

possibly resource-intensive study will provide an idea of how different hyper-parameter settings impact the performance of different algorithms. In addition, experiments investigating the sensitivity shall be especially of interest.

The focus of this study was to explore different types of problems with multiple objectives and constraints. Thus, the number of variables was kept relatively small as this is often the case for computationally expensive problems. Therefore, even though the search space dimensions do not directly impact the idea proposed in this section, it shall be part of a future study of how surrogate assistance performs for large-scale problems. Moreover, the number of solution evaluations per run has been set to 300, which allows using all solutions exhaustively for modeling without a large modeling overhead. However, more solution evaluations might be feasible for mediocre expensive optimization problems.

Nevertheless, this extensive explorative study on the use of surrogates in single and multi-objective optimization with and without constraints has indicated a viable new direction in congruence with existing emerging studies for a generic optimization methodology.

4.6 Summary of the Chapter

In this chapter, we have proposed a framework of generalized probabilistic surrogate-assisted algorithms. The idea is based on improving the convergence of an existing algorithm by incorporating a surrogate’s knowledge. First, we have proposed PSAF for computationally expensive *single-objective* problems where surrogate influences the solutions to be evaluated through a tournament-based procedure with α competitors. An even stronger surrogate’s bias is introduced by using solutions with probability ρ derived from continuing the optimizing for β iterations on the surrogate. Experiments with a parametric study on α , β , and ρ have shown that the proposed approach effectively improves the convergence behavior on various problems. Second, we have extended PSAF to a more general framework GPSAF which can handle multiple objectives and constraints. We have proposed comparing two solutions by using a constrained Pareto-dominance relation under uncertainty by introducing some error noise to the objectives and constraints. The selection of a

single solution from a set is defined based on the pairwise comparisons in a knockout tournament. Both tasks are incorporated into a generalized framework to use surrogates as assistance during convergence. Results indicate that GPSAF applied to eight different unconstrained and constrained, as well as single- and multi-objective algorithms, show a competitive performance. Altogether, the proposed probabilistic surrogate-assisted concepts shall pave the way for new algorithms. PSAF and GPSAF allow using existing algorithms' benefits to solve computationally expensive problems using surrogates efficiently. Thus, this could be an alternative to the widely-used *fit-and-optimize* method used in EGO and other algorithms.

CHAPTER 5

HETEROGENEOUS EXPENSIVE OBJECTIVES AND CONSTRAINTS

This chapter focuses on efficiently optimizing problems with heterogeneously expensive objectives and constraints. First, some examples demonstrate why this is an essential topic for optimization and why it is worth exploiting varying times during evaluation. Second, different evaluation procedures in an algorithm regarding the job granularity and their scheduling are discussed. Then, a more specific but frequently occurring case of computationally expensive objectives with inexpensive constraints is investigated. Lastly, any kind of heterogeneity of objectives and constraints is considered, which requires handling partial information during evaluation and optimization. The majority of this chapter is based on the article published in [26], except Section 5.4 which is based on [25].

5.1 Introduction

Many real-world optimization problems require the consideration of multiple conflicting objectives to reflect the complexity of the application [9]. Additionally, the satisfaction of constraints is necessary to guarantee to find an indeed feasible solution [263]. Let us quickly review the definition of an optimization problem

$$\begin{aligned} &\text{Minimize} && f_m(\mathbf{x}), && \forall m \in (1, \dots, M), \\ &\text{subject to} && g_j(\mathbf{x}) \leq 0, && \forall j \in (1, \dots, J), \\ &&& x_d^{(L)} \leq x_d \leq x_d^{(U)}, && \forall d \in (1, \dots, D), \end{aligned} \tag{5.1}$$

where \mathbf{x} are the variables to optimize, $x_d^{(L)}$ and $x_d^{(U)}$ are the lower and upper bounds for the d -th variable, f_m is the m -th objective, and g_j the j -th constraint function. For brevity, no equality constraints are considered here. The above mathematical description makes it apparent that assessing the performance of one design (solution) \mathbf{x} requires evaluating a set of functions: M

objective and J constraint functions. This results in evaluating a total of $M + J$ functions, from here on, referred to as *target* functions.

Because of the multi-disciplinary nature real-world problems, some of the target functions (a functional group) may require calling a single third-party software, running a simulation [21, 22], or other computing-intensive tasks [67, 153]. Depending on the software being used, the performance assessment might become fairly time-consuming, for instance, a couple of hours or even days [28]. A typical practical optimization problem may have two to five such functional groups which must be independently called and evaluated. In addition, there may be certain simplistic target functions, which are mathematically defined and are much quicker to compute than the software or simulation codes.

When optimizing computationally expensive functions, special attention needs to be paid to the limited solution evaluation budget. Over the last decades, researchers have predominantly used surrogate-assisted optimization methods, where an interpolation or approximation model of the computationally expensive target function is utilized during optimization [264]. Most existing surrogate-assisted algorithms assume that the values of *all* target functions (objectives and constraints) are evaluated within one computing job and become available only at the end of the most expensive target evaluation.

A point-based optimization method requires a single new solution to be evaluated at a time to complete an iteration. Thus, the possibilities of exploiting the heterogeneity in the evaluation time of different target functions are limited. However, for a population-based optimization algorithm, such as a generational evolutionary computation (EC) algorithm, a set of offspring population members must be evaluated to proceed to the next generation. It is intuitive to realize that if some target functions are relatively quick to compute in contrast to others, the partial evaluation of population members can be utilized to determine if a population member needs to go through with more expensive target function evaluations. Thus, there is a need for building an evaluation plan for handling heterogeneous target functions for saving computational time, specifically in a population-based optimization algorithm. This is the main crux of this paper.

The majority of the surrogate-assisted optimization methods create new infill solutions based on optimization of the surrogate models. This is useful in its own right in hopefully finding good solutions in a quick computational time. However, since infill solutions are to be evaluated fully for all target functions before they can be used to update surrogate models or proceed with the algorithm, the ideas must be modified for heterogeneous target functions to harness the full advantage of quick partial evaluation of them. Because neither independently computable nor heterogeneously expensive functions have been a major focus of research in the past, barring a few studies, [25, 140, 141], researchers and practitioners remain using existing optimization methods by letting the optimizer wait until the calculation of all targets has finished. In such a case, the most time-consuming target function determines the waiting time for a solution to be evaluated entirely [140]. The waiting is caused by the optimization method not being capable of processing partial information and results in unused time slowing down the convergence. Other challenges, such as managing computational resources, software licenses, or dealing with hardware or software failures, must be addressed when optimizing real-world problems.

5.2 Related Work

Some effort has been made in the past to investigate the heterogeneous expensiveness of target functions. Mostly, bi-objective problems with no constraints have been considered so far. This implies one objective being computationally inexpensive (cheap) and one being expensive. Since only two target functions are considered, authors also refer to the difference as a *delay* in evaluating the objective functions.

The first paper directly addressing heterogeneously expensive objectives was published by Allmendinger et al. [140] in 2013. The authors have proposed three different ways of dealing with missing an objective value caused by such a delay. First, the missing objective value can be filled by randomly drawing pseudo values in the boundaries of the objective space (random). Second, some Gaussian noise is added to the corresponding objective value of a randomly chosen individual (being evaluated on all objectives) and assigned (noise-based). Third, the missing value is replaced

by the nearest neighbor’s objective values – which can be interpreted as fitness approximation – in the design space being evaluated on all objectives (fitness inheritance). Moreover, for the evaluation selection, the authors have proposed to select always the most recently generated offsprings (sweep selection) or to select them based on a priority score obtained by full or partial non-dominated rank (priority selection). The results indicate that the fitness-inheritance-based pseudo value assignment combined with the sweep selection performs the best.

This initial study has been extended by a number of schemes for handling the delay of an objective function [141]. The authors proposed four different approaches: wait for all objectives to be finished (Waiting); optimize the cheap objective first and evaluate the expensive objective for the optima found (Fast-First); use the cheap objective to look ahead at possible promising offsprings each generation (Brood Interleaving); incorporate even more selection pressure for the expensive objective evaluations by running a single-objective optimization algorithm on the cheap objective (Speculative Interleaving); the experimental study revealed that the performance is affected by the amount of delay for the objective. Speculative Interleaving turned out to perform well when the termination criterion is based on a shorter time limit, and the delay of the objectives is rather significant. Unsurprisingly, the Fast-First strategy outperformed other methods when the objectives were highly positively correlated. The authors also found out that Waiting and Brood Interleaving became increasingly competitive with a longer running time of the algorithms.

In 2018, Chugh et al. [142] proposed HK-RVEA an extension of K-RVEA [124] which can handle two objective functions with different latencies. Moreover, in contrast to other existing methods, where the expensive objective is predicted by relatively simple approximation, the authors have used Kriging (also known as Gaussian process), a powerful approximation model frequently used in surrogate-assisted optimization. Significant changes compared to the original K-RVEA are related to the training and update mechanism of the surrogate, driven by a single-objective evolutionary algorithm. A comparison regarding bi-objective test problems with previously proposed approaches [140, 141] showed that HK-KRVEA works especially well in cases with low latencies.

Thomann et al. have developed the trust region-based algorithm MHT that employs quadratic

approximations for objectives not being evaluated yet [143]. The Tammer-Weidner-functional is used for finding descending directions to make use of the heterogeneity of the objective functions. The trust region limits the surrogate's underlying error and serves as a step size in each iteration. The authors have used the concept of local ideal points given by the minimum of the local quadratic model to calculate the search direction in each step. Because of the limited function evaluation budget and one computationally expensive function, the goal of this initial study was to obtain only a single Pareto-efficient solution. In [265] the same authors have proposed a method that starts from the Pareto-efficient solution found by MHT and attempts to explore the neighborhood of the solution further to cover the whole or at least parts of the Pareto-front.

A surrogate-assisted approach called Tr-SAEA has been proposed by Wang et al. for heterogeneously expensive bi-objective problems [144, 266]. Inevitably, the existence of a computationally inexpensive and an expensive one quickly leads to knowledge asymmetry. Thus, the authors propose a transfer learning scheme within a surrogate-assisted evolutionary algorithm to transfer knowledge from the fast objective to the slower one. The transfer is achieved by fitting models where knowledge about the variables and the fast objective serve as an input. The approach has shown to be more robust to varying levels of latency and correlation between the objectives.

Another informative resource about the state-of-the-art of heterogeneous objectives and future research can be found in [267]. The article focuses on unconstrained multi-objective optimization with heterogeneous objective functions. Heterogeneity is discussed in a general manner with a focus on the heterogeneity of the evaluation times. The authors give an overview of recent developments and possible gaps in this research direction.

In [268], independently computable functions in constrained single-objective optimization have been investigated. The authors have proposed eight different constraint handling techniques by combining the ranking of infeasible/feasible solutions, the evaluation type, and the constraint violation aggregation function. In this first study, the sequence in which the constraints are evaluated is determined randomly. Results have shown that this can already significantly improve the convergence of an optimization algorithm. Later on, the work has been extended by incorporating a

feasibility relaxation mechanism to permit constraint evaluation for potentially important solutions close to the constraint boundary and by using the feasibility ratio to determine the sequence for constraint evaluation [269]. By ordering the constraints based on the likelihood of violation, computational resources can be saved by stopping to evaluate a solution further as soon as the first violating constraint is discovered. Instead of using only knowledge of the feasibility ratio of constraints gathered from the past, a surrogate for predicting a solution's likelihood to violate a specific constraint is used in [270]. Other novelties presented by the authors are a modified infeasibility-driven ranking for ordering the partially evaluated solutions and an adaptive switching between partial and complete evaluation. Whereas the ranking is essential to give the potentially infeasible solution a chance to survive, the switching guarantees a minimum amount of entirely evaluated solutions.

Besides publications directly addressing heterogeneously expensive objectives, the connection to related research directions shall be discussed. One way of addressing the expensive objective is approximating the value with a surrogate before doing the time-consuming evaluation. This introduces a low-fidelity evaluation (using the approximation model) with an underlying prediction error and a high-fidelity model (time-consuming but without any error) for the corresponding objective. Having an objective function with different fidelity levels is also known as multi-fidelity optimization [271]. However, in contrast to heterogeneously expensive problems addressed in this paper, in multi-fidelity optimization, not only one but multiple functions with often different expenses exist for the *same* target. Moreover, the existence of multiple independently computable target functions requires to think about the design of *distributed* and *asynchronous* algorithms [143]. Distributing the evaluation of a set of solutions and their objectives and constraints on different computing nodes causes asynchronicity. An implementation has to address the asynchronicity, for instance, by waiting for all information necessary to obtain and return the results – the most common implementation in algorithms – or by processing asynchronous events and thus partial evaluations. Furthermore, the situation of having partially evaluated solutions is in a way related to not considering some objectives or targets temporarily. Some studies have addressed the more

specific case of removing (redundant) objectives for the whole optimization run [272]. In the case of heterogeneous expensiveness, one usually knows beforehand what objective is expensive and, thus, might less frequently be available for all individuals early on. This changes the viewpoint from what objective is being removed to how to decide whether it is worthwhile to spend time evaluating the time-consuming evaluation of the expensive objective for some individuals or not. Moreover, accessing only partial information of objectives or constraints is related to analyzing a data set with missing values. The occurrence of missing values has been studied thoroughly in data science and machine learning and thus is worth having a look at [273].

To the best of our knowledge, the combination of heterogeneously expensive functions for constrained multi-objective optimization has not been explored yet. Thus, this work shall provide a starting proof-of-principle study for different evaluation times considering both multiple objectives and constraints and should encourage more attention in the near future.

5.3 Background

Before different approaches for optimizing problems with heterogeneous target functions are discussed, the evaluation process must be looked at systematically. In general, the evaluation process itself can be split into two interdependent parts: i) the *jobs* being submitted by the algorithm, and ii) the *scheduling* of these jobs. We propose a scheme of different ways for an algorithm to submit jobs regarding a set of solutions and target functions for the former. This defines the *frequency* and *granularity* of information the algorithms retrieves. However, the schedules determine the point of time the algorithm is notified of the job to be finished. The purpose of the scheduler is to decide what job should be executed next. Since resources are commonly limited, a scheduler often uses a job queue or even more sophisticated load-balancing techniques. Even though this may sound like a minor implementation detail, for practitioners running optimization methods in a distributed computing environment, this can become crucially important.

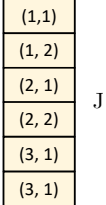
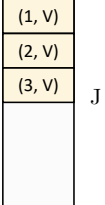
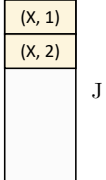
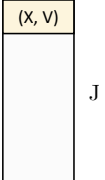
Set of Solutions (X)		Target Values (V) Objectives and/or Constraints	
		Elementwise (\cdot/E)	Batch (\cdot/B)
Set of Solutions (X)	Elementwise (E/ \cdot)	E/E Strategy <pre>function eval(X, V) for i in 1 ... X for j in 1 ... V enqueue(X[i], V[j]) end end</pre> $ J = X \cdot V $ 	E/B Strategy <pre>function eval(X, V) for i in 1 ... X enqueue(X[i], V) end</pre> $ J = X $ 
	Batch (B/ \cdot)	B/E Strategy <pre>function eval(X, V) for j in 1 ... V enqueue(X, V[j]) end</pre> $ J = V $ 	B/B Strategy <pre>function eval(X, V) enqueue(X, V) end</pre> $ J = 1$ 

Figure 5.1: Strategies for the evaluation procedure considering a set of solutions X and target values V to be calculated.

5.3.1 Evaluations Jobs

The frequency and granularity of information the algorithms retrieve opens up new possible ways of asynchronous calculations to efficiently use computing resources. The evaluation of a set of solutions X with multiple target values V can be achieved in several ways. For instance, the algorithm can evaluate each solution in X sequentially by submitting a job for each entry of X separately or a single job containing all solutions in X as a batch. Second, the algorithm can decide what target values each of the jobs should include: Should it be all targets in V that provide complete information about a solution, or just a subset of targets? These choices result in four different ways of packaging the computation jobs defining how the evaluation takes place (see Figure 5.1). We refer to strategy Y/Z where Y and Z are replaced by E (elementwise) if only a single value and by B (batch) if multiple solutions are chosen. The naming convention is applied analogously to Z with respect to the target values. The B/B strategy is most commonly used, where the algorithm

schedules the calculation of all solutions X and target values V only once and retrieves the resulting values when the job has finished. This reduces the number of scheduled jobs to one; however, it does not allow the algorithm to retrieve any intermediate information during evaluation. Contrary to scheduling all jobs at once, the calculation can be split into many small jobs using the E/E strategy. For each solution X_i and each target value V_j , a separated job is submitted. The resulting number of jobs $|J|$ is equal to $|X| \cdot |V|$, and the algorithm retrieves a notification whenever each of the jobs has finished. Thus, it has the highest frequency and granularity of information considered in this schema. However, possible calculations that might be shared across the calculation of target values are done repeatedly. The E/B strategy schedules a single solution at a time but multiple target values, which results in a job list of size $|X|$. Since the set of solutions evaluated at a time is usually larger than the target values, this is the strategy with the second most frequency of information. The B/E strategy submits multiple jobs for each target value but does not split up the set of solutions. This results in $|V|$ jobs to be processed. Analogously, the E/E strategy variables necessary for the calculations of multiple target values cannot be shared. Nevertheless, if some target values are obtained significantly faster than others, the algorithm is notified without waiting for more computationally expensive target values. It is worth mentioning that a target $v \in V$ can also be a group of targets always being evaluated together. Each partitioning has benefits and drawbacks regarding their flow of information to the algorithm and the concrete implementation.

5.3.2 Job Scheduling

Job partitioning defines the general frequency and granularity of information, but the time of retrieving pieces of information remains unknown without concrete timing. The most straightforward implementation of a scheduler is a FIFO or priority queue, allowing new jobs to be added and retrieving the next job to compute. For now, let us assume all jobs in the queue are processed in *parallel* which requires distributing jobs to at least $|X| \cdot |V|$ workers.

Further, the optimization problem consists of two objectives, f_1 and f_2 , and one constraint, g_1 . We assume different execution times for each target value: $t(f_1)$ for the first, $t(f_2)$ for the second

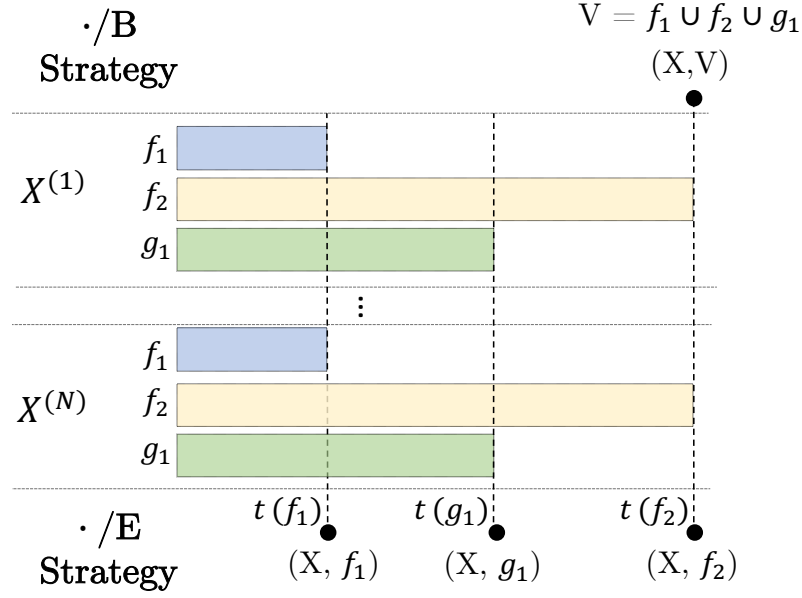


Figure 5.2: A comparison of \cdot/B and \cdot/E strategies assuming parallel processing of all jobs, requiring different average times for evaluation.

objective, and $t(g_1)$ for the constraint. The evaluation times are defined as $t(f_1) < t(g_1) < t(f_2)$. Figure 5.2 demonstrates a possible evaluation procedure for a solution set of size three. Moreover, it shows the information flow for the \cdot/B (E/B and B/B) and \cdot/E (B/E and E/E) strategy. The \cdot/B strategy returns the result given by the union of all target values $V = f_1 \cup f_2 \cup g_1$ exactly once. This implies the algorithm is waiting to obtain full information about all solutions and is idle meanwhile. The waiting time is given by $\max(t(f_1), t(f_2), t(g_1))$ or in general by $\max(t(V_1), \dots, t(V_{|V|}))$. One single outlier (with a rather larger evaluation time) will increase the overall waiting time and greatly impact the algorithm's overall performance. In this example, the maximum evaluation time is given by $t(f_2)$, which is almost three times larger than $t(f_1)$ (see Figure 5.2). On the contrary, the \cdot/E strategy provides some information to the algorithm whenever the calculation of a target value has been finished. Thus, for the three target values the algorithm sends the first notification at $t(f_1)$ with f_1 , the second at $t(g_1)$ with g_1 , and the third at $t(f_2)$ with f_2 . This implies that the optimization algorithm retrieves multiple partial function evaluations at different times, changing the evaluation schedule by step-wise eliminations, thereby making the optimization task more efficient. A batch-wise evaluation of targets would not allow any such advantage.

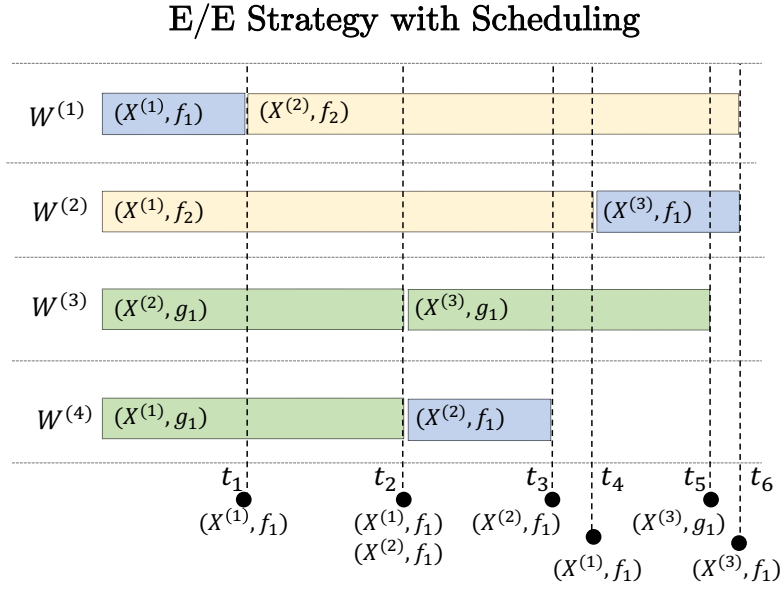


Figure 5.3: Job schedule using a queue.

A different execution of jobs might occur, not assuming that a computing unit exists for all jobs simultaneously. Figure 5.3 illustrates a possible execution of jobs with four instances/worker W_1 to W_4 using the E/E strategy. Because the scheduler decides what jobs to execute next, no prediction about the availability of target values can be made. Moreover, the dispatcher will report different amounts of partial information; thus, an asynchronous algorithm design is desired.

The systematic analysis of the evaluation process will help researchers think of possible implementations and their impact on the algorithm's design. Based on the above evaluation strategies for dealing with heterogeneously expensive objectives and constraints, we propose a population-based optimization method with B/E evaluation strategy, which makes use of partial evaluation of target functions to carefully eliminate evaluation of expensive targets for potentially inferior population members, thereby making the overall method computationally efficient.

5.4 Computationally Expensive Objectives and Inexpensive Constraints

In the past years, a significant amount of research has been done in optimizing computationally expensive and time-consuming objective functions using various surrogate modeling approaches.

Constraints have often been neglected or assumed to be a by-product of the expensive objective computation, thereby being available after executing expensive evaluation routines. However, many optimization problems in practice have separately evaluable computationally inexpensive geometrical or physical constraint functions, while the objectives may still be time-consuming. This scenario probably makes the simplest case of handling heterogeneous and multi-scale surrogate modeling in the presence of constraints. Even though computationally expensive objective functions have been studied extensively [104, 23, 38], computationally inexpensive constraints have been paid little attention to in literature. Thus, we propose IC-SA-NSGA-II, an inexpensive constraint handling method using a surrogate-assisted version of NSGA-II [10]. First, we describe three different methods to generate a feasible design of experiments, and second, the outline of the algorithm. Our proposed method makes explicit use of the computationally inexpensive constraint functions and guarantees a solution’s feasibility before running the time-consuming simulation of objective functions.

5.4.1 Design of Experiments

In surrogate-assisted optimization [104], the so-called Design of Experiments (DOE) needs to be generated to build the initial model(s). The number of initial points N^{DOE} depends on different factors, such as the number of variables or the complexity of the fitness landscape. A standard method frequently used to generate a well-spaced set of points is Latin Hypercube Sampling (LHS) [234]. However, with the availability of an efficient feasibility check, more sophisticated approaches would be preferred. Therefore, three different methods returning a well-spaced set of *feasible* solutions using inexpensive constraint functions are described.

Rejection Based Sampling (RBS): A relatively simple approach is modifying LHS, which already provides a well-spaced set of solutions, to consider feasibility. Such a feasible and well-spaced set of points can be obtained by using LHS to produce a point set P and rejecting all infeasible solutions to obtain a set of *feasible* solutions $P^{(\text{feas})}$. Because infeasible solutions have been discarded, the resulting point set does not have the desired number of points ($|P^{(\text{feas})}| < N^{\text{DOE}}$) and, thus, the

process shall be repeated until enough feasible solutions have been found. If the size of the obtained point set exceeds N^{DOE} , a subset is selected randomly.

Niching Genetic Algorithm (NGA): The sampling process itself can be seen as an optimization problem where the objective is given by the constraint violation $f(\mathbf{x}) = \text{cv}(\mathbf{x})$, which takes a value zero for a feasible solution \mathbf{x} , and a positive value proportional to the sum of normalized constraint violation of all constraints if \mathbf{x} is infeasible. Such an objective function results in a multi-modal optimization problem where a *diverse* solution set with objective values of zeros shall be found. One type of algorithm used for multi-modal problems is niching-based genetic algorithms (NGA), where the diversity is ensured by an ϵ -clearing based environmental survival [274]. For single-objective optimization problems, the ϵ -clearing occurs in the design space and guarantees a distance (usually Euclidean distance is used) from one solution to another. The survival always selects the best performing not already selected or cleared solution and then clears its neighborhood with less than ϵ distance. Thus, the spread of solutions is accomplished by disfavoring solutions in each other's vicinity. After setting the objective to be the constraint violation, a suitable ϵ has to be found. The suitability of a given ϵ depends on the size of the feasible region(s) of the corresponding optimization problem and is unknown beforehand. On the one hand, if ϵ is too large, the number of optimal solutions found by the algorithm will not exceed N^{DOE} . On the other hand, if ϵ is too small, the solution set's spread has room for improvement. For tuning the hyper-parameter ϵ , we start with $\epsilon = \epsilon_0$ (a number close to 1.0) and execute NGA. If the size of the obtained solution set is less than N^{DOE} , we set $\epsilon = 0.9 \cdot \epsilon$ and repeat this procedure until a solution set of at least of size N^{DOE} is found.

Riesz s-Energy Optimization (Energy): The Riesz s-Energy [275] is a generalization of potential energy concept and is defined for the point set \mathbf{z} as

$$U(\mathbf{z}) = \frac{1}{2} \sum_{i=1}^{|\mathbf{z}|} \sum_{\substack{j=1 \\ j \neq i}}^{|\mathbf{z}|} \frac{1}{\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\|^s}, \quad \mathbf{z} \in \mathbb{R}^{n \times M}, \quad (5.2)$$

where the inverse norm to the power s of each pair of points ($\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$) is summed up. It has already been shown in [276] that Riesz s-Energy can be used to achieve a well-spaced point set by

Algorithm 5.1: IC-SA-NSGA-II: Inexpensive Constrained Surrogate-Assisted NSGA-II

Input : Number of Variables n , Expensive Objective Function $f(\mathbf{x})$, Inexpensive Constraint Function $g(\mathbf{x})$, Maximum Number of Solution Evaluations ESE^{\max} , Number of Design of Experiments N^{DOE} , Exploration Points $N^{(\text{explr})}$, Exploitation Points $N^{(\text{exploit})}$, Number of generations for exploitation k , Multiplier of offsprings for exploration s .

```
/* initialize feasible solutions using the inexpensive function g      */
1  $\leftarrow$  constrained_sampling('energy',  $N^{\text{DOE}}$ ,  $g$ )
2  $\mathbf{F} \leftarrow f()$ 
3 while  $|| < ESE^{\max}$  do
    /* exploitation using the surrogate                                */
4      $\hat{f} \leftarrow \text{fit\_surrogate}(\mathbf{F})$ 
5      $(^{(\text{cand})}, \mathbf{F}^{(\text{cand})}) \leftarrow \text{optimize}(\text{'nsga2'}, \hat{f}, g, \mathbf{F}, k)$ 
6      $(^{(\text{cand})}, \mathbf{F}^{(\text{cand})}) \leftarrow \text{eliminate\_duplicates}(^{(\text{cand})}, \mathbf{F}^{(\text{cand})})$ 
7      $C \leftarrow \text{cluster}(\text{'k\_means'}, N^{(\text{exploit})}, \mathbf{F}^{(\text{cand})})$ 
8      $(\text{surrogate}) \leftarrow \text{ranking\_selection}(^{(\text{cand})}, C, \text{crowding}(\mathbf{F}^{(\text{cand})}))$ 
    /* exploration using mating and least crowded selection          */
9      $(', \mathbf{F}') \leftarrow \text{survival}(\mathbf{F})$ 
10     $(^{\text{(mat})}) \leftarrow \text{mating}(', \mathbf{F}', s \cdot N^{(\text{explr})})$ 
11     $(^{\text{(explr})}) \leftarrow \text{feas\_and\_max\_distance\_selection}(^{\text{(mat})}, ^{(\text{cand})}, X, g)$ 
    /* evaluate and merge to the archive                              */
12     $\mathbf{F}^{(\text{explr})} \leftarrow f(^{(\text{explr})}); \mathbf{F}^{(\text{surrogate})} \leftarrow f(^{\text{(surrogate})});$ 
13     $\leftarrow \cup(^{\text{(explr})}) \cup (^{\text{(surrogate)})}$ 
14     $\mathbf{F} \leftarrow \mathbf{F} \cup \mathbf{F}^{(\text{explr})} \cup \mathbf{F}^{(\text{surrogate})}$ 
15 end
```

executing a gradient-based algorithm. The restriction made there that all points have to lie on the unit simplex has been replaced by the feasibility check provided by the computationally inexpensive constraint. Thus, a point is only replaced by its successor obtained by the gradient update if the successor is *feasible*. The algorithm's initial point set, which is necessary to be provided, is first tried to be obtained by RBS, and if NGA could not find a sufficient number of feasible solutions.

5.4.2 Methodology

The outline of IC-SA-NSGA-II is shown in Algorithm 5.1. The initial design of experiments of size N^{DOE} are obtained by the proposed initialization method based on Riesz s-Energy and then evaluated by executing the expensive simulation $f(\mathbf{x})$ (Lines 1 and 2). Afterward, while the

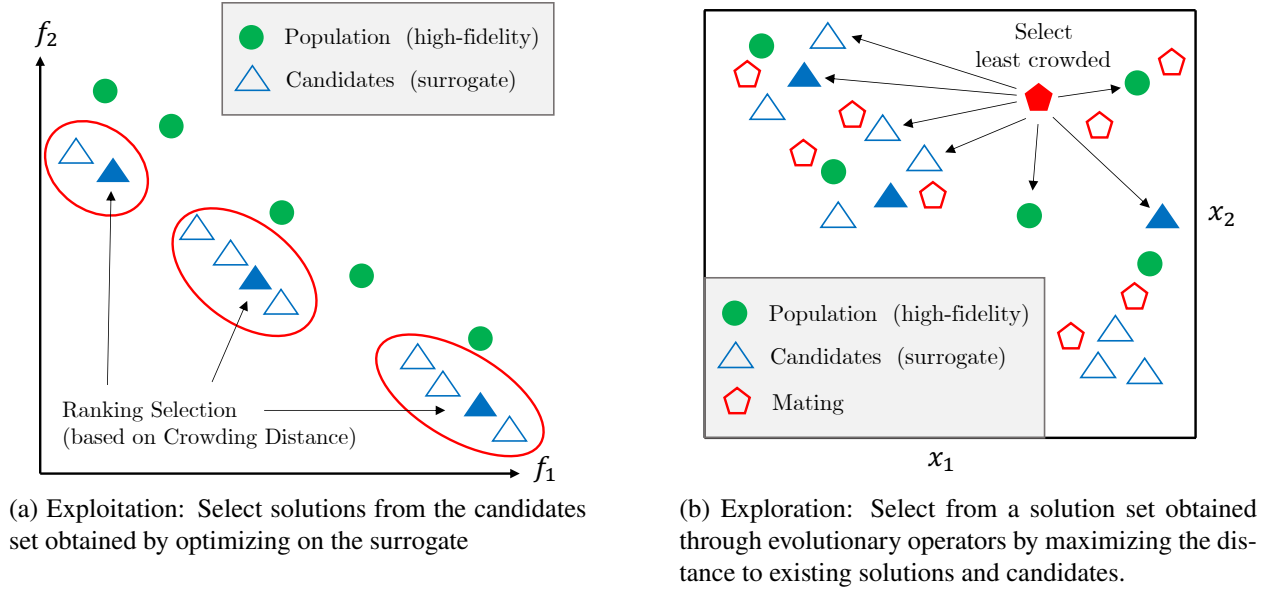


Figure 5.4: The two steps in each iteration: exploitation and exploration.

number of solution evaluations ESE^{\max} is not exceeded (Line 3) the algorithm continues to generate $N^{(\text{exploit})}$ solutions derived from the surrogate for exploitation and $N^{(\text{explr})}$ solutions obtained by mating and a distance-based selection for exploration. The exploitation starts with fitting surrogate model(s) which results in the approximation function \hat{f} (Line 4). Using the surrogates \hat{f} and the computationally inexpensive function g the optimization is continued or in other words simulated assuming $f = \hat{f}$ for k more generations (Line 5). From the last simulated generation, the candidates $^{(\text{cand})}$ and $\mathbf{F}^{(\text{cand})}$ are extracted from the optimum, and duplicates with respect to \mathbf{F} are eliminated (Line 6). This ensures $\mathbf{F}^{(\text{cand})}$ to consist of only non-dominated solutions with respect to \mathbf{F} . From $^{(\text{cand})}$ only $N^{(\text{exploit})}$ solutions are chosen for expensive evaluation by executing ranking selection [1] in each cluster where the ranking is based on the crowding distance in $\mathbf{F}^{(\text{cand})}$. Figure 5.4a illustrates the exploitation procedure of a population with five solutions (circles). The algorithm found 10 candidate solutions (triangles) by optimizing the surrogate and the inexpensive constraint functions. In this example, $N^{(\text{exploit})} = 3$ and, thus, the K-means algorithm is instantiated to find three clusters. From each cluster, the solutions obtained by ranking selection based on the crowding distance are assigned to $^{(\text{surrogate})}$.

Besides the exploitation, some exploration is essential to be incorporated into a surrogate-

assisted algorithm. The exploration is based on the evolutionary recombination of NSGA-II with post-filtering based on the distance in the design space (Line 9 to 11). First, the environmental survival is executed because the mating should not be based on the archive but instead on a subset of more promising solutions $'$. Second, mating takes place to produce $s \cdot N^{(\text{explr})}$ solutions $^{(\text{mat})}$ and, third, the set of *feasible* solutions from $^{(\text{mat})}$ being maximally away from $^{(\text{cand})}$ are assigned to $^{(\text{explr})}$. Figure 5.4b demonstrates this explorative step in more detail. All infeasible solutions generated through mating have already been eliminated. The solution with the maximum distance to others is selected, which represents the least crowded solution with respect to $^{(\text{cand})}$. The exploration step purposefully chooses solutions not suggested by the surrogate and helps to escape from local optima if necessary. After selecting the first solution with the maximum distance to others, the solution is marked as selected and considered in the distance calculations for the second iteration in the selection procedure. Finally, the infill solutions $^{(\text{surrogate})}$ and $^{(\text{explr})}$ are evaluated on the expensive objective functions f and merged with $^{(\text{cand})}$ and \mathbf{F} (Line 12 to 14).

In our implementation, we set the number of the initial design of experiments to $N^{\text{DOE}} = 11n - 1$, where n is the number of variables. Moreover, we simulate NSGA-II for $k = 20$ generations with 100 offsprings each generation on the surrogate model. We have used the NSGA-II implementation available in pymoo [29] for optimization and the Radial Basis Function (RBF) implementation with a cubic kernel and linear tail available in pySOT [277] for the surrogate model. In each iteration five new solutions are evaluated using the expensive objective function, where $N^{(\text{exploit})} = 3$ and $N^{(\text{explr})} = 2$. Furthermore, we set the multiplier of offsprings during exploration to $s = 100$. Moreover, it is worth pointing out that we compare our method with SA-NSGA-II, which does not assume the constraints are inexpensive but follows overall the same procedure. In contrast to IC-SA-NSGAII, it uses regular Latin Hypercube Sampling for the initial design of experiments and fits a surrogate of the constrained function(s) to evaluate feasibility.

5.4.3 Results

The proposed method uses the inexpensiveness of the constraint function(s), and, thus, the performance on *constrained* multi-objective optimization problems will be evaluated. In this chapter, we focus on bi-objective problems with up to 10 constraint functions. In contrast to the constraint function, we treat all objectives to be computationally expensive.

To evaluate the algorithm's performance, we use the CTP test problems suite [278], which has been designed to address constraints of varying difficulty. Moreover, the performance on other bi-objective constrained optimization problems frequently used in literature such as OSY [9], TNK [9], SRN [9], C2DTLZ2 [258], C3DTLZ4 [258], and Car Side Impact (CAR) [258] shall be evaluated. The number of solution evaluations ESE^{\max} is kept relatively small to mimic the evaluation budget of time-consuming simulations. In the following, first, the performance of methods proposed to generate a feasible solution set for the design of experiments are *visually* analyzed, and, second, the algorithm's performance on test problems is discussed.

In Figure 5.5, the results of Rejection Based Sampling (RBS), Niching GA (NGA), and Riesz-s-Energy (Energy) are shown. Compared to RBS and NGA, Energy obtains a very uniform and well-spaced point set in the inside of the feasible region across all problems. Also, it is worth noting that points on the constraint boundary are found, which can be very valuable to start with because, in practice, optima frequently lie on constraint boundaries. For the purpose of visualization, the CTP8 problem with two variables (and nine feasible disconnected regions) has been investigated. All methods were able to obtain more than one feasible solution in all regions.

Table 5.1 lists the median values (obtained from 11 runs) of the Inverted Generational Distance (IGD) [240] indicator of 14 constrained bi-objective optimization problems. The obtained results have been normalized with respect to the ideal and nadir point of each problem's True front. The best performing method and other statistically similar methods (Wilcoxon rank test, $p = 0.05$) are marked in bold. Besides IC-SA-NSGA-II, we ran a more steady-state version of NSGA-II with five offsprings in each generation and an initial population of size $11n - 1$ sampled by LHS. Moreover, SA-NSGA-II is the proposed method without the modifications made to exploit the availability

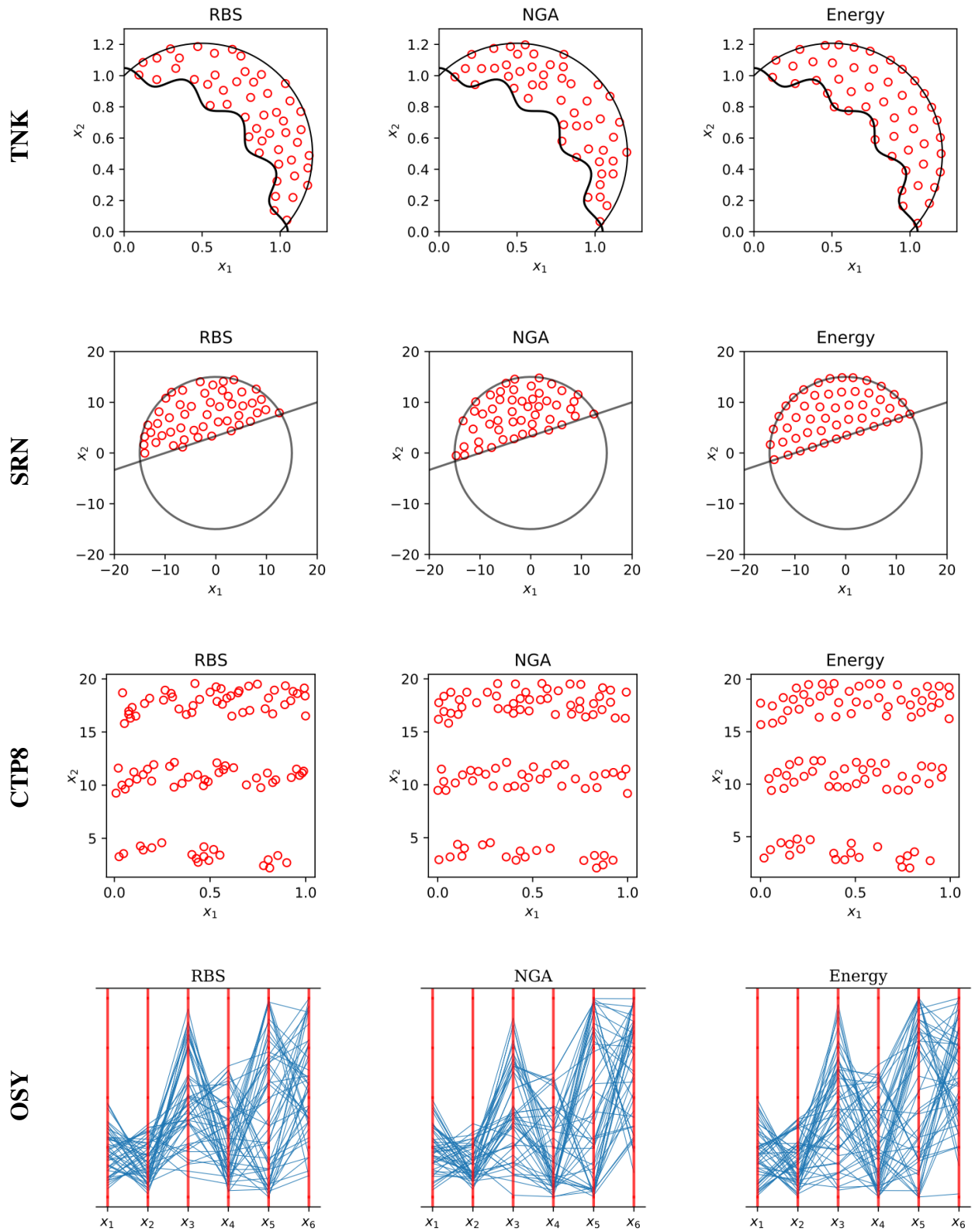


Figure 5.5: Sampling the design of experiments only in the feasible space using Rejection-Based Sampling (RBS), Niching Genetic Algorithm (NGA) and Energy Method.

Table 5.1: The median normalized Inverted Generational Distance (IGD) values out of 11 runs for NSGA-II, SA-NSGA-II and IC-SA-NSGA-II on constrained bi-objective optimization problems. The best performing method and other statistically similar methods are marked in bold.

Problem	Variables	Constraints	ESE ^{max}	NSGA-II	SA-NSGA-II	IC-SA-NSGA-II
CTP1	10	2	200	3.6399	0.0237	0.0196
CTP2	10	1	200	1.4422	0.1721	0.0173
CTP3	10	1	200	1.2282	0.2752	0.0357
CTP4	10	1	400	0.8489	0.3969	0.0736
CTP5	10	1	400	0.7662	0.1145	0.0139
CTP6	10	1	400	7.7155	0.1909	0.0117
CTP7	10	1	400	1.5517	0.0164	0.0032
CTP8	10	2	400	11.6452	0.5963	0.0074
OSY	6	6	500	0.4539	0.0273	0.0381
SRN	2	2	200	0.0263	0.0112	0.0108
TNK	2	2	200	0.1281	0.0200	0.0092
C2DTLZ2	12	1	200	0.3787	0.1185	0.0484
C3DTLZ4	7	2	200	0.2622	0.1210	0.0481
CAR	7	10	200	0.2362	0.0168	0.0147

of inexpensive constraints. Clearly, IC-SA-NSGA-II outperforms the other approaches for most of the problems. For 11 out of 14 optimization problems, our proposed method shows the best performance significantly; for two problems (CTP1 and SRN), SA-NSGA-II performs statistically similarly; and for one problem (OSY), SA-NSGA-II shows slightly better results. NSGA-II is not able to find a near-optimal set of solutions with the limited ESE^{max} for any of the selected problems.

Figure 5.6 shows the obtained solution set for each method of representative runs for CTP2, CTP4, CTP8, C3DTLZ4, TNK, and OSY, for which well-converged and well-diversified non-dominated solutions are found with 200 to 500 solution evaluations. For the difficult problem CTP2, our proposed method converges near the true optima, which lies on the constraint boundary. Similarly, for CTP8, where nine feasible islands for which three contain the Pareto-optimal set exist and, thus, a good exploration of the search space is needed. For C3DTLZ4, IC-SA-NSGA-II has obtained a better diversity in the solution set than SA-NSGA-II.

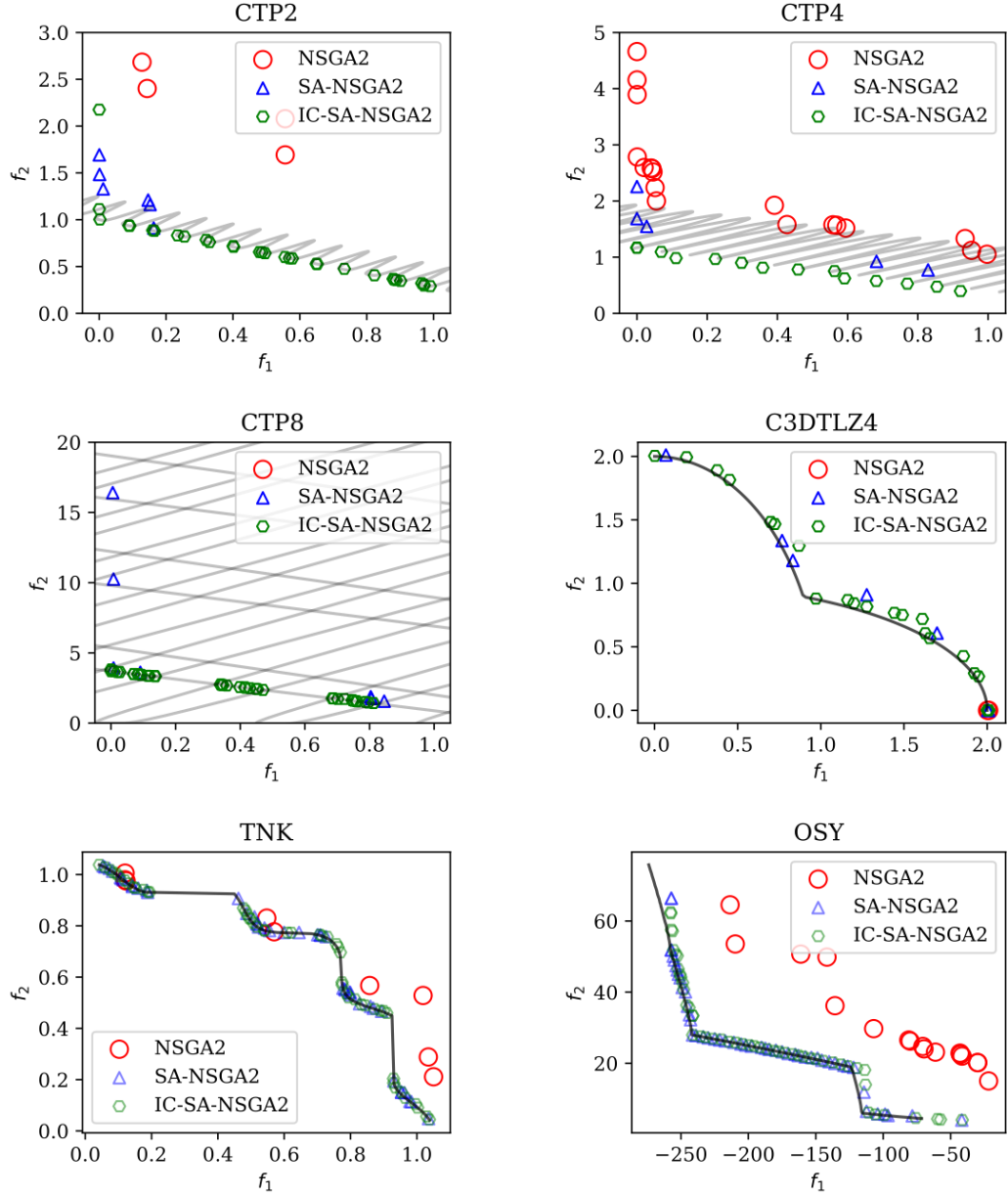


Figure 5.6: Solutions in the objective space of representative runs for CTP2, CTP4, CTP8, C3DTLZ4, TNK, and OSY.

5.4.4 Summary of Section 5.4

The optimization of computationally expensive optimization problems has become more important in practice. Often such problems have physical or geometrical constraints that are relatively computationally inexpensive and can be formulated in equations without running the simulation. To solve these kinds of problems, we have proposed IC-SA-NSGA-II, a surrogate-assisted NSGA-II, which efficiently handles inexpensive constraint functions in the initial design of experiments as well as in each iteration. We have tested our proposed method on 14 constrained bi-objective optimization problems, and our results indicate that efficiently handling the inexpensive constraints helps to converge faster.

This section has focused on solving constrained optimization problems with inexpensive constraints and expensive objective functions with limited solution evaluations. However, some other heterogeneous problems with different time scales of objective and constraint evaluations must be considered next. For instance, the constraints can be even more time-consuming than the objective function, or the objectives and constraints can have different time scales within themselves. After addressing such cases, there is a need for developing a unified algorithm for generic heterogeneous optimization problems. Moreover, studies about heterogeneously expensive optimization problems have so far been limited to bi-objective optimization problems. This section has started to add some more complexity by adding constraint functions. However, the effect of heterogeneity for many-objective optimization problems shall provide more insights into exploiting the discrepancy of evaluation times and asynchronicity.

5.5 Constrained Multi-Objective Optimization Problems With Heterogeneous Evaluation Times

After investigating a scenario with computationally inexpensive constraints and expensive objectives, this shall now be generalized to independently computable heterogeneously expensive target functions. Most existing algorithms do not particularly exploit the practical fact that the objectives

and constraints are often independently computable. The independent evaluation usually originates from different software packages being executed to determine the performance of a solution. The procedure of assessing the performance of a solution might be fundamentally different for each software package, and thus most of the time, the computing time will vary. This results in an optimization problem with independently heterogeneously expensive objectives and constraints that the optimization method must evaluate and use.

For most practical problems, at least one of the target functions involves a time-consuming evaluation process (we refer here as *high-fidelity* evaluations), thereby causing an optimization run to go on for hours or days. In order to optimize such problems, *surrogate-assisted* optimization methods are prevalent. Using a few high-fidelity solution evaluations, a *surrogate* model (an approximate mathematical function) of each target function is created. Instead of using high-fidelity expensive target functions, surrogate models are usually optimized to find a set of *infill* points. Evaluating a solution using the surrogate models is referred to as *low-fidelity* evaluation. The created infill points are evaluated using high-fidelity target functions, and the surrogate models are updated. Since the optimization task is performed on the surrogate models (low-fidelity evaluation), there is usually a substantial gain in computational time compared to optimization with high-fidelity evaluations. In general, there is a trade-off between gain in computational effort and the resulting accuracy of the obtained solutions in surrogate-assisted optimization methods. If a few high-fidelity solutions are used to build surrogate models, the computational time will be small, but the resulting surrogate model may be inaccurate. Thus, optimizing the surrogate models may result in inferior infill solutions in terms of the high-fidelity target functions. Surrogate-assisted optimization algorithms make a fine balance between the building cost of surrogate models and how extensively they are optimized.

However, it is important to note that this paper does not propose another surrogate-assisted optimization algorithm, nor does it plan to compare the proposed methodology with an existing surrogate-assisted optimization method. Here, we focus on handling heterogeneously expensive target functions within an optimization algorithm utilizing surrogate models. Thus, no effort is

made to directly find new infill points using the surrogates. However, instead, models are used to evaluate new solutions to estimate their expected target function values without computing them with high-fidelity evaluation procedures. This is not to say that built surrogate models cannot be exploited further beyond the scope of our proposed methodology, and rather this is something we plan to execute in a subsequent study. Here, we address the presence of heterogeneity in practical target function evaluations and how a population-based algorithm can exploit it to come up with a computationally quick algorithm.

5.5.1 How to Exploit Heterogeneity of an Optimization Problem?

Most existing population-based algorithms do not assume target functions are independently computable and thus wait to update the old population until all new population members are evaluated for all target functions. If all target functions must be computed simultaneously by a single evaluation procedure, or the evaluation time for all target functions is negligible compared to the desired time for executing an optimization run, no special treatment for any evaluation schedule is necessary.

However, let us consider that not all target functions can be evaluated as a single block of computation, but rather independent groups of target functions must be evaluated using different evaluation schemes. Moreover, some groups of target functions require comparatively more time to get evaluated compared to other groups, so there is heterogeneity in the computing efforts. Such problems are predominant in most practical problems, including science and engineering. A practical design must be evaluated from multi-physics considerations, such as aerodynamics, fluid mechanics, solid mechanics, aesthetics, and others. In such a scenario, if a new design is already evaluated to be worse for certain relatively inexpensive target functions, it can be avoided for further processing within the algorithm and thereby saving computational time by not evaluating expensive target functions. Importantly, such decisions can only be made relative to a set of solutions and cannot be made for a single solution. This is the reason why EC methods are ideal candidates for handling heterogeneous target functions.

For instance, let us assume a genetic algorithm after having executed a few iterations when

solving a bi-objective optimization problem with one constraint. The algorithm has completed the mating process and created a set of new offspring solutions X to be evaluated for two objectives f_1 and f_2 and constraint g_1 . Instead of evaluating all solutions at once for f_1 , f_2 , and g_1 , only one or multiple solutions can be chosen from X to retrieve one target function value at a time, say, the constraint g_1 . Depending on the values of g_1 , some solutions can be discarded already because they are found to be infeasible. The more “promising” solutions are kept and will continue to be evaluated on the next target f_1 . Analogously, some solutions can be eliminated, and only a few are finally sent to obtain the values f_2 . By processing partial information, not all solutions will be evaluated for all targets, which speeds up the overall evaluation process. Such exploitation of partial information requires answering two elementary questions.

(Q1) Target Order Problem: A relevant question arises: “In what order should the targets be evaluated?” Intuitively, this should depend on the actual evaluation times of target functions and their predicted target values. For the example discussed above, if computational times are having the following relationships: $t(f_1) < t(g_1) < t(f_2)$, then an ordering of the targets by evaluation time should follow least to most expensive, or f_1 followed by g_1 , which is then followed by f_2 . But the values of the target functions must also play an essential role in deciding on the order of evaluation. Suppose g_1 is found to be positive. In that case, this indicates that the solution is infeasible, and a smart algorithm may decide not to evaluate f_1 and f_2 at all for this solution to save computational time. That is well and good, but there is a problem with the above method. In order to know their relative target function values, the solution has to be evaluated for all target functions. If all functions are already evaluated, there is no need to do any ordering. This rounding argument can be answered by making low-fidelity evaluations of offspring population members using the current surrogate models. But since surrogate models are approximate, each surrogate model may have different prediction accuracy. Thus, the order of their evaluation should depend on a solution’s rank in the population-based on its predicted target values (for example, in multi-objective NSGA-II, a combination of non-dominated rank and crowding distance), their accuracy of prediction, and computational time for high-fidelity evaluation of target functions. Despite the importance of g_1 in

determining the feasibility of a solution, it may turn out that computation of the cheapest objective function (f_1) first is beneficial over the constraint evaluation to determine if the most expensive objective (f_2) needs to be evaluated at all. A combined metric for ordering target functions is provided in Subsection 5.5.2.3.

(Q2) Elimination Problem: The next question is as follows: “Under what circumstances should an offspring solution be eliminated and not continued to be evaluated for the remaining targets?” If all targets would be evaluated for all solutions, then the optimization method does not make any use of separately computing solutions. Therefore, some solutions need to be eliminated during the evaluation process. Whereas the order defines what partial information should be made available next, the elimination decides whether a solution is worth keeping and evaluated for more targets. The decision is based on partial information, where some targets are evaluated, their high-fidelity values are available, and a surrogate only predicts others. Another valuable piece of information is the surrogate accuracy derived from the past, which helps to judge how reliable the predictions are.

5.5.2 Methodology

5.5.2.1 Survival Under Uncertainty

An environment selection or survival decides given a set of solutions which one are the fittest and shall survive. Under certainty, numerous survivals have been proposed in the literature, for instance, for unconstrained single-objective genetic algorithms simply a selection based on the objective values [1] or in NSGA-II, survival based on non-dominated sorting and crowding distance [10]. However, most environmental survivals proposed in the evolutionary computation literature assume that the exact values for objectives and constraints are known. The goal of the proposed *probabilistic* survival is to make existing survival procedures applicable under uncertainty. With uncertainty, we refer to the situation that some target values originate from a prediction with an underlying error. Despite the situation where either all targets are based on predictions or all are exact, the survival also needs to handle cases of mixed uncertainty, where some targets are exact and some predicted.

Algorithm 5.2: Probabilistic Survival: Subset Selection under Uncertainty

Input : Population P , Offsprings Q , Uncertain Targets V , Predictions errors e , Iterations γ

```
/* Repeat the experiment  $\gamma$  times */
1 foreach  $k \leftarrow 1$  to  $\gamma$  do
2    $\alpha_i \leftarrow 0 \quad \forall i \in (1, \dots, |Q|)$ 
3    $M \leftarrow \text{merge\_and\_copy}(P, Q)$ 
4   foreach  $i \leftarrow 1$  to  $(1, \dots, |M|)$  do
5     /* Add noise for uncertain target */
6     foreach  $v \in V$  do
7        $M[v] = M[v] + \mathcal{N}(0, e_v^2)$ 
8     end
9      $M' \leftarrow \text{survival}(M)$ 
10    foreach  $i \leftarrow 1$  to  $(1, \dots, |Q|)$  do
11      /* If survived, increase the counter */
12      if  $Q_i \in M'$  then  $\alpha_i \leftarrow \alpha_i + 1$  ;
13    end
14  end
15  /* Convert survival counts to probabilities */
16  foreach  $i \leftarrow 1$  to  $(1, \dots, |Q|)$  do  $\alpha_i \leftarrow \alpha_i / \gamma$  ;
17 return  $\alpha$ 
```

The proposed probabilistic survival does not change an existing survival but calls it repeatedly with some introduced error noise for predicted targets. The procedure is illustrated in Algorithm 5.2. Given a parent population P , offsprings Q , a set of uncertain targets V , the average prediction error e_v of each target $v \in V$, the number of iterations the experiment is repeated γ , and a survival probability α_i for each offspring Q_i . In total, the survival under certainty calls the survival considering certainty exactly γ times. In each iteration, first, the population P and offsprings Q are merged and copied to M . Then for each solution, for each uncertain target $v \in V$ Gaussian noise is added $\mathcal{N}(0, e_v^2)$. The population, M with error noise, is sent to the survival selection, and the survivors are assigned to M' . If solution i has survived and thus is in M' , its counter α_i is increased by one. Finally, the survival counters α_i are converted to probabilities by dividing by the number of experiments conducted γ .

With a mix of certain and uncertain targets, the proposed survival might look as follows. Assuming the objective space values f_1 already have been assessed using the high-fidelity evaluation,

Algorithm 5.3: Probabilistic Surrogate-Guided Mating

Input : Population P , Surrogate S , Predictions errors e , Mating Iterations β , Prob. Surv. Iterations γ

```
/* Regular mating used by EA */
1  $Q \leftarrow \text{mating}()$ 
2 foreach  $k \in (1, \dots, \beta)$  do
    /* Double the number of offsprings */
3      $Q' \leftarrow Q \cup \text{mating}()$ 
    /* Surv. Prob. of each offspring */
4      $\alpha \leftarrow \text{prob\_surv}(P, Q', V, e, \gamma)$ 
    /* Discard unpromising offsprings */
5      $Q \leftarrow \text{top}(Q', |Q|, \alpha, \text{'descending'})$ 
6 end
7 return  $Q$ 
```

and \hat{f}_2 and \hat{g}_1 are predicted by a surrogate with a known prediction error of $e_{\hat{f}_2}$ and $e_{\hat{g}_1}$, respectively. In each iteration, \hat{f}_2 is provided with error noise $\mathcal{N}(0, e_{\hat{f}_2}^2)$, as well as the constraint \hat{g}_1 with $\mathcal{N}(0, e_{\hat{g}_1}^2)$. The outcome of multiple survival experiments with different amounts of introduced error noise for uncertain targets (f_2 and g_1) lets us derive the probability of a solution to survive in a mixed certain and uncertain environment.

5.5.2.2 Probabilistic Surrogate-Guided Mating

Given the prediction and the average error for each target, one can calculate the survival probability α for each offspring originating from mating. Since the mating only uses information about the parents and no predictions, many solutions may have a relatively low survival probability and might be directly discarded. In order to increase the survival probability, this information can be directly used during mating.

In Algorithm 5.3 the proposed modified mating is demonstrated. The idea is based on repeating the original mating procedure `mating()` multiple (β) times by only keeping the most promising offspring solutions. Initially, the offspring population Q is created. Then in each iteration, another offspring population is merged to create Q' . For each solution in Q' , the survival probability is determined to keep solutions most likely to survive. This is achieved by taking the top $|Q|$ solutions

from Q' based on a descending sorting by α . After having the process repeated β times, the solutions that have repeatedly survived are returned.

5.5.2.3 Heterogeneously Expensive Evolutionary Algorithm (HE-EA)

The pseudo-code of the proposed heterogeneously expensive evolutionary algorithm (HE-EA) is shown in Algorithm 5.4. HE-EA assumes that an approximation of evaluation times (ET) for each target exists beforehand. However, if this should not be the case, the evaluation time can be kept track of using a book-keeping approach after evaluating each target. Initially, a list of all targets V is created where first all objectives and then all constraints appear. Afterward, HE-EA creates a space-filling set of designs P . Then, for each target $v \in V$, the initial population P is evaluated by calling the high-fidelity evaluation function $\text{evaluate}(P, v)$, the survival error ρ_v is set to one, the surrogate S_v is fit, and the mean absolute error e_v is estimated using cross-validation. Cross-validation is helpful to provide to measure the complexity of each target.

Until the time limit of running the optimization procedure has been met, the algorithm's main loop is repeated. It starts by performing the mating procedure to generate the offspring population Q and predicting the objective and constraints $\text{predict}(S, Q)$ using the surrogate S . Analogously, the current population P is copied to P' , and the surrogate is used to obtain approximations. Next, the order in which the targets are supposed to be evaluated needs to be determined. The order should be based on the trade-off between evaluation time ET and surrogate error e . The function $\text{order}(\text{ET}, \alpha)$ first calculates an indicator value for each target. We propose a metric called information gain (IG_k) of the k -th target as the survival error (ρ_k) per unit evaluation time (ET_k), as follows:

$$\text{IG}_k = \frac{\rho_k}{\text{ET}_k}. \quad (5.3)$$

Targets with larger information gain are preferred to be evaluated first and thus, the target evaluation order τ is given by sorting IG in *descending* order. To illustrate the intuition behind this order, let us consider a few examples where two targets, v_1 and v_2 , are compared with each other. Assuming both targets have the same evaluation time $\text{ET}_1 = \text{ET}_2$, but target one has a larger survival error

Algorithm 5.4: Heterogeneously Expensive Evolutionary Algorithm (HE-EA)

```
Input : Evaluation Times ET, Max. Survival Prob.  $\alpha^{(\min)}$ 
/* Initialize the target vector */
1  $V \leftarrow (f_1, \dots, f_m, g_1, \dots, g_J)$ 
/* Sample design of experiments */
2  $P \leftarrow \text{doe}()$ 
3 foreach  $v \in V$  do
4    $\text{evaluate}(P, v)$ 
5    $\rho_v \leftarrow 1.0$ 
6    $S_v \leftarrow \text{fit\_surrogate}(P, v)$ 
7    $e_v \leftarrow \text{estm\_mae}(S, P, v)$ 
8 end
9 while time left do
/* Create the offspring population */
10  $Q \leftarrow \text{prob\_mating}(P, Q, S, e, \gamma, \beta)$ 
/* Prediction of P and Q */
11  $Q' \leftarrow \text{predict}(S, Q)$   $P' \leftarrow \text{predict}(S, P)$ ;
/* The order to eval. targets */
12  $\tau \leftarrow \text{order}(ET, \rho)$ 
/* Targets with uncertainty */
13  $V^{(U)} \leftarrow V$ 
/* Survival prob. before evaluation */
14  $\alpha^{(0)} \leftarrow \text{prob\_surv}(P', Q', V^{(U)}, e)$ 
15 foreach  $k \leftarrow 1$  to  $|V|$  do
16    $v \leftarrow V[\tau_k]$ 
/* Evaluate and copy targets */
17    $\text{copy}(P, P', v); \text{evaluate}(Q', v)$ 
18    $V^{(U)} \leftarrow V^{(U)} \setminus \{v\}$ 
19    $\alpha^{(k)} \leftarrow \text{prob\_surv}(P', Q', V^{(U)}, e)$ 
/* Calculate the survival error */
20    $\rho_v \leftarrow \sum_{i=1}^{|Q'|} |\alpha_i^{(k)} - \alpha_i^{(k-1)}|$ 
/* Fit target surrogates and calc. e */
21    $S_v \leftarrow \text{fit\_surrogate}(P, v)$ 
22    $e_v \leftarrow \text{mae}(S, P, v)$ 
/* Eliminate unpromising solutions */
23    $Q \leftarrow \text{eliminate}(Q, \alpha^{(k)}, \alpha^{(\min)})$ 
/* If all solutions were eliminated */
24   if  $|Q| = 0$  then break;
25 end
26  $P \leftarrow \text{survival}(P \cup Q)$ 
27 end
```

$\rho_1 > \rho_2$. This results in $\text{IG}_1 > \text{IG}_2$ and the first target to be evaluated first. Intuitively this is the right decision because the target with a more significant estimation error can potentially eliminate

more solutions already and thus save computation time. On the other hand, let two targets have the same survival error $\rho_1 = \rho_2$, but the first one have a larger evaluation time $ET_1 > ET_2$. This results in $IG_1 < IG_2$. In this case, the effort for evaluation is identical, and the target modeled less accurately is evaluated first. The intuition behind information gain is defining a target order giving preference to targets that are difficult to predict by the surrogates or are computationally less expensive during evaluation.

Before starting with the evaluation procedure, the uncertain targets $V^{(U)}$ are initialized to be all targets V , and the initial survival probabilities $\alpha^{(0)}$ are obtained by executing the probabilistic survival (see Algorithm 5.2). The evaluation process of the offsprings Q loops over the targets V in the order of τ using the counter variable k . In each iteration, the target $v \leftarrow V[\tau_k]$ is then evaluated for the offsprings and copied over for the population. After removing the current target v from the remaining targets $V^{(U)}$, the new survival probability $\alpha^{(k)}$ is calculated, and the survival error ρ_v determined. The survival error is the mean absolute error between the two different α 's and represents the error introduced by the prediction of target v . Afterward, the surrogate used for the predictions of target v is updated and its prediction error set. In the end of each target iteration, offsprings with a survival probability $\alpha_i^{(k)} \leq \alpha^{(\min)}$ are eliminated. The elimination of unpromising offsprings saves time because their evaluation of remaining targets is skipped over.

At the end of the evaluation process, the deterministic survival of the remaining fully evaluated offspring population and the current population is performed. In some iterations, no offspring might be left due to the iterative elimination, and one might wonder if the algorithm can be caught in a deadlock. However, in such iterations, the surrogate for each target has been updated using the already eliminated but partially evaluated offspring solutions. The procedure is then repeated until the time limit of running the algorithm has been reached. A time limit as a termination criterion instead of counting solution evaluation is recommended because it considers full and partial evaluations of individuals. Let us have a close look at one iteration of HE-EA for a bi-objective optimization problem having two objectives (f_1, f_2) and one constraint (g_1) . Let us assume that the target order has been determined to be (f_1, g_1, f_2) . A flowchart diagram

illustrates the process of evaluating the offspring population (see Figure 5.7). Initially, the current population P was copied and predicted by the surrogate to become P' . This step can be essential because comparing only predictions with predictions ensures comparing only apples with apples and oranges with oranges. However, if all surrogates fit through the data set exactly, this step is skipped. After generating the predicted population, the offspring population Q is created by surrogate-guided mating, and their predictions are available. In the flowchart, targets predicted by surrogates are shown in blue, and the ones with exact values are shown in orange. In the first iteration of the elimination-based evaluation, the parent and the offspring population are predicted by the surrogate, and the survival probability of $\alpha^{(0)}$ is calculated.

After evaluating the first target $v \leftarrow f_1$, the survival probability $a^{(1)}$ having no error noise for f_1 is predicted. The survival error ρ_{f_1} is then determined by the mean absolute error of survival probabilities. Before moving on to the next target, all offspring members with $a^{(1)}$ less than $a^{(\min)}$ are eliminated. Then, the target evaluation is continued by evaluating g_1 , performing probabilistic survival result in $a^{(2)}$ and updating the surrogate error ρ_{g_1} . Analogously, in the last iteration, f_2 is evaluated. It is worth pointing out that $\alpha^{(3)}$ does not require any probabilities survival because after having evaluated all targets, no noise will be added, which results in a deterministic survival procedure. Thus, the survival probabilities are either zero or one. Finally, the population and the fully evaluated offspring population are sent to the survival operator to determine the population for the next iteration. Notice that the above algorithm also degenerates to the single-objective constraint problems with heterogeneous evaluation times among objective (f) and multiple constraints (g).

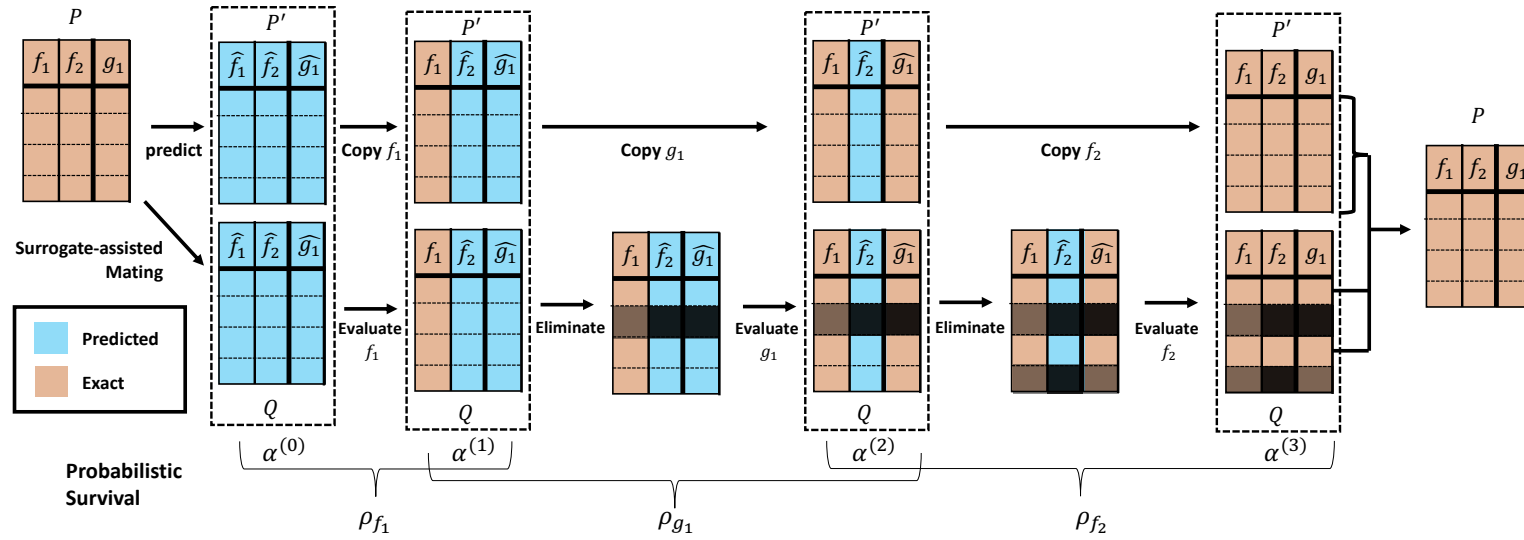


Figure 5.7: One iteration of HE-EA consisting of an ordered target evaluation (f_1, g_1, f_2) and offspring eliminations.

5.5.2.4 Surrogate Management

Even though surrogate modeling is not the focus of this study, a few words need to be said to complete the algorithm description. In this study, each target value is modeled independently to avoid any error accumulation across targets. Thus, in total $|M| + |J|$ surrogate models are built [23]. For each target, different surrogate models are fitted, and then one with the least mean absolute error is chosen as a predictor. We have used two different types of models: Radial Basis Functions (RBFs) [34] and Kriging [35, 55]. Each of them is instantiated with different hyper-parameters, resulting in a list of potential models from which one is selected. It is worth mentioning that the number of data points for each model may vary because of partial evaluations. In order to decrease the computational burden of surrogate fitting, in this study, only the previous 200 data points are considered. The mean absolute error is estimated in each iteration by considering the data points as the training set and the infill solutions evaluated as a validation set. This allows a realistic error estimation in each iteration.

5.5.3 Results and Discussions

The capability of the proposed method to exploit independently computable and heterogeneous expensive optimization problems shall be examined next.

We have chosen NSGA-II [10] with a population size of 100 for unconstrained and constrained bi-objective problems. For problems with three objectives, we have used NSGA-III [279, 258, 12] with 91 reference directions originating from uniform weight sampling [280] with a partition number of 12. For both algorithms, the default parameter settings proposed in the papers and defined in their implementations available in the multi-objective optimization framework in Python called pymoo [29] is used. In each experiment, three different algorithms are compared with each other. First, the baseline algorithm without any modifications: This represents an optimization method waiting for all targets to be evaluated despite their heterogeneous evaluation times. Second is a modification of the baseline algorithm incorporating the Elimination-Based Evaluation (EBE) with the default mating procedure where the survival probabilities have been assessed by repeating

the survival a hundred times ($\gamma = 100$). The third is the complete heterogeneously expensive (HE) evolutionary algorithm with a surrogate-guided mating ($\beta = 30$) and elimination-based evaluation, where the consideration of two different variations shall help to give credit to eliminating unpromising solutions during evaluation and creating solutions more likely to survive beforehand separately. Throughout the experiment, we have fixed the minimum survival probability to $\alpha^{(\min)} = 0.3$. Some numerical experiments have shown that this seems to be a reasonable value for discarding unpromising individuals. When proposing a new method, the number of hyper-parameters shall be kept as small as possible. It is worth mentioning that in this study, the hyper-parameters γ , β , and $\alpha^{(\min)}$ have an intuitive meaning which helps to set them properly. Their values were determined through an empirical study, and the performance of configurations close to the suggested one has shown to be similar. Thus, no significant sensitivity could be observed.

The performance of the proposed methods is assessed on unconstrained and constrained multi-objective test problems where the time for objective and constraint functions has been systematically varied. We have conducted 11 runs for each problem and algorithm to address the stochastic behavior of the underlying optimization method. All tables presented in this section show the average IGD values [240]. The best-performing algorithm for each problem is marked as the winner (*), and other algorithms performing significantly similar (Wilcoxon signed-rank test, $p = 0.05$) are labeled by (\approx), and ones are performing significantly worse by ($-$). Moreover, we have denoted the number of variables by N , the number of objectives by M , and the number of constraints by J for each problem. Starting with bi-objective problems, we have used the ZDT [253] problem suite with the evaluation times for a solution are fixed to 20 time units. The overall evaluation budget is set to seven hours for ZDT1-3 and to 10 hours for ZDT4. This equals 13 and 18 generations of fully evaluated individuals, respectively. For the experiment, the evaluation time of the first objective is set to 1, 5, 15, or 19, and the one of the second complementary to make their sum equal to 20. The results are shown in Table 5.2. First, one can note that the different evaluation times always lead to identical results for the baseline algorithm, caused by the algorithm waiting for all targets to be evaluated before proceeding. Second, EBE and HE were both able to outperform the NSGA-II

Table 5.2: Average IGD values for unconstrained bi-objective problems from the ZDT test suite. NSGA-II does not use heterogeneous evaluation time information, hence produce identical IGD value for all different evaluation time combinations.

$t(f_1, f_2)$	NSGA-II	EBE-NSGA-II	HE-NSGA-II
ZDT1			
$(N = 10, M = 2, J = 0)$			
(1,19)	0.3258 (−)	0.1053 (−)	0.0166 (*)
(5,15)		0.1078 (−)	0.0169 (*)
(10,10)		0.1162 (−)	0.0120 (*)
(15,5)		0.0968 (−)	0.0119 (*)
(19,1)		0.0882 (−)	0.0099 (*)
ZDT2			
$(N = 10, M = 2, J = 0)$			
(1,19)	0.6457 (−)	0.1942 (−)	0.0099 (*)
(5,15)		0.2303 (−)	0.0107 (*)
(10,10)		0.2123 (−)	0.0139 (*)
(15,5)		0.2233 (−)	0.0148 (*)
(19,1)		0.2316 (−)	0.0143 (*)
ZDT3			
$(N = 10, M = 2, J = 0)$			
(1,19)	0.2009 (−)	0.0854 (−)	0.0376 (*)
(5,15)		0.0930 (−)	0.0474 (*)
(10,10)		0.0777 (−)	0.0348 (*)
(15,5)		0.0597 (−)	0.0324 (*)
(19,1)		0.0540 (−)	0.0194 (*)
ZDT4			
$(N = 5, M = 2, J = 0)$			
(1,19)	27.7984 (−)	19.4416 (*)	20.4623 (≈)
(5,15)		21.1689 (≈)	17.5283 (*)
(10,10)		17.7420 (≈)	16.9622 (*)
(15,5)		16.2275 (*)	16.2289 (≈)
(19,1)		14.9382 (*)	15.5179 (≈)
Total	0 (*)	3 (*)	17 (*)
	0 (≈)	2 (≈)	3 (≈)
	20 (−)	15 (−)	0 (−)

no matter what time variation of $t(f_1, f_2)$ has been chosen. By comparing EBE and HE with each other, one can conclude that increasing the probability of a solution surviving during mating is in general helpful. For ZDT1 and ZDT2, much better results, and for ZDT3, still significantly better results have been achieved. For ZDT4, none of the methods can converge close enough to the

Table 5.3: Average IGD values for the constrained bi-objective problem TNK.

TNK $(N = 2, M = 2, J = 2)$			
$t(f, g)$	NSGA-II	EBE-NSGA-II	HE-NSGA-II
(1,19)	0.0214 (−)	0.0034 (*)	0.0043 (≈)
(5,15)		0.0035 (−)	0.0030 (*)
(10,10)		0.0040 (−)	0.0031 (*)
(15,5)		0.0043 (−)	0.0030 (*)
(19,1)		0.0046 (−)	0.0030 (*)
Total	0 (*)	1 (*)	4 (*)
	0 (≈)	0 (≈)	1 (≈)
	5 (−)	4 (−)	0 (−)

true Pareto front, given the limited evaluation budget. Thus, for ZDT4, even though both methods outperform NSGA-II, no clear winner can be declared. Altogether, we can conclude that for the considered unconstrained bi-objective problems, HE-NSGA-II shows the best results by winning 17/20 test instances and being significantly similar in the remaining ones.

One benefit of the proposed approach is considering groups of targets and the capability of extending the concept of heterogeneously expensive functions to multiple objectives and constraints. This will become apparent when discussing the following constrained multi-objective problems. First, we investigate TNK [9, 261], which has two objectives and two constraints and a discontinuous Pareto front. We have considered all objectives and all constraints, each as a group of targets. This imitates the real-world scenario where Software A returns both objective values and Software B calculates the constraints.

The evaluation time variations have been set analogously to the unconstrained bi-objective problems, and for each run, the time limit is set to three hours. The results listed in Table 5.3 show the superiority of EBE and HE over the NSGA-II. Across all time variations, EBE and HE converge to the Pareto-optimal set. For four out of five problems, HE-NSGA-II turns out to be significantly the best performing method. Interestingly for $t(f, g) = (1, 19)$ representation, the case of objectives being much less computationally expensive than constraints, EBE performs marginally better.

For the welded-beam design problem [281], the first objective f_1 is the cost of fabricating the

Table 5.4: Average IGD values for the constrained bi-objective problem Welded Beam.

Welded Beam			
$(N = 4, M = 2, J = 4)$			
	NSGA-II	EBE-NSGA-II	HE-NSGA-II
t	0.078 (-)	0.0577 (-)	0.0168 (*)

welded beam and can be written in a closed-form mathematical term. Thus, it is relatively quick to compute. The second objective f_2 and constraints g_1 and g_2 are the deflection of the beam-end and hence belong to the same target function group. Constraint g_4 is less time-consuming, but g_3 is the buckling load, requiring more computational effort. Following relative computational times are considered for the target functions: $t(f_1) = 1$, $t(\{f_2, g_1, g_2\}) = 12$, $t(g_3) = 12$, $t(g_4) = 1$. Again, the time limit has been set to three hours.

Overall, the results indicate that HE-NSGA-II performs significantly better than the two competitors. However, it is also worth mentioning that the difference between the average IGD values is relatively small. Some further analysis has shown that is caused by the $\{f_2, g_1, g_2\}$ groups of targets being responsible for a relatively high survival prediction error. Therefore, even though their evaluation is more time-consuming than others, they are scheduled first when the order of targets τ is determined. This clearly shows the challenge of defining the evaluation order consisting of more complex target groups with mixed complexity and expense.

Moreover, the proposed methods shall be analyzed for a DTLZ2 [257], an unconstrained three objective optimization problem. The experiment is set up that the evaluation times of all objectives f_1 , f_2 , and f_3 sum up to 30. In total, we have run the methods for 16 different combinations, varying the expensiveness from being completely homogeneous (10,10,10) to one objective being 28 times more computationally expensive to evaluate than the cheapest function. The time limit for each run has been set to four hours. The average results IGD values for all time variations are shown in Table 5.5. Whereas EBE improved the performance from NSGA-III, incorporating a more sophisticated mating in HE-NSGA-III outperforms the other competitors significantly across all problems. This implies that the prediction of values during the run was quite accurate and that

Table 5.5: Average IGD values for the three-objective problem DTLZ2.

DTLZ2			
$(N = 10, M = 3, J = 0)$			
$t(f_1, f_2, f_3)$	NSGA-III	EBE-NSGA-III	HE-NSGA-III
(28,1,1)	0.2824 (–)	0.1992 (–)	0.0794 (*)
(1,28,1)		0.2053 (–)	0.0806 (*)
(1,1,28)		0.1926 (–)	0.0820 (*)
(25,4,1)		0.2051 (–)	0.0802 (*)
(25,1,4)		0.2077 (–)	0.0791 (*)
(1,25,4)		0.2050 (–)	0.0811 (*)
(4,25,1)		0.2017 (–)	0.0810 (*)
(1,4,25)		0.2015 (–)	0.0825 (*)
(4,1,25)		0.2011 (–)	0.0815 (*)
(15,10,5)		0.2068 (–)	0.0916 (*)
(15,5,10)		0.2095 (–)	0.0957 (*)
(5,15,10)		0.2021 (–)	0.0930 (*)
(10,15,5)		0.2088 (–)	0.0961 (*)
(5,10,15)		0.2049 (–)	0.0889 (*)
(10,5,15)		0.2027 (–)	0.0901 (*)
(10,10,10)		0.2047 (–)	0.1040 (*)
Total	0 (*)	0 (*)	16 (*)
	0 (\approx)	0 (\approx)	0 (\approx)
	16 (–)	16 (–)	0 (–)

putting some more bias into the offspring population has shown its effect. Another interesting fact is that the more heterogeneous the evaluations become, the better the results. Whereas for homogeneous times (10,10,10), HE-NSGA-III achieved an average IGD value of 0.104, which is decreased to 0.0794 for (28,1,1).

Lastly, some visualizations of objective space from different optimization problems are discussed. Figure 5.8 shows the median performing runs for the baseline algorithms (NSGA-II or NSGA-III) and their EBS and HE variants. We have chosen some representative evaluation times for each problem. The scatter plots confirm the discussion of the results based on IGD values and give the reader an idea of the differences in convergence and diversity to expect by exploiting the heterogeneity. For ZDT1-3 (see Figure 5.8a to 5.8c), one can observe the significant difference between the baseline approach and the proposed variants. For ZDT4 (see Figure 5.8d), the limit evaluation budget was not sufficient to converge for any method. It is worth noting that the figure

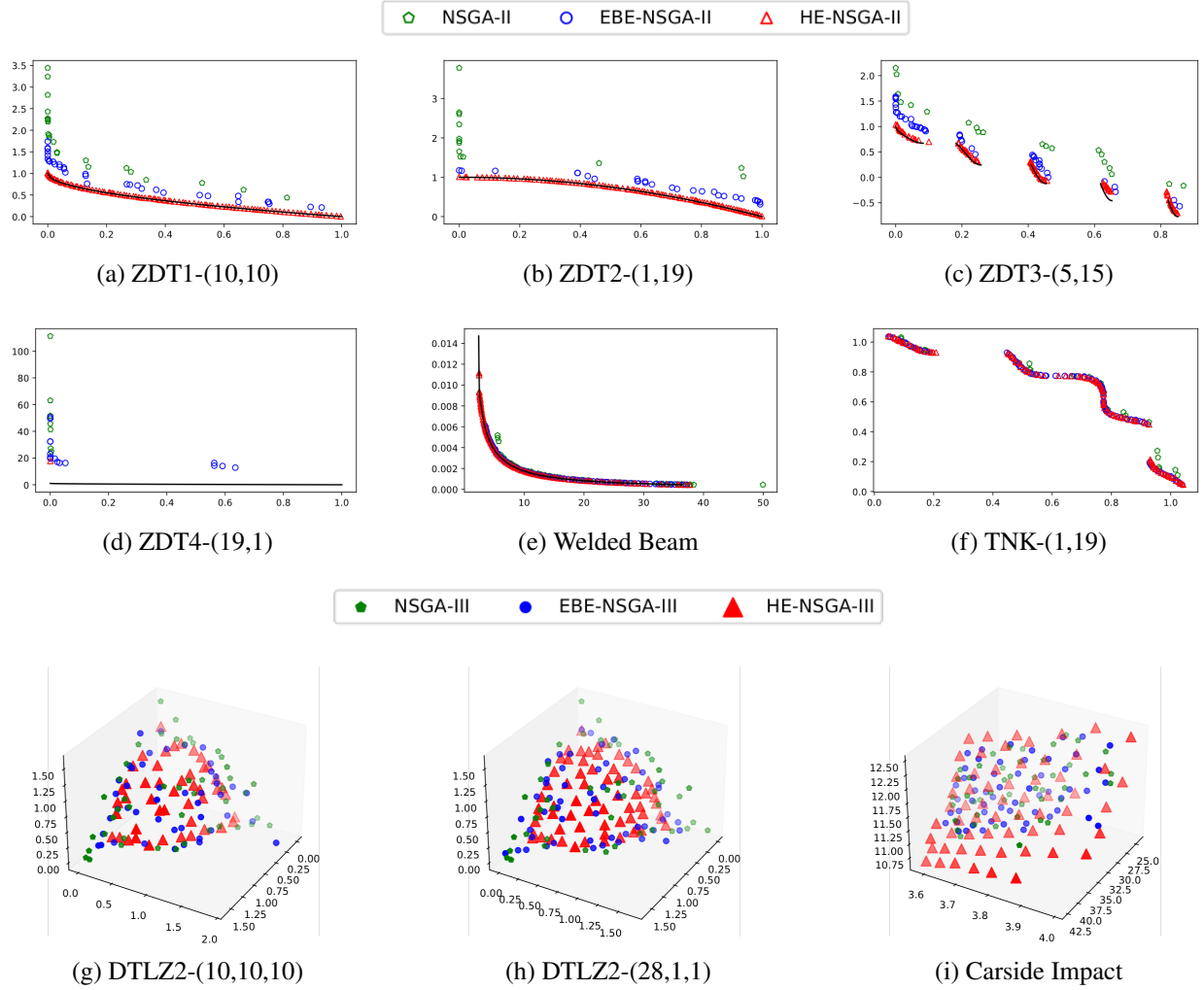


Figure 5.8: An illustration of the objective space for different types of unconstrained and constrained multi-objective problems. The results are based on representative run of the median performance for each problem. The different expensiveness of target functions and termination criteria are set analogously to the other experiments. The visibly better red-colored points are obtained using the proposed HE-NSGA-III procedure.

reveals that EBE achieves a better diversity than HE. For Welded Beam (see Figure 5.8e), HE-NSGA-II can find more solutions with a smaller value of f_1 whereas for TNK (see Figure 5.8f), visually no significant difference can be observed. For DTLZ2, Figures 5.8g and 5.8h show that a larger amount of heterogeneity in fact helps to converge faster and find a more diverse set. Another problem where the diversity of solutions has been shown to be significantly better is the Carside Impact problem shown in Figure 5.8i. The problem consisting of three objectives and 10 constraints has been set up so that the objectives are computationally inexpensive and the constraints

computationally expensive. Altogether, the visual inspections of the median runs of the experiment show how the exploitation of heterogeneously expensive functions can improve the convergence of an existing algorithm.

5.5.4 Summary of Section 5.5

In this section, the differently expensive target functions have been exploited by an elimination-based evaluation which discards partially evaluated solutions based on their likelihood of surviving. The concept can consider each target function separately but also handle target groups. This can especially be useful for practitioners when a software package returns more than one objective or constraint to be used in optimization. Moreover, the proposed approach is applicable to other evolutionary methods where an elitist environment survival is incorporated. This has been demonstrated by adding the support of differently expensive targets to two well-known multi-objective algorithms. Results on unconstrained and constrained bi-objective and multi-objective problems indicate that the proposed method can efficiently exploit the fact of differently time-consuming target functions.

5.6 Summary of the Chapter

This chapter has investigated the evaluation of independently computable functions during optimization. Four different strategies for evaluating the objectives and constraints of a solution set have been proposed. Afterward, optimization problems with inexpensive constraints and expensive objectives were studied. The optimization has started with a well-spaced point set in the feasible space; the inexpensiveness of constraints is exploited by filtering out any kind of infeasible solution before evaluating the more expensive objectives. Then, problems with heterogeneity across constraints and objectives have been considered. There, the strategy of evaluating a set of solutions for a specific target (B/E) has been used to handle constrained multi-objective optimization problems with heterogeneous evaluation times. The proposed evolutionary algorithm has addressed the

order of targets during evaluation by sorting the targets by the survival prediction error divided by evaluation time.

This chapter shall also be extended in three directions. First, a further complication of the expensiveness of target functions is worth investigating. In this work, the evaluation time of each target function is kept constant, independent of the solution being evaluated. However, this does not need to be necessarily the case when solving real-world problems. Besides requiring a more sophisticated book-keeping approach of evaluation times, this also introduces another level of uncertainty to the target ordering problem, which needs to be addressed.

Second, the generational evaluation approach proposed in this proof-of-principle study can be extended to develop a steady-state method for further gain in overall computational time. In such a method, each offspring member can be evaluated with respect to the parent population in an appropriate adaptively obtained order of the target functions. Its acceptance and elimination can be determined using surrogate models.

Third, this chapter has focused on how to take advantage of heterogeneous functions for objectives and constraints within a population-based optimization algorithm. Although surrogate models are created and used to determine the order of evaluating target functions and eliminating evaluation of some population members depending on their multi-objective rank and evaluation time, the surrogate models themselves can be exploited further in arriving at better infill solutions. We defer such studies, which will eventually allow a complete surrogate-assisted heterogeneity-handling EC method that would be practically viable than the usual all-at-a-time evaluation-based algorithms. Nevertheless, our suggestion of a target function ranking scheme based on a member's worth in terms of non-domination and diversity in the population, accuracy of the prediction models, and actual computational times is unique and marks a start of the further future studies for solving heterogeneously expensive problems.

CHAPTER 6

REAL-WORLD APPLICATIONS

This chapter presents two case studies of real-world applications with computationally expensive objectives functions. The first case study consists of the optimization of a cylinder head water jacket design considering two objectives. The article has been originally published in [28] and is presented with minor modifications to ensure consistency throughout this thesis. Two of the three elementary questions of surrogate assistance, *What* and *With What*, have been answered analogous to Chapter 4. However, the *How* question is addressed by employing a local search on the surrogates using a trust-region approach. It is worth mentioning that this study is one of the outcomes of a two-year collaboration with a well-known automobile company.

The second case study addresses the optimization of the design of an electric machine with two objectives and various (geometric) constraints. Analogously to the first case study, the notation has been adapted for consistency based on the original article published in [282]. In total, 10 design variables with a precision of two decimal places are considered. The problem consists of computationally expensive objectives but much less time-consuming constraints. The inexpensiveness of constraints is an important fact to be exploited by the optimization method. Both case studies shall give some insights into how to deal with computationally expensive problems in practice.

6.1 Case Study I: Cylinder Head Water Jacket Optimization

6.1.1 Introduction

Evolutionary algorithms are robust optimization tools that can handle different challenges of practical optimization problems such as uncertainty [283], multimodality [284], constraints [263], and conflicting objectives [9]. These algorithms do not require any major simplification of the actual problem, a matter which is often required by the classical point-based methods [285]. However, this flexibility usually comes at the cost of a requirement for a high number of solution evaluations.

For example, the evaluation budget in BBOB2009 test suite was set to $10^6 D$ [286], in which D is the number of variables in the problem.

In many applications, commonly referred to as computationally expensive problems, the evaluation of each solution requires a significant amount of computation time and effort. For example, in optimal mechanical design, the evaluation of each candidate design may require a finite element or computational fluid dynamic analysis, which can take anywhere from a few hours to a few days. In some extreme cases, a solution evaluation may even require a costly prototyping and experimental testing process [287]. The computation time of a design evaluation is the bottleneck of the optimization process even if the evaluation process can be parallelized; therefore, in such applications, the evaluation budget is generally limited to a few hundred or less.

The goal in such problems is to make the most out of the available evaluation budget, which is fulfilling the design objective(s) as much as possible. A limited evaluation budget clarifies the importance of a careful selection of candidate solutions for evaluation, which motivates the use of surrogate models [104], also known as metamodels, function approximation [121], and response surface methodologies [89].

One decisive feature of surrogate-assisted algorithms is the metamodel update, which is also known as model management or evolution control [105]. Quite often, a single metamodel is used, which is updated during the optimization process [121]. The most suitable metamodel is not known beforehand and the choice of the metamodel is often based on its popularity in the related field. Such metamodel-assisted algorithms are generally less accurate than more sophisticated methods which employ an ensemble of metamodels. Updating the surrogate is more complicated in the latter group and requires a metric to assess candidate metamodels in order to select the most suitable metamodel at each cycle. Another prominent algorithmic aspect of surrogate-assisted methods is the way to select new solutions for high-fidelity evaluation, which is referred to as infill criterion or infill strategy [121]. Regarding recent development in parallel computing, it is practically crucial that a surrogate-assisted method be able to select several infill solutions so that they can be evaluated in parallel to reduce the optimization process wall-clock time.

This section develops a surrogate-assisted evolutionary algorithm for single and multi-objective optimization problems. This method, called Proximity-Based Surrogate-Assisted Evolutionary Algorithm (PSA-EA), selects new candidate solutions following two goals. First, these solutions should optimize the predicted objective values, and second, they should maximize the information collected about specific regions of the problem landscape. This industry-motivated method is tailored to problems with the following features:

- The number of high-fidelity solution evaluations is limited to 50-100.
- The number of variables varies between 5 and 10.
- The required computational time for each solution evaluation is high (a few minutes to a few days); therefore, the computation time for other parts of the optimization process is negligible.
- The number of cycles is limited; thus, the method must be able to select several infill solutions in parallel.

6.1.2 Proposed Proximity-Based Surrogate-Assisted Optimization Method

This section elaborates Proximity-Based Surrogate-Assisted Evolutionary Algorithm (PSA-EA) for computationally expensive single-objective and multi-objective problems.

6.1.2.1 Selection of Initial High-Fidelity Solutions

The selection of initial high-fidelity solutions is an important phase because the sampling strategy directly affects the goodness of the initial metamodel. The space-filling approach in this section is based on the *Maximin* method [288], which aims to maximize the minimum distance among all solutions. The employed algorithm in this study is an adaption of the initialization procedure in [289]: The first initial point is generated randomly. Then, a new random point is sampled and accepted if it maintains a distance of at least R_0^{ini} from already selected solutions; otherwise, it is discarded and a new random point is generated. If several successive attempts are rejected (e.g.,

100 attempts), R_0^{ini} is slightly reduced ($R_0^{\text{ini}} \leftarrow 0.99R_0^{\text{ini}}$). This process continues until all initial high-fidelity solutions are generated. The distance variable R_0^{ini} is initially set to a conservatively large value, which is half of the maximum distance that can exist in the search space between two points ($R_0^{\text{ini}} = 0.5\|X^U - X^{(L)}\|_2$).

6.1.2.2 Parallel Infill Strategy

Evolutionary algorithms should be able to maintain a reasonable trade-off between exploration and exploitation. The likelihood of missing the global optimum increases if the search is not exploratory enough. On the other hand, when focusing more on exploration, no time might be left to exploit the gathered information about the design space. Evolutionary algorithms generally focus on exploration in the beginning and gradually emphasize exploitation of the available information. For example, the initialization of the population is an entirely exploratory process since the objective values of the solutions are not considered.

Finding a reasonable trade-off between exploration and exploitation is more complex for parallel infill strategies. If exploration is ignored, infill solutions may be selected very close to each other. To address this issue, Sóbester et al. [290] performed multimodal optimization of the surrogate model and selected the optima of the approximate function as new infill solutions. This strategy, however, limits the number of infill solutions in a cycle to the number of optima in the approximate function. Furthermore, this strategy might be too exploratory for the final cycles, when the method should make the most out of the existing information. To address this issue, Zhan et al. [291] allowed selection of more infill solutions in the vicinity of fitter optima; however, their method needed to tune a threshold value.

As discussed, each new solution provides some new information on the landscape of the objective function(s). If this solution is far from the existing ones, this information will be significant. For surrogate-assisted methods, this information can provide an extra contribution: It can improve the goodness of the metamodel in subsequent cycles if the new infill solution is relatively far from the existing high-fidelity solutions. To take the diversity of new infill solutions

into account, this study defines an infeasible spherical region with radius R_{prox} around each existing high-fidelity solution so that subsequent infill solutions are not selected close to existing ones. A greater R_{prox} enforces that the new infill solutions considerably contribute to improving the goodness of the surrogate model in future cycles (primary goal). Solutions with good predicted values are preferred only if they are sufficiently far from the existing high-fidelity solutions (secondary goal).

The accuracy of the predicted values strongly depends on the goodness of the metamodel. Predictions close to observations are more accurate than predictions far from them. Therefore, the concept of trust regions [292] is used to control the amount of acceptable uncertainty in the search process. The trust region defines spherical regions of radius R_{trust} around each solution. Subsequent solutions for high-fidelity evaluation must be selected in the trust region. If R_{trust} is large, points with high predicted fitness are probably those with high uncertainty. This can mislead the search to regions where the metamodel is less accurate. The value of R_{trust} can control the exploitation of the optimization process: a small R_{trust} forces the algorithm to select new infill solutions close to existing ones, where the prediction error is small. Combining both concepts, a permissible search region is defined where all infill points must lie. $R_{\text{trust}} \geq R_{\text{prox}} > 0$ should gradually decrease to allow for a gradual transition from exploratory to exploitative search. In this study, R_{trust} is set to be proportional to R_{prox} .

Figure 6.1 shows the change in the permissible region during the optimization process. Areas with a horizontal hatch pattern represent the region defined by R_{prox} , whereas the region beyond the trust region is delineated by an inclined hatch pattern. The permissible search region is demarcated with a square-shaped hatch pattern. In the initial cycle, exploration is emphasized by having a large R_{trust} and a large R_{prox} . In intermediate cycles, both radii have decreased to improve exploitation. During the final cycles, both regions are relatively small to maximize exploitation.

Ideally, the rate of the reduction in R_{trust} and R_{prox} can be controlled by a single user-defined parameter. Although different functions can be used, an exponentially decreasing function is preferred in this study:

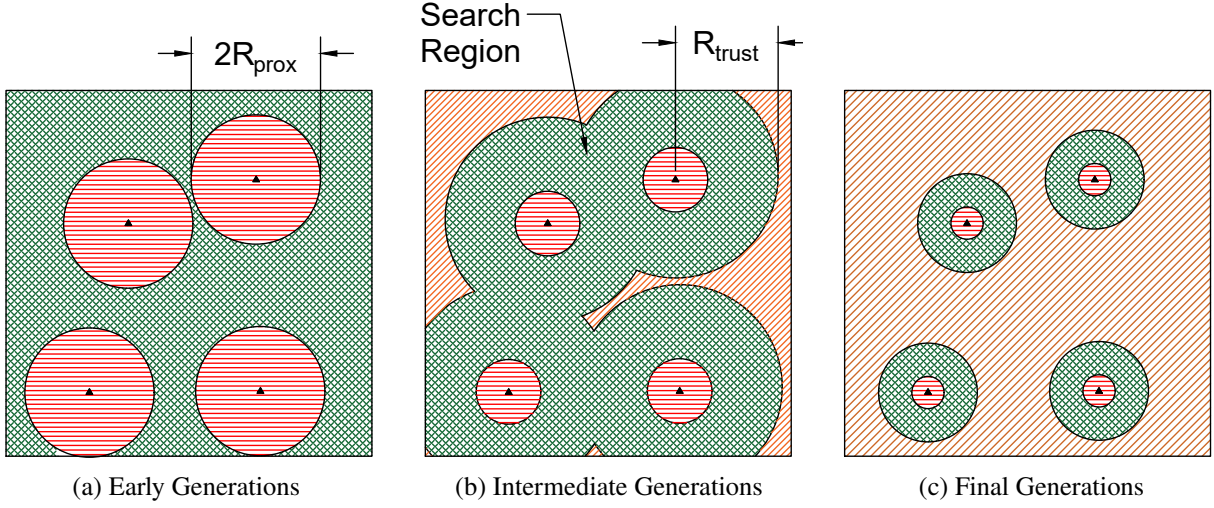


Figure 6.1: Adaptive trust region approach restricts the search within the cross-patterned region allowing the optimization algorithm to focus near high fidelity solutions.

$$R_{\text{prox}} = \max \left(R_0^{\text{ini}} \left(1 - \frac{FE - r_{\text{ini}} FE_{\text{max}}}{FE_{\text{max}} (1 - r_{\text{ini}})} \right)^{\tau_R}, \epsilon_{\text{prox}} \right), \quad (6.1)$$

$$R_{\text{trust}} = r_{R2R} \cdot R_{\text{prox}}, \quad (6.2)$$

in which R_0^{ini} is the minimum distance between two solutions after generating initial high-fidelity solutions (see Section 6.1.2.1), FE_{max} is the solution evaluation budget, FE is the number of evaluations so far, r_{init} is the fraction of the evaluation budget that was used for the initial high-fidelity solution, and τ_R specifies the reduction rate of R_{prox} . $\epsilon_{\text{prox}} > 0$ is the lower bound for the distance between two solutions. $\epsilon_{\text{prox}} = 10^{-6} \|\mathbf{X}^{(U)} - \mathbf{X}^{(L)}\|_2$, which is 10^{-6} times the euclidean distance of the upper minus the lower bound for each variable. This limit was defined solely to prevent having multiple infill solutions on or extremely close to each other. Besides, it ensures $R_{\text{trust}} > 0$. A parameter study is performed in Section 6.1.3 to find a good value for τ_R . Notably, R_{prox} and R_{trust} can be embedded into any optimization algorithm by adding constraints that declare solutions as infeasible when they are in the proximal or outside of the trust region.

6.1.2.3 Management of the Surrogate Model

PSA-EA employs an independent metamodel for each objective function. It utilizes the DACEFIT module [55], a Kriging-based metamodel provided by the Matlab Surrogate Model Optimization Toolbox. The module allows setting the regression type, correlation function, and the initial length scale(s) θ of the corresponding kernel. The regression can either be constant, linear, or quadratic. The quadratic option is excluded since it requires a large number of high-fidelity solutions, leaving two options for the regression type. For the correlation, an exponential, generalized exponential, gaussian, linear, spherical, or cubic spline function can be used (six options). The bounds for θ were set to $[10^{-6}, 10^2]$ for which the candidate initial values are set to $[10^{-5}, 10^{-4}, \dots, 10^0]$ (six options). Since the accuracy of DACEFIT depends on the proper selection of the metamodel parameters, the best parameter setting should be found for each objective at the beginning of each cycle. Each parameter configuration results in a different metamodel. From this perspective, PSA-EA can be regarded as a method that employs an ensemble of metamodels.

For the first cycle, the set of candidate metamodels includes all settings for the parameters of the metamodel, resulting in $2 \times 6 \times 6 = 72$ metamodels. Each candidate metamodel is trained and assessed, and the best one is selected for the current cycle. A fraction of the worst settings for the metamodel parameters is discarded such that in the last cycle, only two parameter settings are tested. For the case when there are 72 candidate solutions in the beginning, this fraction is:

$$r_{\text{drop}} = \left(\frac{2}{72} \right)^{(1/N_{\text{cycle}}-1)}. \quad (6.3)$$

This reduction in the number of candidate metamodels significantly decreases the training and assessing time by excluding metamodels unlikely to be a suitable in future.

A common strategy to assess a metamodel is to divide the existing high-fidelity solutions into training and testing data, e.g. 80% for training and 20% for testing, as performed by [136]. However, since the number of high-fidelity solutions is very limited, this study suggests using stratified n -fold cross-validation, according to which all existing solutions are used as training and testing sets. Each time one solution is excluded from the training set, and its value is predicted using the trained

metamodel. This process continues until the predicted values of all solutions are calculated. Having the actual and predicted values of each solution, the goodness of each metamodel can be assessed. The best-found metamodel is then selected and retrained using all available solutions as there is no more need for test data.

There are several measures for comparing the goodness of a metamodel. The most commonly used one [104] is Mean Squared Error (MSE):

$$E_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^n \left(\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i) \right)^2, \quad (6.4)$$

in which n is the number of available solutions and $\hat{f}(\mathbf{x}_i)$ and $f(\mathbf{x}_i)$ are the predicted and real values, respectively. This study, however, favors a goodness measure that takes only the rank of solutions into account after they are sorted according to their function values. Jin et al. [293] proposed a number of such measures based on the difference between the predicted and actual rank of (selected) solutions after they have been sorted according to their predicted and actual values. This study suggests a more intuitive measure, called selection error probability (SEP). SEP considers all pairwise comparisons of solutions and sums up the number of times that comparing the predicted values has resulted in incorrect identification of the better solution:

$$E_{\text{SEP}} = \frac{1}{0.5n(n-1)} \sum_{i=1}^n \sum_{j=i+1}^n q(\mathbf{x}_i, \mathbf{x}_j), \quad (6.5)$$

where

$$q(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1, & \text{if } (f(\mathbf{x}_i) - f(\mathbf{x}_j)) (\hat{f}(\mathbf{x}_i) - \hat{f}(\mathbf{x}_j)) < 0, \\ 0, & \text{otherwise.} \end{cases}$$

The number of possible pairs with n solutions is $0.5 \cdot n \cdot (n-1)$. For each pair $q(\mathbf{x}_i, \mathbf{x}_j)$ returns 1 if the comparison of two solutions using their predicted values is not the same as using their true values, otherwise it returns 0. A metamodel with a smaller E_{SEP} is considered as a better one.

Figure 6.2 illustrates how SEP works in a common situation in model-based optimization. The true function values are shown in blue and the predictions in red. The absolute metamodel error is the integral of the difference between both functions. The optimization algorithm aims

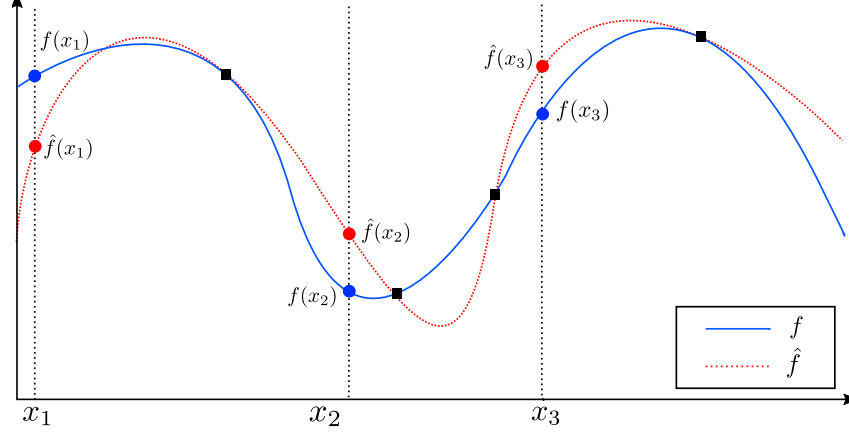


Figure 6.2: Selection Error Probability: Pairwise comparison between high-fidelity and prediction values.

to minimize $f(x)$ (True) by using $\hat{f}(x)$ (Prediction). The optimization algorithm will make pairwise comparisons. For instance, if point x_1 and x_2 are compared, then $f(x_1) > f(x_2)$ and $\hat{f}(x_1) > \hat{f}(x_2)$. The metamodel does predict the domination relation correctly. Contrarily, when x_1 and x_3 are compared, $f(x_1) > f(x_3)$, but $\hat{f}(x_1) < \hat{f}(x_3)$. The metamodel prediction leads to an incorrect comparison result. Considering minimizing using this metamodel the algorithm will favor solution x_1 over x_3 which is indeed wrong because $f(x_1) > f(x_3)$. This measures intuition and mimics the decisions of an evolutionary algorithm during optimization.

6.1.2.4 Optimization of the Approximate Function(s)

After selecting and training the best metamodel, an evolutionary algorithm is employed to perform optimization of the approximate function(s). In principle, any black-box optimization method can be used for this purpose. In PSA-EA:

- The Covariance Matrix Adaptation Evolution Strategy (IPOP-CMA-ES) with restarts [294] is used for single-objective problems. Since new solutions undergo high-fidelity evaluation at the end of each cycle, IPOP-CMA-ES is executed multiple times consecutively, generating one new point each time. This new solution does not affect the metamodel, but it changes the optimization problem landscape since each new point modifies the permissible region

defined by R_{prox} and R_{trust} . At the same time, R_{prox} and R_{trust} are updated whenever a new solution is generated.

- The recent unified version of Non-Dominated Sorting Genetic Algorithm III (NSGA-III) [279] is used for multi-objective problems. In this case, the solutions obtained by optimization are candidate solutions, from which some solutions are selected for high-fidelity evaluation. As a constraint, each solution must be in R_{trust} , but not be in R_{prox} . The selection procedure is similar to NSGA-III except that it uses a clearing strategy. Existing points are assigned to reference lines. The solution assigned to the least crowded reference line is selected for high-fidelity evaluation, and all points within the proximity region of this solution are cleared. If another point must be selected, but all points have been cleared, NSGA-III is run again after applying the effect of recent solutions on the permissible region. For practical problems, the user may manually select new points from candidate solutions.

Since the surrogate provides a computationally cheap prediction for a solution value, the budget of $5000D\sqrt{N_F}$ of approximate evaluations are provided per each infill solution, in which D and N_F are the numbers of decision parameters and the objectives in the problem. This setting is based on numerical experiments conducted by the authors.

6.1.2.5 Flowchart

Figure 6.3 presents the flowchart of PSA-EA. After generating initial high-fidelity solutions, an exhaustive search is performed to find the best metamodel. Then, a set of infill solutions is generated by optimizing the approximate function(s). The new infill solutions are used to update the metamodel(s), and this process continues until the budget for high-fidelity evaluation is depleted.

6.1.3 Descriptive Experiments

This section performs a few descriptive experiments to demonstrate the effects of different components of PSA-EA. For this purpose, four test problems were selected:

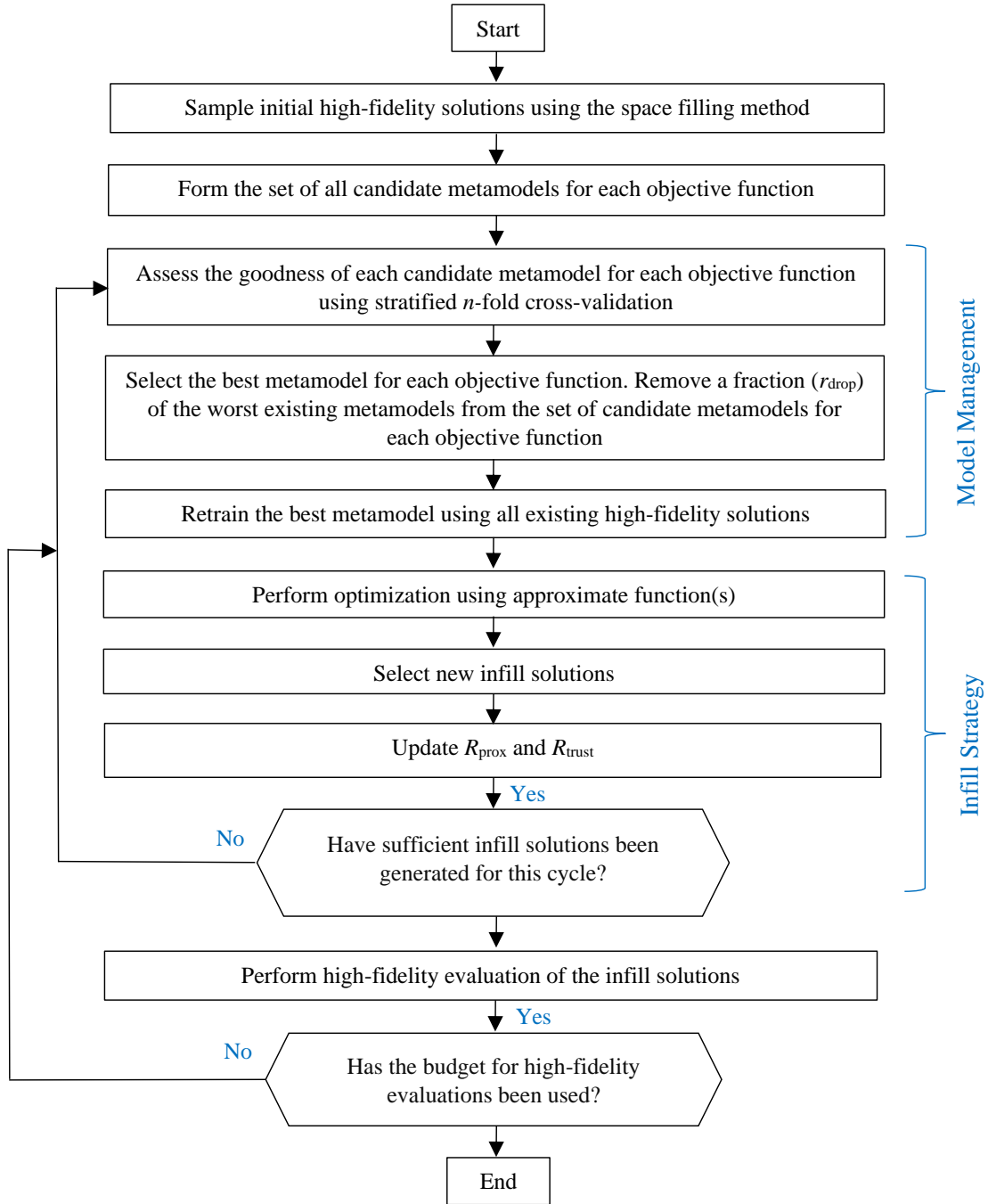


Figure 6.3: Flowchart of PSA-EA.

- Modified Rastrigin function, a highly multimodal function with a symmetric bowl-shaped global structure.
- Schwefel function, a multimodal function in which good local minima lie close to the corners of search space. Approaching any corner means going away from the rest of good local minima.
- shifted Ackley function, which shows a sudden reduction in the objective function value when approaching the global minimum.
- ZDT3 function, a two-objective test problem with a disjoint Pareto front.

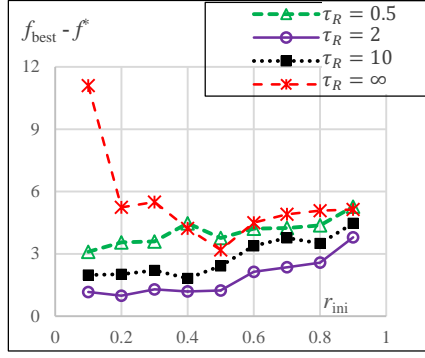
Each problem is optimized 50 times independently. For single-objective problems, the performance indicator measures the difference between the best value found by the method and the global optimum value ($f_{\text{best}} - f^*$). For multi-objective problems, the performance indicator is hypervolume ratio (HVR), which is the ratio of the measured hypervolume of the non-dominated solutions divided by the hypervolume of the true Pareto front. Therefore, $0 \leq HVR \leq 1$. The reference point (X_{ref}) for the calculation of HV is calculated as follows:

$$X_{\text{ref}} = \alpha_{\text{ref}} (X_{\text{nadir}} - X_{\text{ideal}}) + X_{\text{ideal}}, \quad (6.6)$$

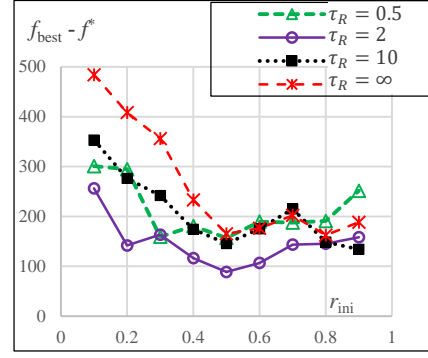
in which $\alpha_{\text{ref}} > 1$. For a reasonable value of α_{ref} , $HVR \approx 1$ means that the final solutions could provide a good approximation of the Pareto front, regardless of the shape of the problem and the nadir points. HVR is known to be a Pareto-compliant performance indicator [295]. By default, this study sets $\alpha_{\text{ref}} = 1.1$, as recommended by [296].

6.1.3.1 Effect of R_{prox}

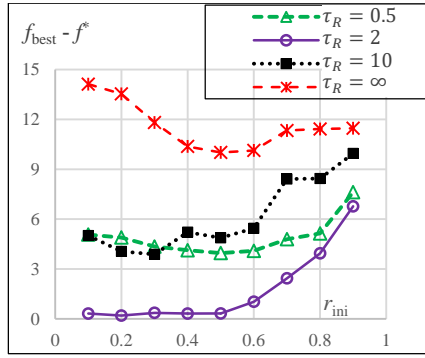
In PSA-EA, a larger τ_R makes a faster transition from exploration to exploitation. $\tau_R = \infty$, for example, suddenly reduces R_{prox} to ϵ_{prox} , which actually suppress the effect of R_{prox} concept. This section analyzes the effect of τ_R on the performance of PSA-EA. The budget of high-fidelity



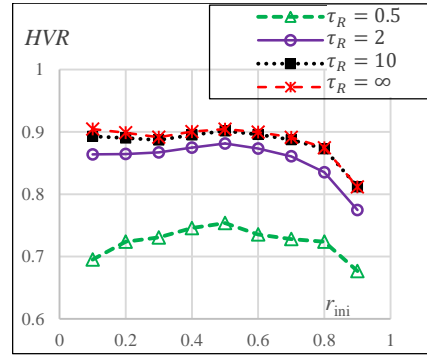
(a) Modified Rastrigin Problem.



(b) Schwefel Problem.



(c) Shifted Ackley Problem.



(d) ZDT3 Problem.

Figure 6.4: Effect of r_{ini} and τ_R on the performance of PSA-EA on each test problem when $FE_{max} = 100$ and $N_{cycle} = 100$.

solutions is set to $20D$, and different values for fraction of initial high-fidelity solutions (r_{ini}) and τ_R are tried. For this experiment, r_{R2R} is set to a very large value to suppress the effect of R_{trust} . Furthermore, only one solution is generated at each cycle. Figure 6.4 illustrates the performance metric for each setting. As it can be observed:

- In general, a proper τ_R can significantly improve final results. This is more detectable for Rastrigin, Schwefel, and Ackley functions. $\tau_R = 2$ results in significantly better final solutions, when compared to $\tau_R = \infty$ or $\tau_R = 0.5$.
- A gradual reduction of R_{prox} (for example, $\tau_R = 2$) improves the robustness of the method to the fraction of initial solutions (r_{ini}).
- When $\tau_R = \infty$, exploration is limited to the initialization phase. For a small value of r_{ini} , this

results in a more considerable performance drop compared to $\tau_R = 2$, in which exploration diminishes gradually. For the same reason, a higher r_{ini} can be beneficial when $\tau_R \rightarrow \infty$ since it improves exploration.

- Suppressing the idea of R_{prox} is advantageous for ZDT3 problem, possibly because of the simplicity of the objective functions in this problem or the fact that in multi-objective optimization, some diversity is automatically preserved in the selection process. Nevertheless, $\tau_R = 2$ is only a little worse than $\tau_R \rightarrow \infty$.

Consequently, a gradual reduction of R_{prox} , motivated by a gradual shift from exploration to exploitation, can improve both the quality of final solutions and robustness to the fraction of initial solutions.

6.1.3.2 Effect of N_{cycle}

A smaller N_{cycle} is desirable from an application point of view since it allows for the parallel evaluation of new infill solutions; however, it degrades the optimization performance by postponing the exploitation of true values of new infill solutions. Therefore, it is practically important to investigate the sensitivity of a surrogate-assisted method to this parameter. This section explores whether a gradual shift from exploration to exploitation can contribute to the performance of PSA-EA when N_{cycle} is limited. For this experiment, r_{R2R} is set to a very large value to suppress the effect of R_{trust} . Figure 6.5 demonstrates the median, the first quartile (Q1), and the third quartile (Q3) of $f_{\text{best}} - f^*$ or HVR as a function of N_{cycle} when $\tau_R = 2$. It reveals that:

- For the single objective problems, the performance substantially improves by increasing N_{cycle} up to $N_{\text{cycle}} = 5$. After that, the rate of improvement diminishes, and for $N_{\text{cycle}} > 16$, no significant improvement was obtained by increasing N_{cycle} . Knowing this trade-off makes decision-making easier on the value of N_{cycle} by considering the available computing resources.

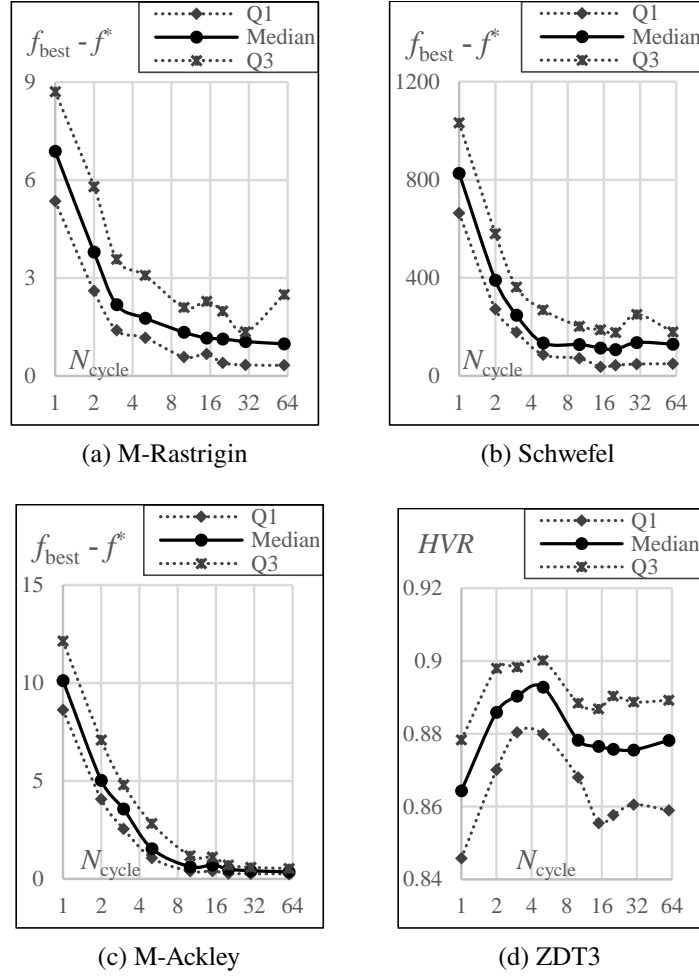


Figure 6.5: Effect of N_{cycle} on the performance of PSA-EA ($\tau_R = 2$, $r_{\text{ini}} = 0.4$) for the corresponding test problem.

- For the ZDT3 problem, N_{cycle} has no detectable effect on the performance. This demonstrates that the initial surrogate model, which is trained using the initial 40 solutions, is sufficiently accurate, and new infill solutions do not change the model much. This observation explains why for this specific problem, a gradual transition from exploration to exploitation was not beneficial in Section 6.1.3.1.

6.1.3.3 Effect of r_{R2R}

R_{trust} may enforce exploitation by limiting the search to solutions near the existing high-fidelity solutions. To check the effect of R_{trust} , this section optimizes four test problems using different

values of r_{R2R} . For this experiment, $\tau_R = 2$, $N_{\text{cycle}} = 20$, and $FE_{\text{max}} = 100$. Figure 6.6 shows Q1, Median, and Q3 of the performance indicator. As observed:

- for ZDT3 and Schwefel problems, a large value for r_{R2R} is advantageous. For these two problems, any $r_{R2R} \geq 4$ is a reasonable choice.
- No considerable effect of r_{R2R} can be detected for the shifted Ackley and modified Rastrigin problems.

The results of this section suggest that the default value of r_{R2R} should be equal or greater than four; however, no upper limit can be defined at this stage. This observation can be explained as follows: A smaller r_{R2R} results in a smaller difference between the predicted and the actual values of new infill solutions because it restricts the search to the neighborhood of existing high-fidelity solutions. On the other hand, this restriction limits the potential improvement from new infill solutions since farther solutions are disregarded even though the updated surrogate model predicts a good value for them.

6.1.4 Numerical Comparison

This section compares the performance of PSA-EA with two of the recently proposed surrogate-assisted optimization methods:

- Surrogate Optimization of Computationally Expensive Multi-Objective (SOCEMO) [297], a surrogate-assisted optimization method for multi-objective problems.
- Mixed Integer Surrogate Optimization (MISO) [228], a surrogate-assisted optimization method for mixed-variable single-objective problems.

One main reason for selecting these methods for comparison is the availability of their source code and documentation [298], which facilitates the simulation of arbitrary test problems. All parameters of MISO/SOCEMO are set to their default values, including the number of initial samples, which is set to $2D + 1$. To the best of the author's knowledge, no option to control the

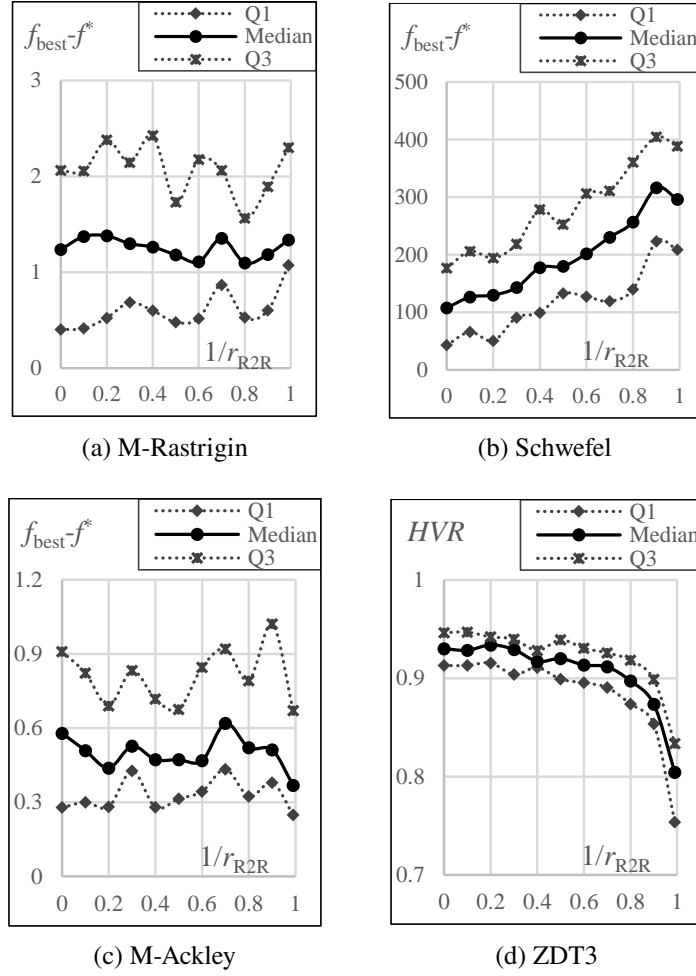


Figure 6.6: Effect of r_{R2R} on the performance of PSA-ES when $\tau_R = 2$, $N_{\text{cycle}} = 20$, and $FE_{\text{max}} = 100$.

number of cycles in these methods is provided. To ensure a fair comparison, the same number of initial samples is used for PSA-ES. Moreover, two values for N_{cycle} are considered for PSA-EA:

- $N_{\text{cycle}} = FE_{\text{max}}(1 - r_{\text{ini}})$, in which only one solution is generated and evaluated at each cycle. It allows for a fair comparison with MISO/SOCOMO. This variant of PSA-EA is denoted by PSA-EA^(S).
- $N_{\text{cycle}} = 8$, in which multiple solutions are generated in each cycle and evaluated in parallel. This practically demanding variant is denoted by PSA-EA^(P).

Four single-objective and six multi-objective test problems are employed in this study for

numerical evaluation and comparison. The single-objective test problems are some of the most commonly used test problems in the global optimization literature. This study uses slightly modified variations of these problems mainly by shifting the fitness function to relocate the location of the global optimum. The family of ZDT [253] and DTLZ [257] test problems are widely used in the multi-objective optimization literature. A few of them were excluded either because of excessive simplicity (ZDT1 and ZDT2) or similarity to other selected problems. The mathematical definitions of the customized test problems used in this study are provided in the supplementary document.

These test problems are optimized for $D = 5$ and $D = 10$, when $FE_{\max} = 10D$. This setting is based on the types of practical problems that motivated this study. For each problem, 50 independent runs are performed and the performance indicator, which was explained in Section 6.1.3, is reported for each method. By default, $\alpha_{\text{ref}} = 1.1$ (see Equation 6.6); however, for some more difficult problems, this setting may result in $\text{HVR} \approx 0$ for the tested methods. If so, this indicator may not determine which method has performed better. Alternatively, a greater α_{ref} was chosen such that at least one method can reach $\text{HVR} \geq 0.5$.

Wilcoxon rank-sum test with significance level $\alpha = 0.01$ is employed to check whether there is a statistically significant difference between the performance of PSA-EA and that of the other tested method (MISO for single-objective and SOCEMO for multi-objective problems). Table 6.1 presents the performance indicator for both methods when $D = 5$. It also provides the CPU time for each method to optimize all the problems once. The signs +, =, and – denote that PSA-EA^(S) or PSA-EA^(P) is statistically better, equal, and worse when compared with SOCEMO/MISO, respectively. The same data for $D = 10$ are provided in Table 6.2. The obtained results reveal that:

- PSA-EA^(S) outperforms MISO/SOCEMO for seven problems when $D = 5$, but it is outperformed by MISO/SOCEMO for two problems. For $D = 10$, PSA-EA^(S) outperforms MISO/SOCEMO in nine problems but it is not outperformed in any problem. This observation demonstrates that the superiority of PSA-EA^(S) over MISO/SOCEMO intensifies for problems of higher dimensions. For example, PSA-EA^(S) is outperformed by MISO/SOCEMO for 5-D M-DTLZ1, but it excels for 10-D version of this problem. In particular,

Table 6.1: Median of the performance indicator ($f_{\text{best}} - f^*$ for single-objective problems and HVR for multi-objective problems) and the outcome of the statistical test for the performance of the tested methods for $D = 5$. α_{ref} defines the selected reference point for calculation of HRV.

PID	Function	N_{obj}	α_{ref}	MISO or SOCEMO	PSA-EA ^(S)	PSA-EA ^(P)
1	M-Rastrigin	1	—	7.43	3.32 (+)	3.92 (+)
2	Schwefel	1	—	-1358	-1569 (+)	-1366 (=)
3	M-Ackley	1	—	6.31	3.36 (+)	6.37 (=)
4	M-Rosenbrock	1	—	110	250 (—)	512 (—)
5	ZDT3	2	1.1	0.507	0.882 (+)	0.883 (+)
6	M-ZDT4	2	7	0	0.597 (+)	0.522 (+)
7	ZDT6	2	3	0.485	0.535 (=)	0.435 (=)
8	M-DTLZ1	3	20	0.549	0.206 (—)	0.004 (—)
9	DTLZ2	3	1.1	0.686	0.888 (+)	0.886 (+)
10	DTLZ6	3	1.1	0.131	0.649 (+)	0.632 (+)
CPU time (minute)				4.7	51	36

Table 6.2: Median of the performance indicator ($f_{\text{best}} - f^*$ for single-objective problems and HVR for multi-objective problems) and the outcome of the statistical test for the performance of the tested methods for $D = 10$. α_{ref} defines the selected reference point for calculation of HRV.

PID	Function	N_{obj}	α_{ref}	MISO or SOCEMO	PSA-EA ^(S)	PSA-EA ^(P)
1	M-Rastrigin	1	—	19.64	6.40 (+)	8.52 (+)
2	Schwefel	1	—	-2644	-3088 (+)	-2619 (=)
3	M-Ackley	1	—	8.37	4.31 (+)	8.69 (=)
4	M-Rosenbrock	1	—	907	627 (=)	1249 (—)
5	ZDT3	2	1.1	0.077	0.936 (+)	0.935 (+)
6	M-ZDT4	2	25	0.147	0.668 (+)	0.535 (+)
7	ZDT6	2	5	0.546	0.609 (+)	0.577 (=)
8	M-DTLZ1	3	60	0	0.648 (+)	0.421 (+)
9	DTLZ2	3	1.1	0.493	0.830 (+)	0.837 (+)
10	DTLZ6	3	1.1	0.069	0.718 (+)	0.720 (+)
CPU time (minute)				11.2	449	180

PSA-EA^(S) significantly outperforms MISO/SOCOMO for M-Rastrigin, M-Ackley, ZDT3, ZDT4, and DTLZ6.

- Compared to MISO/SOCOMO, PSA-EA^(P) still excels in 11 problems and falls behind only in three problems.
- For some problems, a detectable performance drop can be observed for PSA-EA when the number of cycles is reduced to eight. In contrast, there is no considerable difference between PSA-EA^(S) and PSA-EA^(P) for PID=5, 9, and 10. Besides, both variants of PSA-EA could reach a relatively high HVR with the default value of α_{ref} . This implies that for these problems, even the initial surrogate model could predict the rank of solutions with good reliability. An efficient surrogate-assisted method is an ideal tool for optimizing such problems.
- Neither of the tested methods could provide a good approximation of the Pareto front of M-DTLZ1, even though the difficulty of the original problem was moderated; therefore, a large deviation from the recommended value of α_{ref} was necessary to obtain discriminating values for HVR.
- The CPU time for PSA-EA variants are much higher than that of MISO/SOCOMO; however, for problems in which each solution evaluation may take a few hours or more, the computation time of PSA-EA is still negligible.
- Compared to MISO/SOCOMO, PSA-EA^(P) has an important practical advantage: It can submit the new infill designs for external evaluation in a group of the desired size. Based on the author's evaluation, MISO/SOCOMO does not have this flexibility and requires the evaluation of new infill solutions to proceed. This is a critically important feature when it comes to practical problems.

6.1.5 Application: Cylinder Head Water Jacket Design

In this work, PSA-EA is employed to optimize the design of an engine cylinder head. In the following, the problem is described, the optimization procedure explained, and finally, the obtained results are discussed.

6.1.5.1 Problem Description

The problem has eight design parameters, which are the area of four inlets and four outlets of the cooling water jacket. These parameters are normalized with respect to the largest possible section. This way each area variable takes a value within $(x_i \in [0, 100], i = 1, 2, \dots, 8)$. In the base design, all inlets and outlets are set to their maximal size, which means $x_i = 100$ for all $i = 1, 2, \dots, 8$. Two conflicting objectives were defined – $f_1(x)$ to be maximized and $f_2(x)$ to be minimized. Each design evaluation requires a detailed CFD simulation which was performed by the engineering team at Ford. The CFD model consists of 2.4 million volume cells, 13.7 million interior faces, and 11.4 million vertices. The CFD analysis terminates after 4,000 iterations. Residuals such as continuity and energy are monitored but not used as convergence criteria.

Each design analysis takes about one hour using 32 CPUs. The optimization budget is limited to 61 design evaluations to complete the optimization task in a reasonable time. Furthermore, two different boundary conditions are considered for CFD simulation, resulting in two separate problems. They are named B34 and B38 here. These problems have no constraints except the bounds of the search range.

6.1.5.2 Results

Two optimization approaches are tested on problems B34 and B38 in parallel and independently. In the first approach, a commercial software is utilized by engine design engineers for surrogate model training and optimization. The selection of new infill solutions, however, is performed manually by the engineering team. This approach is denoted by CS (Commercial Software).

In the second approach, PSA-EA is employed by the research team at Michigan State University (MSU) for design optimization. For PSA-EA, the selection of new infill solutions is performed automatically by the algorithm, except for the final two cycles, in which the engineering team acted as decision-makers to choose preferred solutions from PSA-EA results. At the end of each cycle, three to five new solutions obtained by PSA-EA were sent to the engineering team at Ford for evaluation.

The CS approach was run independently by the engine design engineers, while PSA-EA was run at MSU with assistance on solution evaluation. It is worth noting that at the end of the ninth cycle, the search range for decision variables was manually reduced based on the range of non-dominated solutions.

Figure 6.7 shows the predicted values of new infill solutions, as well as their true values after CFD simulation for both methods (CS and PSA-EA) and both problems (B34 and B38). All obtained solutions by both methods are illustrated in Figure 6.8. The region of interest is focused in Figure 6.9, which also demonstrates the final solutions selected by the engineering team for fabrication and experimental testing. Based on the obtained results, the following conclusions can be made:

- PSA-EA and CS generate solutions that dominate most initial infill solutions (Figure 6.8). This can be considered a checkpoint for the validity of the optimization process, even when the evaluation budget is highly limited.
- The selected solutions for fabrication are from PSA-EA results for problem B34 and from CS results for problem B38 (Figure 6.9). For both solutions, f_2 is slightly greater than 10.
- The prediction error is initially high for both methods and both objectives. The error remains high for the CS method until the end but gradually reduces for the PSA-EA method with iteration (Figure 6.7). It is remarkable that the proposed approach, starting with wide discrepancies between true and surrogate model function values, match so well in a few iterations. The use of the trust region method and the overall surrogate modeling approach

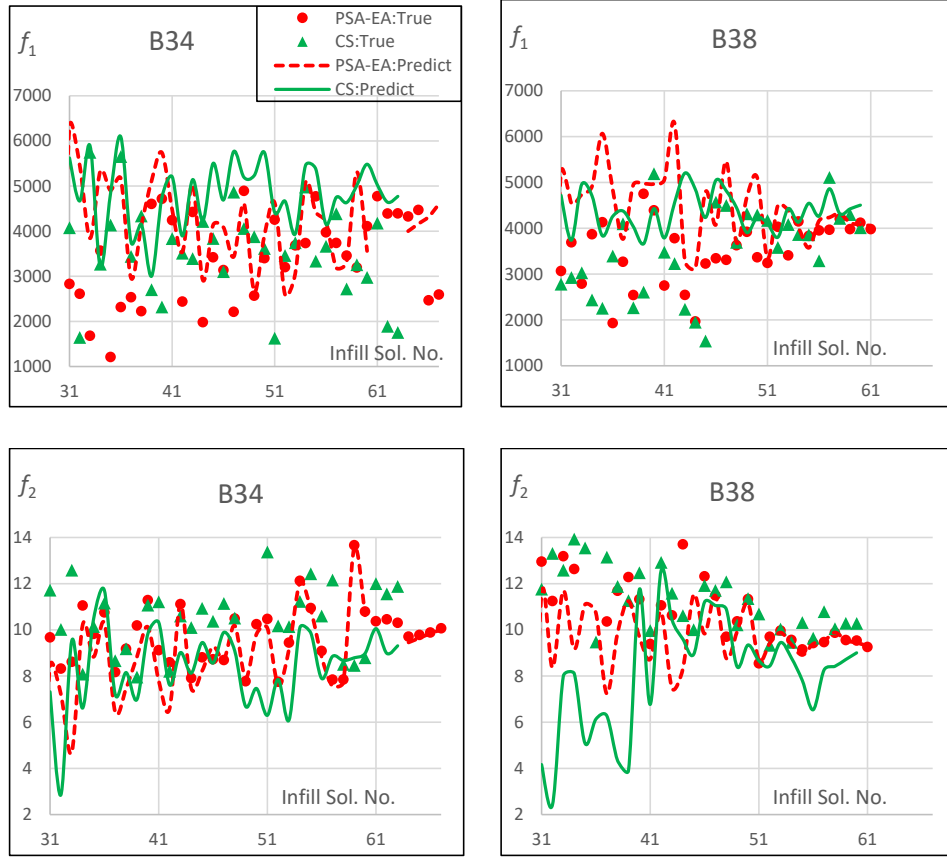


Figure 6.7: Predicted values of new solutions and their true values after CFD simulation for both methods, both problems, and both objectives.

make this reliable and remarkable correlation. This is more detectable for f_2 , for which the prediction error becomes almost zero in final cycles when using PSA-EA. This advantage of PSA-EA is presumably the result of a better exploration of promising regions in early cycles and better exploitation in the final cycles, and the manual reduction of the search range.

- Compared with the base design, the selected solutions show a maximum of 88% and 114% improvement of f_1 for problems B34 and B38, respectively. This is a piece of evidence for the superiority of design by optimization in comparison with intuition-based design methodologies.
- One interesting and unexpected feature of the selected design for problem B34 is that for this design, the size of one of the inlets/outlets is close to zero ($x_6 = 6.27$). This unex-

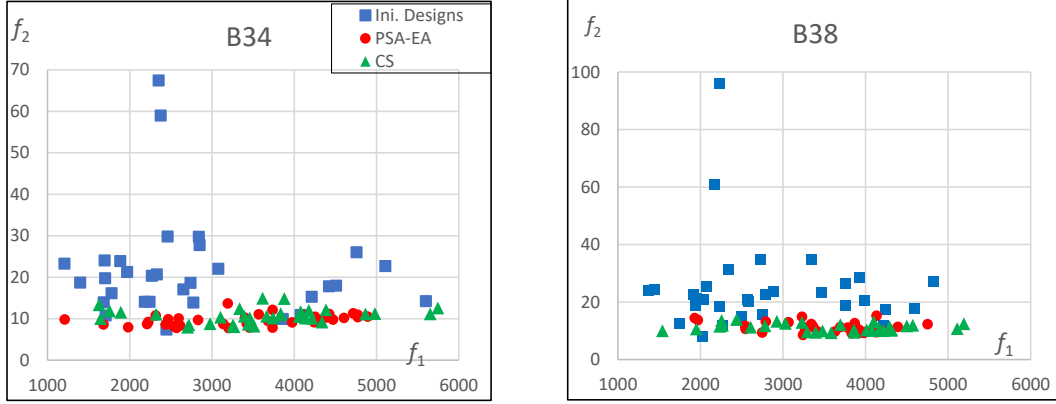


Figure 6.8: All generated solutions by PSA-EA and CS for problems $B34$ and $B38$.

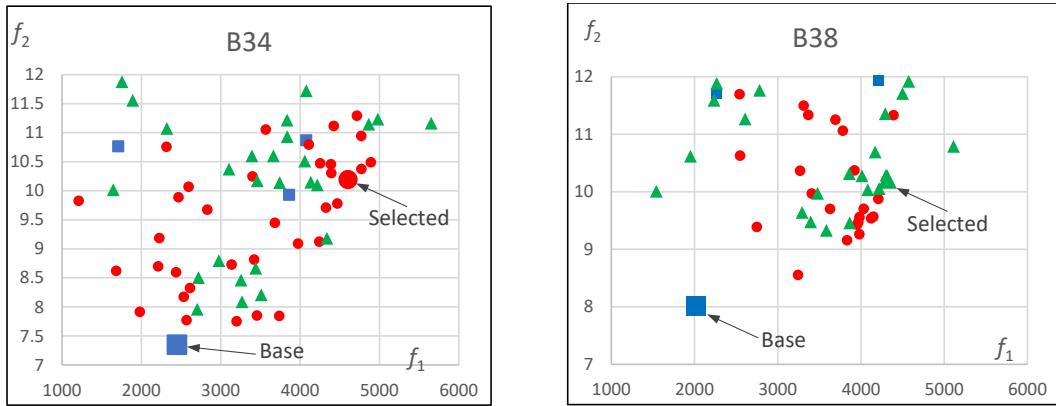


Figure 6.9: Generated solutions in the vicinity of the interest region for problems $B34$ and $B38$. The base and the selected designs for fabrication are demonstrated by arrows.

pected observation demonstrates the possibility of using optimization to develop innovative knowledge about key features of optimal solution(s). Such information can be used in earlier stages of design to determine the number of inlets/outlets or even their locations. Although simultaneous optimization of different features is more challenging than optimization of only sizes of inlets/outlets, it is predictably much more rewarding as well.

- For this problem, considering the soft constraint $f_2 \leq 10$ from the beginning could have been advantageous. It would automatically concentrate the search to the region of interest in the f -space; however, such information about the upper bound of f_2 may not have been known beforehand and may have resulted from the non-dominated solutions over cycles.

Although both methods were tested independently, the CS approach was run by the design

engineering team at Ford, who knew the designers' region of interest. This significant privilege helped to introduce a focus in the search to a small region, which provides more infill solutions for that region and improves the accuracy of the metamodel. For the PSA-EA, such information was provided and used only for the last two cycles, when the exploration phase was almost completed.

6.1.6 Summary of Section 6.1

In this work, we have developed a proximity-based surrogate-assisted evolutionary algorithm (PSA-EA) for single and multi-objective optimization of computationally expensive problems. PSA-EA selects new infill solutions according to their predicted fitness (exploitation) but imposes a constraint on their diversity to substantially improve the accuracy of the surrogate model for future cycles (exploration). It follows a gradual transition from exploration to exploitation by reducing the proximity radius over time. The importance of such a gradual transition has been numerically demonstrated, especially when the number of optimization cycles is limited.

The proposed surrogate-assisted method has been applied to optimize the cylinder head water jacket design. Besides the optimization procedure itself, it demonstrates what optimization in practice may look like. Considerations such as the existence of two different problems with similar goals, a reference design always compared to during optimization, solution evaluations being carried out via E-Mail communication, and introducing a soft constraint toward the end of the optimization run. Altogether, this should give the reader of this dissertation an idea of how diverse optimization in practice is and how such challenges can be addressed.

6.2 Case Study II: Electric Machine Design

6.2.1 Introduction

Electric machine design is an iterative process where each iteration focuses on improving the design's quality. Machines are complex systems where many variables, geometric and physical, interact non-linearly and affect the machine's performance. These performance measures can

include evaluating electromagnetic, thermal, and structural performances, making electric machines a multi-physics MOP. The need for design optimization is mainly application-driven, and since the field of applications for electric machines is large, there is much scope for improvement.

Research efforts in machine design optimization have been focused on improving solution accuracy and quality while reducing the optimization run-time. Some of the early electric machine design optimization studies include using pattern search and sequential unconstrained minimization techniques to optimize induction motor [299, 300]. However, it was demonstrated that evolutionary algorithms (EAs) are superior to point-by-point methods in finding global optimums for complex systems like electric machines [301]. Consequently, the use of EAs has increased with time in the optimization of machine design [302, 303, 304, 305, 306, 307]. Since the electric machine is a nonlinear system, finite element analysis (FEA) is the most preferred method for their evaluation. Examples of optimization studies where EAs were combined with FEA can be found [308, 309]. Since EAs require many function evaluations to converge to the global optimum, using FEA to evaluate each case during optimization requires substantial computational resources. In this regard, researchers have explored techniques, including surrogate-assisted optimization methods, to substitute FEA for reducing computation time; however, at the expense of solution accuracy [310, 311, 312, 313, 314]. For example, the authors used a combination of a second-order response surface model (RSM) and GA to optimize the magnet shape and placement in the rotor of an interior permanent magnet (IPM) machine to achieve the largest constant power speed region (CPSR) [310]. The second-order RSM was used to predict d-axis and q-axis inductances and magnet flux linkage of an IPM machine. Similarly, a surrogate-based optimization approach using multi-objective differential evolution (MODE) algorithm was employed to minimize active material mass and total losses of an axial flux PM (AFPM) machine at rated operation [313]. Moreover, a local refinement strategy improving a Pareto-optimal design further after termination of the optimization method was presented [315]. Results showed that a posteriori local search, even with fewer function evaluations, produced similar results compared to an approach solely relying on global optimization.

Although researchers have explored different optimization algorithms and methods to evaluate objective functions, constraint handling in literature for electrical machine optimization is inefficient. In optimizing electric machines, constraints define feasible regions in the design and objective space [316]. Typically, every machine optimization problem involves some geometric constraints that must be respected to generate feasible designs. Unlike objective functions, these geometric constraints can be quickly evaluated through analytical expressions. A procedure to identify the geometric feasibility of a candidate solution can be included in the optimization algorithm itself. A common approach to handle geometrically infeasible solutions is to discard them and rely on random initialization of variables and repeat this process until a feasible solution has been found [317]. However, random sampling becomes relatively inefficient as the number of geometric variables and constraints increases, which is eventually reflected in the quality of Pareto-optimal solutions when the computational resources are limited. To tackle this problem, we propose an embedded optimization problem where the information from constraint violation is used to repair geometrically infeasible solutions and improve the Pareto-optimal front.

The main contributions of this work are the following:

- The proposal of a repair operator ensuring the feasibility of designs. This operator exploits inexpensive constraints, e.g., geometric constraints calculated using analytical expressions, while respecting manufacturing accuracy limitations.
- A demonstration of improvement in quality of the Pareto-optimal solution set by integrating a repair operator into the optimization cycle. Additionally, a performance validation of the proposed surrogate assistance for predicting the computationally expensive objectives.
- A detailed physical explanation of Pareto-optimal solutions and recommendations for selecting preferred solutions based on two different approaches: (1) a domain specific a posteriori multi-criteria decision-making (MCDM) method which involves machine expertise, and (2) a trade-off analysis of the Pareto-optimal set.

Table 6.3: Parameters of IPM machine used for optimization.

Parameters	Values	Parameters	Values
Mechanical power	69 kW	Turns per coil	11
Rated speed	3000 rpm	Slot/ pole/ phase	2
Peak current	177 A	Slot fill factor	0.46
Stator outer diameter	264 mm	Air-gap	0.75 mm
Rotor outer diameter	160.4 mm	Stack length	50.8 mm
Average torque	214.8 Nm	Magnet type	NdFeB
Torque pulsations	36.2 Nm	DC-link voltage	650 V

The remainder of this case study is structured as follows. Section 6.2.2 discusses the formulation of the optimization problem. Section 6.2.3 explains the proposed optimization method, which exploits the computationally inexpensive constraints by introducing a repair operator and the computationally expensive objective functions by using surrogate models. The impact of the proposed repair operator and surrogate assistance on the convergence of the algorithm is discussed in Section 6.2.4. A detailed discussion about Pareto-optimal solutions and the selection of preferred electric machine designs is also included. Finally, conclusions are drawn in Section 6.2.5.

6.2.2 Electric Machine Design and Optimization Problem Formulation

The optimization of electrical machines typically starts with the selection of a machine template. Based on machine performance requirements, objective functions are formulated, followed by the selection of the design variables, variable ranges, and constraints. Average torque, torque pulsations, losses, and efficiency are some of the most common objective functions used in electric machine design. Designers usually select variables based on the domain knowledge and then proceed to sensitivity analysis to keep only the most significant variables during optimization [318]. Variable ranges can be selected arbitrarily or based on the machine designer's experience. Since the search space is generally high dimensional, a proper definition of geometric constraints can ensure the reliability of solutions.

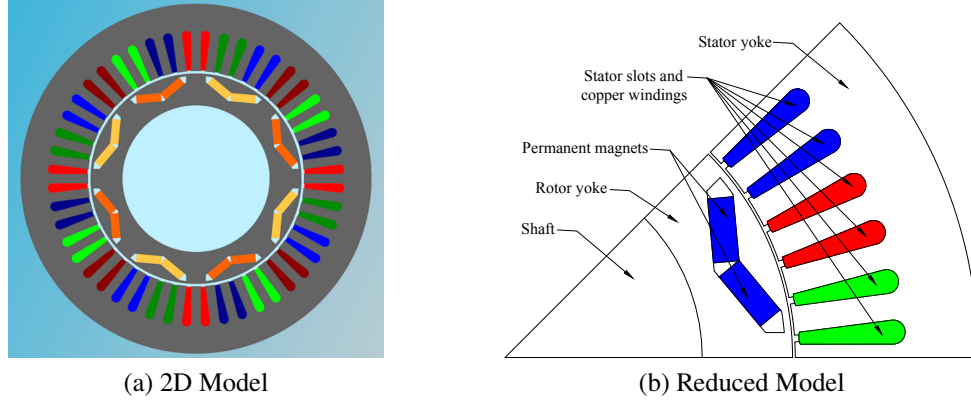


Figure 6.10: IPM machine used for optimization.

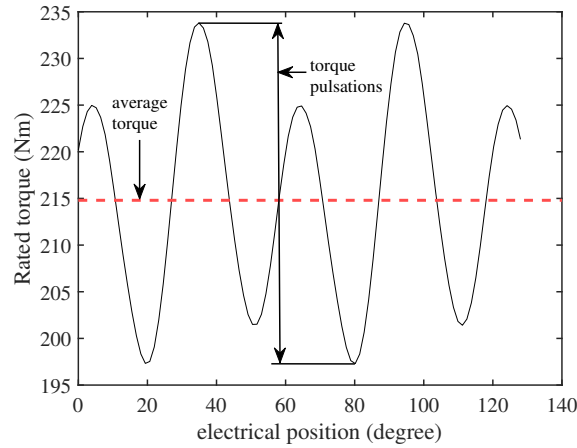


Figure 6.11: Torque profile of reference design at rated operating conditions.

6.2.2.1 Selection of Machine Topology, Objective Functions, and Evaluation Method

Permanent magnet synchronous machines (PMSMs) are known for their high torque density and efficiency. They are extensively used in hybrid and electric vehicle applications. In this work, an electrical machine used in the 2010 Toyota Prius is chosen for optimization. The machine is a 3-phase, 48-slot, 8-pole IPM machine with a single layer of a V-shaped magnet. Details of the machine parameters are given in Table 6.3 and a visualization is shown in Figure 6.10a and 6.10b [319]. Since IPM machines are highly nonlinear, FEA is chosen as the objective function evaluation tool. Because FEA is time-consuming, periodicity is exploited, and only $1/8^{\text{th}}$ of the model is considered for evaluation. The goal is to maximize average torque while minimizing torque pulsations, where pulsations are defined from peak to peak as shown in Figure 6.11.

6.2.2.2 Definition of Feasible Search Space

The following geometric variables are kept constant as in the original 2010 Prius motor:

- Inner diameter (ID) and outer diameter (OD) of stator and rotor
- Air-gap length between rotor and stator
- Stack length of the machine
- Number of turns per coil
- Slot fill factor and maximum current density in stator slot

Next, a sensitivity analysis is performed to identify the 10 most significant variables shown in Figure 6.12. Out of the 10 variables, six define magnet shape and placement in the rotor, while four control slot size and shape. Variable ranges are defined to be within 20% variation from the reference design. All variable values are limited to have only two decimal places because of manufacturing accuracy limitations. Variable values for the reference design ($x^{(\text{ref})}$) along with lower ($x^{(L)}$) and upper bounds ($x^{(U)}$) are given in Table 6.4. A total of 10 geometric constraints are considered for the optimization problem. The geometric constraints allow a quick check of a design for geometric feasibility without running the FEA simulation. The constraints are defined by having domain knowledge in mind, which is vital for a good torque profile and a typical manufacturing tolerance of ± 0.05 mm.

6.2.2.3 Selection of Operating Point for Optimization

The performance of an IPM machine directly depends on the speed and torque requirements. Figure 6.13 shows the efficiency contour map of the 2010 Prius motor with a dc-link voltage of 650 V [2]. The peak torque rating of the machine stays constant until a particular speed called the base speed of the machine, which is also very close to the rated speed. PMSMs are famous for their torque density, and therefore, their operation at rated speed is of high importance. To increase

Table 6.4: Values of geometric variables used for optimization.

x_d	Variable	Unit	$x^{(\text{ref})}$	$x^{(L)}$	$x^{(U)}$
x_1	Height of rotor pole cap	mm	9.56	7.65	11.47
x_2	Magnet thickness	mm	7.16	5.73	8.59
x_3	Magnet width	mm	17.88	14.30	21.46
x_4	Angle between magnets	degree	145.35	116.28	174.42
x_5	Bridge height	mm	1.99	1.59	2.39
x_6	Q-axis width	mm	13.9	11.12	16.68
x_7	Slot height	mm	30.9	24.72	37.08
x_8	Slot width	mm	6.69	5.35	8.03
x_9	Height of slot opening	mm	1.22	0.98	1.46
x_{10}	Width of slot opening	mm	1.88	1.50	2.26

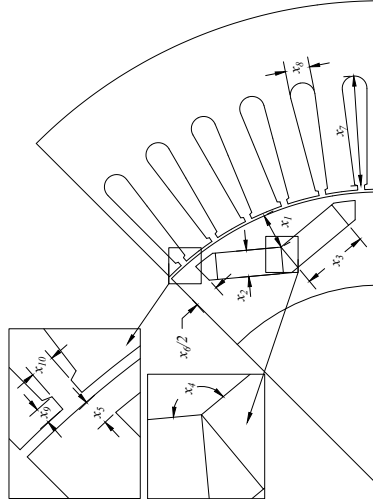


Figure 6.12: Geometric variables used for optimization.

the efficiency of PMSMs in the speed region up to the base speed, they are operated in a way to minimize the excitation current fed to stator copper windings and, therefore Joule losses, while meeting the torque requirements. This specific mode of operation is called maximum torque per ampere (MTPA) operation. For the optimization problem formulation, the rotational speed of the rotor and the excitation angle is kept equal to the values of the reference design at rated MTPA operation. Moreover, it is worth noting that the excitation current is not kept constant. The slot fill factor denotes the proportion of the slot area filled by copper windings. As the slot fill factor is kept constant in contrast to the slot cross-section, it results in different current ratings for different designs.

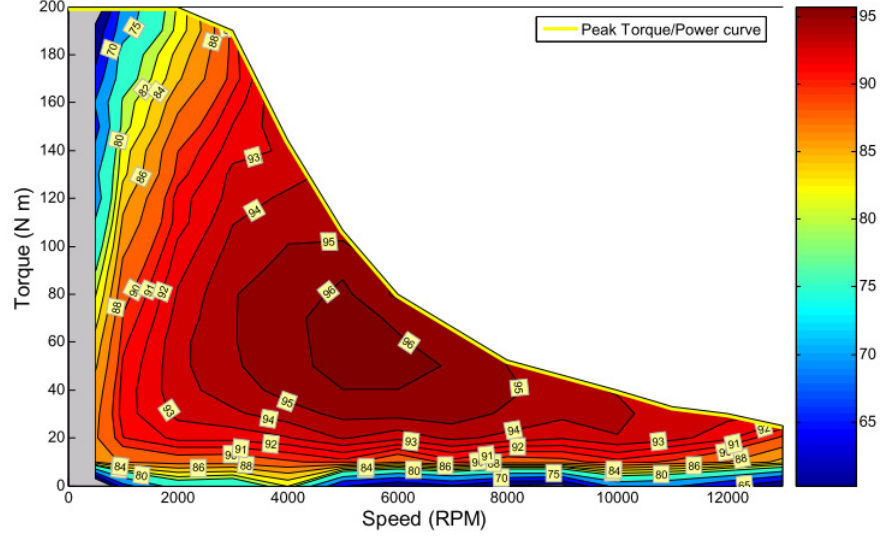


Figure 6.13: 2010 Prius motor efficiency contours for 650 Vdc [2].

Mathematically, the 10-variable ($D = 10$) MOP is now defined as:

$$\begin{aligned}
 &\text{Maximize } \text{Average Torque}(\mathbf{x}), \\
 &\text{Minimize } \text{Torque Pulsation}(\mathbf{x}), \\
 &\text{subject to } g_j(\mathbf{x}) \leq 0, \quad \forall j \in (1, \dots, 10), \\
 &\quad \quad \quad x_d^{(L)} \leq x_d \leq x_d^{(U)}, \quad \forall d \in (1, \dots, 10), \\
 &\text{where } \mathbf{x} \in \mathbb{R}^D,
 \end{aligned} \tag{6.7}$$

where $g_j(\mathbf{x})$ represent the geometrical constraints, \mathbf{x} the variables to optimize, and $x_d^{(L)}$ and $x_d^{(U)}$ the lower and upper bound of the d -th variable respectively. All variables are restricted to have a precision of two decimal places due to manufacturing accuracy limits. Both objective functions, $\text{Average Torque}(\mathbf{x})$ and $\text{Torque Pulsation}(\mathbf{x})$, are based on the result of a 2D transient electromagnetic analysis.

6.2.3 Methodology

When solving real-world optimization problems, the definition of the problem itself can be challenging, and numerous design decisions have to be made. A careful observation of the optimization problem in the previous section reveals that during optimization, along with two objectives, 10

geometric constraints need to be dealt with as well. In particular, while objective functions are expensive to compute, geometric constraints are relatively inexpensive and are computed using mathematical expressions. A preliminary study [282] showed that the computational inexpensive-ness of constraint evaluations could be exploited through a repair operator. The goal of the repair operator is to convert an infeasible solution to a feasible one that satisfies all constraints. Additionally, design optimization of electric machines is an expensive problem to solve and an effort must be made to reduce the computational cost. The evaluation of 1000 design solutions using a 2D transient magnetic study in Flux-2D on a single core of a PC-based workstation can take about 14 hours even with only $1/8^{th}$ periodic model. In order to maintain the statistical significance of results, an optimization run has to be repeated several times, making the whole procedure extremely expensive. Therefore, this study also presents a strategy for the incorporation of surrogates into the proposed optimization algorithm along with a repair operator.

In this study, the well-known evolutionary multi-objective optimization (MOO) algorithm NSGA-II [10] is used as the base optimization algorithm. NSGA-II is a modular, parameter-less optimization algorithm well suited for bi-objective optimization problems, including optimization of electric machines. NSGA-II starts with a population of random solutions called the parent population. After evaluating the population members, pair-wise comparisons are made to select non-dominated and less-crowded solutions [9] in order to meet the main goals of multi-objective optimization. The selected population members are then recombined and mutated to create an offspring population of the same size as the parent population. After their evaluation, the offspring population is merged with the parent population to execute a final survival selection to pick the top half of the population. The selected population becomes the parent population of the next generation. This process is continued until a termination criterion is satisfied. The incorporation of a repair operator and surrogate assistance is explained below.

6.2.3.1 Repair Operator

The repair operator implementation consists of *two* phases: first, the geometric constraints are satisfied by using an embedded but relatively simple optimization procedure; second, the precision of two decimals is satisfied by rounding each variable up or down, preserving the satisfaction of the geometric constraints for an easy implementation purpose. Thus, the repair operator finally returns a feasible solution considering all the electric engine design problem specifications. The proposed repair ensures that a solution is feasible before its evaluation of objectives. In order to ensure feasibility, the constraint functions are called more frequently than the objective functions, exploiting the inexpensiveness of constraints. Incorporating this repair operator into an evolutionary algorithm is relatively straightforward and yet an effective strategy of adapting an existing optimization method to the needs of a real-world optimization problem.

6.2.3.2 Surrogate Incorporation

Commonly, surrogates – approximation or interpolation models – are utilized during optimization to improve the convergence behavior. First, one shall distinguish between two different types of evaluations: ESEs that require to run the computationally expensive evaluation; and ASEs which is a computationally inexpensive approximation by the surrogate. Where the overall optimization run is limited by ESE^{\max} function evaluation, function calls of ASEs are only considered as algorithmic overhead. In order to improve the convergence of NSGA-II, the surrogates provide ASEs and let the algorithm look several iterations into the future without any evaluation of ESEs. The surrogate models are used to create a set of infill solutions as follows: First, NSGA-II is run for k more iterations (starting from the best solutions found so far), returning the solution set $^{(cand)}$. The number of solutions in $^{(cand)}$ corresponds to the population size of the algorithm fixed to 100 solutions in this study. After eliminating duplicates in $^{(cand)}$, the number of solutions N desired to run using ESEs needs to be selected. The selection first creates N clusters (in the objective space based on $\mathbf{F}^{(cand)}$) using the k-means algorithm and then uses a roulette wheel selection based on the predicted crowding distances. Note that this will introduce a bias towards boundary points as

they have been depicted with a crowding distance of infinity. Altogether, this results in N solutions to be then evaluated using ESEs in this optimization cycle.

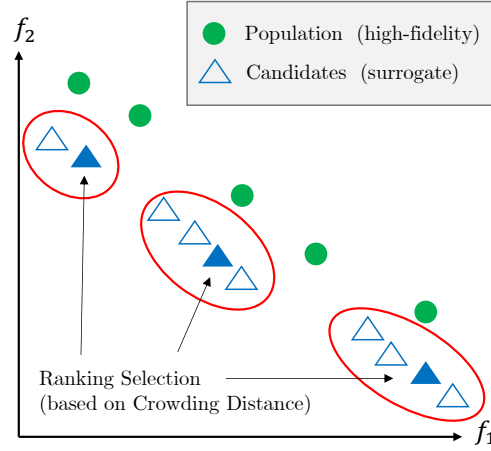


Figure 6.14: Ranking selection of solutions obtained by optimizing the surrogate-based optimization problem.

A few more words shall be said about the surrogate itself. Since the electric machine design is formulated with two objectives to be optimized, two different models are built. Separately fitting a model for each objective corresponds to the *MI* method proposed in the surrogate usage taxonomy [23]. For each objective, the best model type is found by iterating over different model realizations of RBF [34] and Kriging [35] varying normalization, regression, and kernel type. Finally, the best model type is chosen based on the validation set's performance.

6.2.3.3 NSGA-II-WR-SA

In Algorithm 6.1, a detailed pseudo-code demonstrating the solution repair and surrogate usage is provided. The algorithm's parameters are the expensive objective functions $f(\mathbf{x})$ and the inexpensive constraint functions $g(\mathbf{x})$; the maximum number of exact solution evaluations ESE^{\max} serves as an overall termination criterion; the number of the initial design of experiments N^{DOE} describes how many designs are evaluated before optimization starts; the number of solutions N evaluated in each optimization cycle; and the number of surrogate optimization cycles k , or in other words for how many iterations the surrogate are used to look into the future.

Algorithm 6.1: NSGA-II-WR-SA: NSGA-II with Repair and Surrogate Assistance.

Input : Expensive Objective Function $f(\mathbf{x})$, Inexpensive Constraint Function $g(\mathbf{x})$, Maximum Number of Exact Solution Evaluations ESE^{\max} , Number of Design of Experiments N^{DOE} , Number of ESEs in each Iteration N , Number of Surrogate Optimization Cycles k

```
/* initialize feas. solutions using the inexpensive function g */
1  $\leftarrow \text{constrained\_sampling}(N^{\text{DOE}}, g)$ 
2  $\mathbf{F} \leftarrow f()$ 
3 while  $|| < ESE^{\max}$  do
    /* exploitation using the surrogate */
4      $\hat{f} \leftarrow \text{fit\_surrogate}(\mathbf{F})$ 
5      $(^{(\text{cand})}, \mathbf{F}^{(\text{cand})}) \leftarrow \text{optimize}(\text{'NSGA-II-WR'}, \hat{f}, g, \mathbf{F}, k)$ 
6      $(^{(\text{cand})}, \mathbf{F}^{(\text{cand})}) \leftarrow \text{eliminate\_duplicates}(^{(\text{cand})}, \mathbf{F}^{(\text{cand})})$ 
7      $C \leftarrow \text{cluster}(\text{'k\_means'}, N^{(\text{exploit})}, \mathbf{F}^{(\text{cand})})$ 
8      $(^{\text{surrogate}}) \leftarrow \text{ranking\_selection}(^{(\text{cand})}, C, \text{crowding}(\mathbf{F}^{(\text{cand})}))$ 
    /* evaluate and merge to the archive */
9      $\mathbf{F}^{(\text{surrogate})} \leftarrow f(^{\text{surrogate}});$ 
10     $\leftarrow \cup(^{\text{surrogate}})$ 
11     $\mathbf{F} \leftarrow \mathbf{F} \cup \mathbf{F}^{(\text{surrogate})}$ 
12 end
```

First, the algorithm starts by sampling N^{DOE} solutions in the feasible space using a sampling strategy producing only feasible solutions (for more details, we refer to [282]) and evaluates the solution set (Line 1 and 2). Then, while the overall evaluation budget ESE^{\max} has not been used yet, surrogates \hat{f} are built for the objectives (Line 4). By applying NSGA-II for k optimization cycles starting from X using the surrogate models $\hat{f}(x)$ and the inexpensive objective functions $g(x)$, a candidate set of solutions $^{(\text{cand})}$ and $\mathbf{F}^{(\text{cand})}$ is retrieved (Line 5). Depending on the surrogate problem, some solutions in $^{(\text{cand})}$ can be identical to the ones already evaluated in X ; thus, a duplicate elimination ensures these solutions are filtered out (Line 6). Since the size of $^{(\text{cand})}$ exceeds N , a subset selection based on the predicted crowding distances takes place (Line 7 and 8). Finally, the resulting solution set $^{\text{surrogate}}$ of size N is evaluated using ESEs and is appended to the archive of solutions.

6.2.4 Results and Discussion

In the following, the performance contribution of each of the components in the proposed method shall be examined. This includes answering the following key questions:

- Is the repair operator helpful during optimization, and what is its impact?
- Does the usage of surrogates improve the convergence behavior?
- What insights are gained from the Pareto-optimal designs, and what can we learn from them for the electric machine design?

The first two questions are related to the optimization method itself and its convergence. The latter is vital as it addresses the ultimate goal of optimization, which is gaining more insights and finally choosing an electric machine design.

6.2.4.1 Analysis of Constraints

Before analyzing the impact of the repair operator, the constraints shall be investigated. As an initial study, 100 random solutions are sampled using the Latin hypercube sampling method. In order to ensure statistical significance, each experiment is repeated 100 times. Results show that 69.7% of these randomly sampled solutions are infeasible, or in other words, only 30.3% are feasible. A solution is considered infeasible if one or more constraints are violated. An analysis of each constraint separately provides more information on what type of constraints are more difficult to satisfy. In Table 6.5, the percentage of infeasible solutions is shown for each constraint. The percentages reveal that some constraints are more difficult to satisfy than others. The constraints g_8 and g_9 , which control the slot shape, and g_1 , which controls the magnet placement close to the shaft, have not been violated in any of the 10,000 solutions. In contrast, some other constraints, such as g_2 and g_7 , related to magnet placement close to rotor OD, are responsible for infeasible solutions 31.43% and 36.28% of the time. It is worth mentioning that these 10,000 solutions are

Table 6.5: The constraint violation of each constraint value from g_1 to g_{10} .

Value	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
Infeas.	0.0%	31.43%	19.16%	19.16%	21.94%	19.59%	36.28%	0.0%	0.0%	20.25%
Rank	7	2	6	6	3	5	1	7	7	4

generated from the ranges of the variables defined in Table 6.4. For a different search space with arbitrarily defined variable bounds, the percentage of infeasible solutions could be even larger.

6.2.4.2 Impact of Repair Operator

In order to investigate the impact of the proposed repair operator, experiments with two optimization methods are conducted: (1) NSGA-II and (2) NSGA-II-WR, which refers to NSGA-II with the proposed repair operator. Both approaches use the binary tournament selection, simulated binary crossover (SBX) operator with a probability of 0.9, and polynomial mutation. The distribution index used for crossover and mutation operators are $\eta_c = 15$ and $\eta_m = 20$, respectively. For both methods, a population size of 100, 20 offsprings in each generation, and 1500 function evaluations in total are chosen for one optimization run. Five such optimization runs are completed for each method to maintain statistical significance, making the total number of evaluations 7500. The overall setup and results of all experiments are shown in Table 6.6 and Figure 6.15.

First of all, one can note that both algorithms have successfully converged to a set of Pareto-optimal solutions showing trade-offs between average torque and torque pulsation. Second, NSGA-II-WR outperforms NSGA-II, which demonstrates the positive impact of the repair operator. The use of the repair operator obtains more non-dominated solutions, which also have a larger hypervolume value. Moreover, it should be noted that for the calculation of hypervolume, the worst and the best points are found from the combined set of the two Pareto-optimal fronts. Thereafter, the objective functions are normalized to obtain the normalized hypervolume, as shown in Table 6.6. Now, one might argue why this experiment was necessary and how a repair operator could have harmed the convergence in the first place? One possible risk of adding a deterministic repair of infeasible solutions is a diversity loss because the natural exploration of an evolutionary algorithm has been

Table 6.6: Optimization setup and results. Evaluations (Evals) correspond to total functional evaluations performed in five runs. The reported hypervolume is calculated after normalization of objective functions.

Algorithm	Description	Evals	Feasible	Non-dominated	Hypervolume
NSGA-II	Conventional	7,500	5,446	27	0.7206
NSGA-II-WR	With Repair	7,500	7,500	59	0.7382

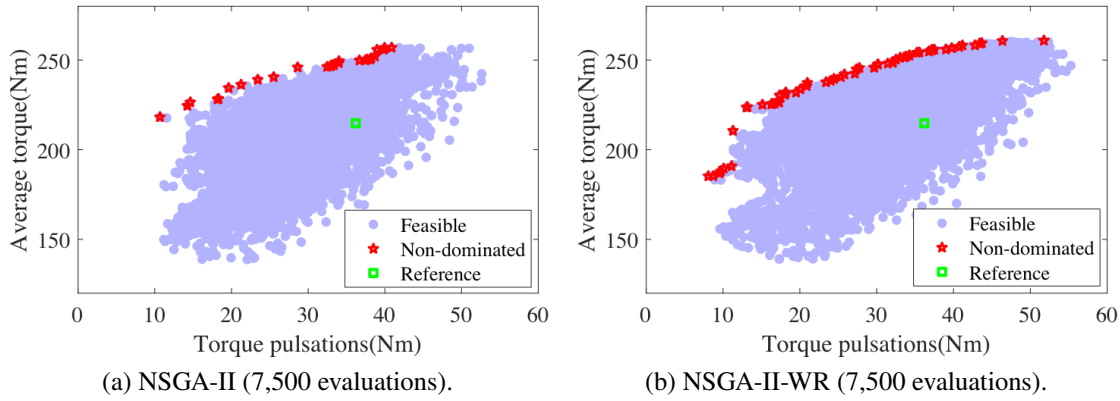


Figure 6.15: Objective space illustrating dominated and non-dominated solutions for optimization runs completed using NSGA-II and NSGA-II-WR.

interfered with. Thus, influencing the exploration by adding a repair can also have a negative impact. However, results indicate that the proposed customization is well-suited for optimizing the design of an IPM machine and increases the diversity of obtained solutions on the Pareto-optimal front.

6.2.4.3 Parameter Study for Surrogate-Assisted Optimization

In this section, the following three hyperparameters related to the surrogate-assistance are varied to analyze the performance of the proposed optimization method.

- N : Number of ESEs in each iteration
- k : Number of generations for exploitation using ASEs
- N^{DOE} : Number of initial design of experiments

To carefully analyze the impact of the above parameters, three optimization Setups, A, B, and C, are defined in a sequential manner. Each setup consists of four cases with variations applied to only one hyperparameter while keeping the other two constant. Each case is repeated five times with 200 functional evaluations in each run, making the total number of evaluations equal to 1,000. To make fair comparisons, the same set of N^{DOE} are used for all runs defined in Setup A and B, while the same seed is used to generate the initial population for all runs defined in Setup C. The complete setup for this study is shown in Table 6.7. Clearly, Setup A, B, and C quantify the impact of N , k , and N^{DOE} , respectively. In order to compare the performance of surrogates in different cases, three criteria are selected, (1) the number of non-dominated solutions (N_{nds}), (2) Hypervolume (HV), and (3) the rate of change of HV with evaluations ($RHVE$). Since the study aims to find a suitable hyperparameter setting in a sequential manner, results of Setup A are used to define Setup B, and combined results of Setups A and B are used to define Setup C. The calculation of the three criteria is explained below.

- N_{nds} : All five runs of each case are combined to obtain one non-dominated (Pareto) front, yielding the number of non-dominated solutions.
- HV: The best and the worst objective function values are found from the Pareto-optimal sets of the cases being analyzed, and objective functions values are normalized to calculate the corresponding HV.
- $RHVE$: Five runs of each case provide five arrays of $RHVE$. Then the median of these five arrays is used to obtain the final $RHVE$ for the corresponding case.

Results for all the cases of three setups are shown in Table 6.8 and Figure 6.16. The observations from this study are listed below.

- Increasing the number of N , initially improves the non-dominated front. However, a large value of N may lead to an over-early convergence with a biased search, as can be seen for $N = 20$ and $N = 25$. A possible way to overcome this could be to increase the total number

Table 6.7: Complete setup for analyzing impact of hyperparameters on performance of surrogate-assisted optimization. For all cases, number of functional evaluations is limited to 200 in a single run. Each case is repeated 5 times, thus, making total evaluations 1,000 for each case.

Case	Runs	Setup A			Setup B			Setup C		
		N	k	N^{DOE}	N	k	N^{DOE}	N	k	N^{DOE}
1	5	5	25	100	10	10	100	10	35	60
2	5	10	25	100	10	20	100	10	35	80
3	5	20	25	100	10	25	100	10	35	100
4	5	25	25	100	10	35	100	10	35	120

Table 6.8: Results for Setups A, B, and C defined for analyzing impact of hyperparameters on performance of surrogates. Hypervolume (HV) is calculated after normalization of objective functions.

Case	Setup A		Setup B		Setup C	
	N_{nds}	HV	N_{nds}	HV	N_{nds}	HV
1	42	0.8051	32	0.8062	47	0.8650
2	40	0.8304	33	0.7851	51	0.8269
3	56	0.7711	40	0.8304	43	0.8211
4	30	0.7475	43	0.8397	38	0.7983

of evaluations. Nevertheless, that would also result in an unwanted increase in the associated computational cost.

- Increasing the value of parameter k leads to a better Pareto-optimal front. This makes sense since more generations for exploitation means more ASEs before surrogates produce infill solutions.
- Increasing the value of N^{DOE} results in smaller HV. For a smaller value of N^{DOE} , the surrogate has more ESEs to improve the model fit and generate better offsprings in future generations. This means running the optimization cycle with more ESEs could lead to an improvement in the quality of the Pareto-optimal set for a larger value of N^{DOE} . However, this will also increase the computational cost of optimization, which is again undesirable.

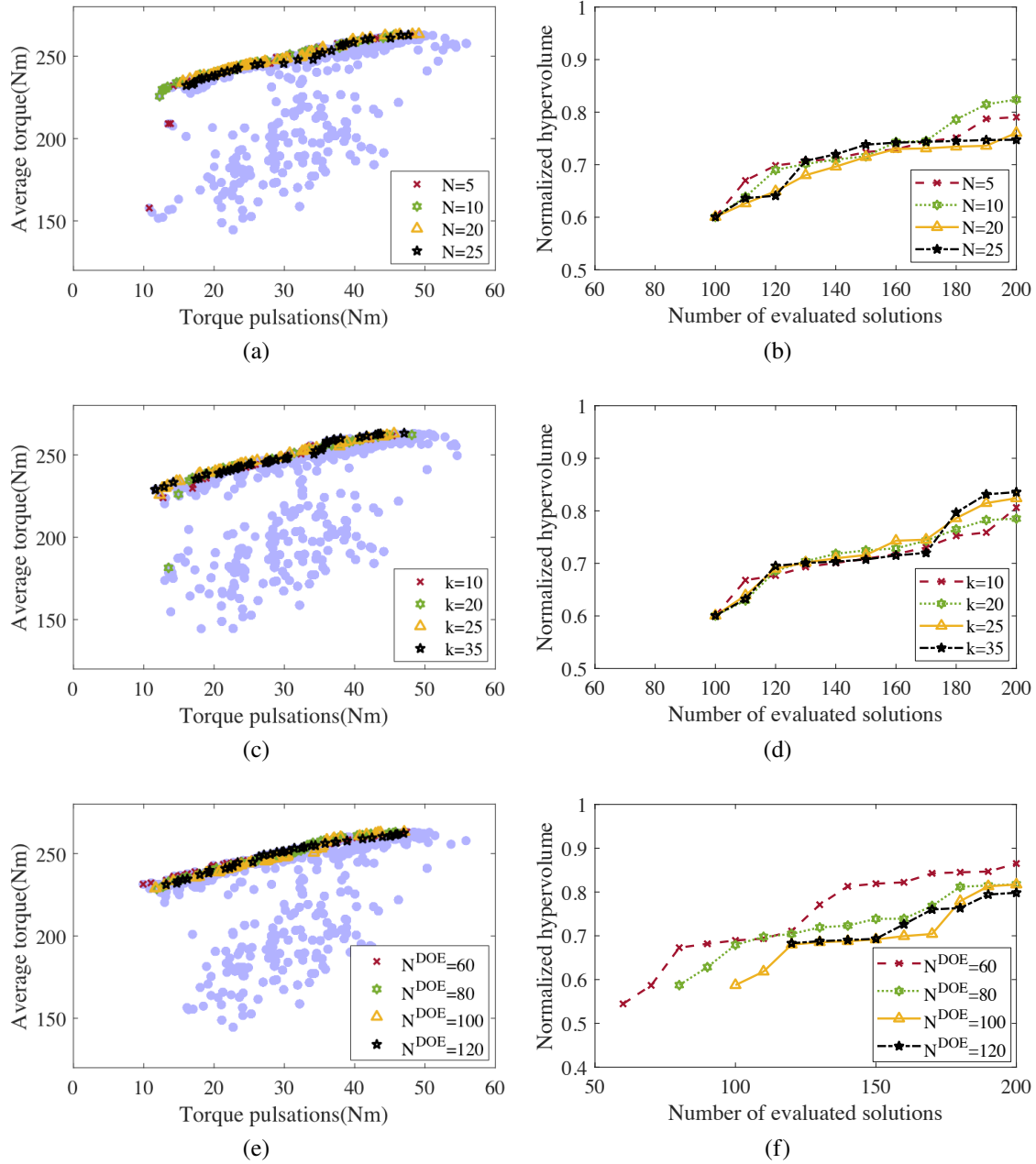


Figure 6.16: Objective space illustrating Pareto-optimal fronts for cases of Setups A, B, and C.

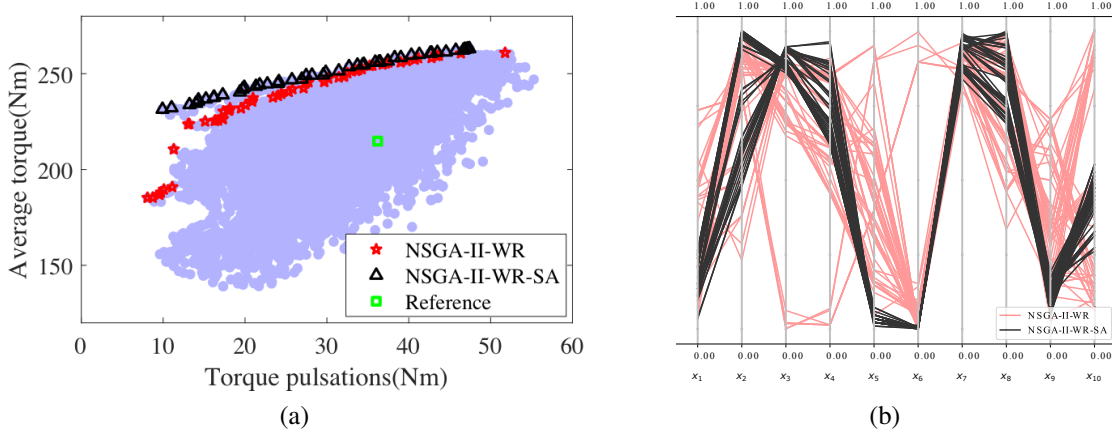


Figure 6.17: Comparison of objective space with Pareto-optimal fronts and normalized design space of Pareto optimal sets. Pareto-optimal sets are obtained from two optimization methods, NSGA-II-WR and NSGA-II-WR-SA.

6.2.4.4 Convergence Analysis with and without Surrogates

Based on the hyperparameter study, Case 1 from Setup C, with $N = 10$, $k = 35$, and $N^{\text{DOE}} = 60$, is identified as the best setting for surrogate-assisted optimization and its results are compared with those obtained from NSGA-II-WR. For the remainder of this section, we refer to the best hyperparameter configuration found as NSGA-II-WR-SA.

Figure 6.17a shows a comparison of the two Pareto-optimal sets obtained by NSGA-II-WR and NSGA-II-WR-SA. One can observe that NSGA-II-WR-SA clearly outperforms NSGA-II-WR as the Pareto-optimal front obtained with the former method dominates most of the Pareto-optimal front obtained with the latter. To understand the convergence of each optimization method, the design space of the two Pareto-optimal sets is plotted using a parallel coordinates plot (PCP) as shown in Figure 6.17b. Each vertical axis in the PCP plot represents the normalized optimization variable x_d with its lower and upper bounds as 0 and 1, respectively, and each horizontal line represents a solution. The design space of the two Pareto-optimal sets shows that almost all the variables have converged to an optimal value with NSGA-II-WR-SA, whereas, with NSGA-II-WR, some of the variables still have significant variations with some further scope of convergence. These observations validate the incorporation of surrogates in the proposed optimization method,

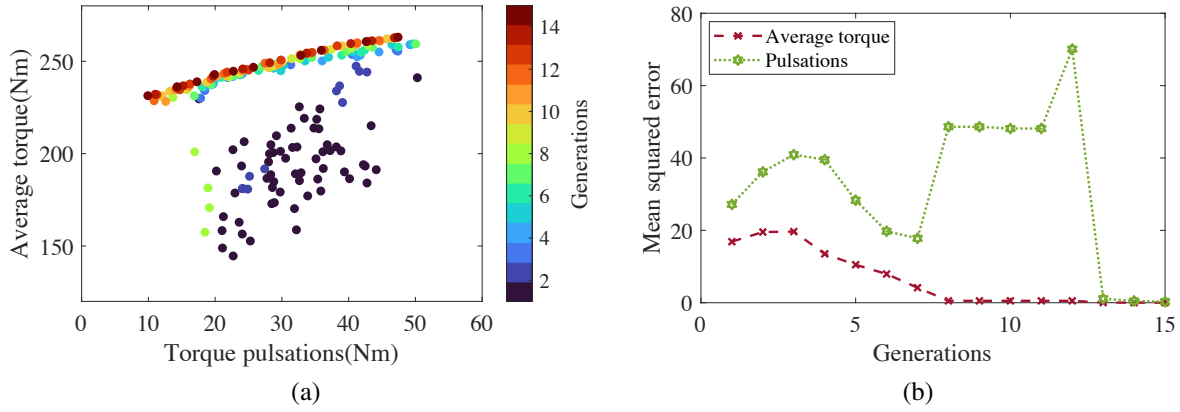


Figure 6.18: Exploration of objective space and MSE in prediction of objective functions, for each generation using NSGA-II-WR-SA.

demonstrating the improvement of the algorithm's convergence.

It should be noted that while NSGA-II-WR uses 7,500 ESEs, the evaluation budget for NSGA-II-WR-SA is limited to only 1,000. However, there is more to optimization with surrogates than just the number of function evaluations. As explained in the previous section, surrogates look into the future for k iterations, 35 in the case of NSGA-II-WR-SA, before producing N number of infill solutions for the next generation. In each iteration, surrogates provide ASEs, of size equal to 100 in this study, effectively evaluating 3,500 solutions before generating infills. This way, the effective number of evaluations using surrogates exceeds those without surrogates, which leads to better convergence. Further insights into the convergence can be obtained by analyzing how the surrogates explore the objective space and how close the predicted objective values are to the actual expensive ones. Figure 6.18 shows the objective space and the mean squared error (MSE) of predictions in each generation using NSGA-II-WR-SA (in one single run). An important observation is that surrogates can predict average torque with higher accuracy than torque pulsations, which are greatly influenced by the electrical steel's non-linearity (magnetic saturation). The sudden rise in MSE of pulsations from generation 7 to 8 can be explained by analyzing the corresponding solutions in objective space. Due to a sudden increase in average torque, more and more solutions are generated with higher slot cross-section and magnet volume, which results in an operation in a high saturation region. Consequently, it takes some time before the surrogates can predict the pulsations accurately.

6.2.4.5 Analysis of Pareto-optimal Solutions

After discussing the optimization procedure in detail, Pareto-optimal solutions shall be analyzed in order to gain insight into the electric machine design. For this purpose, the design space of the Pareto-optimal set obtained by NSGA-II-WR-SA is analyzed, as shown in Figure 6.17b. Since the size of the machine is kept constant, magnet width (x_3), slot height (x_7), and slot width (x_8) converge to the higher end of variable ranges for most solutions. While a larger magnet width increases magnet flux linkage leading to higher average torque, it also reduces the q-axis width (x_6), which has converged to the lower end of the variable range. Similarly, an increase in slot cross-section area results in more space for winding, which directly translates to higher allowable excitation current and an increase in average torque. Additionally, a reduction in bridge height (x_5) directly increases the air-gap flux density, which increases average torque.

On the other hand, torque pulsations are affected by the magnet pole arc and material saturation. Magnet pole arc is directly proportional to magnet width (x_3) and angle between the magnets (x_4). Material saturation is a nonlinear behavior observed in magnetic materials, such as electrical steel, introducing saturation harmonics in magnetic flux density. While a larger slot cross-section increases average torque by means of more excitation current, it also increases magnetic material saturation, leading to more torque pulsations. Lastly, the height and width of slot opening, x_9 and x_{10} respectively, which are responsible for slot harmonics, have converged to the lower end of the variable range. This makes sense as semi-closed slots are used as an effective method to reduce slot harmonics contributing to torque-pulsations [320].

6.2.4.6 Selection of Preferred Solutions

The selection of an electric machine design is primarily application-dependent. A popular approach uses a scalarized function for optimization, which yields a single optimal solution at the end of an optimization run. However, selecting weights for a scalarized function is rather difficult. Scalarization also prevents the possibility of analyzing trade-offs offered by Pareto-optimal solutions. In this study, two different approaches are used to select the preferred solutions; (1) a domain-specific

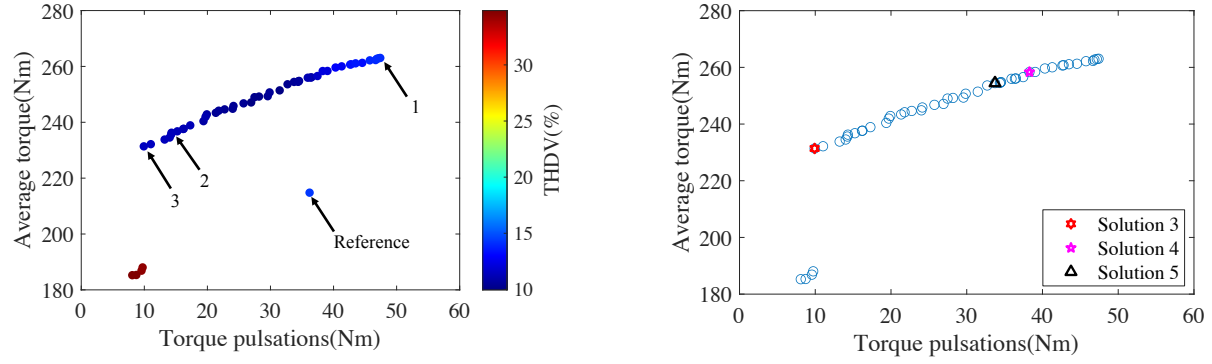
a posteriori multi-criteria decision-making (MCDM) method which involves machine expertise, and (2) trade-off analysis of the Pareto-optimal set to identify and choose the solutions with the highest trade-off. Pareto-optimal solutions obtained from combined runs of NSGA-II-WR and NSGA-II-WR-SA optimization methods are used in both approaches.

Domain Specific A Posteriori MCDM Method: For domain-specific a posteriori MCDM method, the following performance measures, important to all PMSMs, apart from the two objective functions defined in Equation 6.7, are used to select preferred solutions from the Pareto-optimal set.

- Total harmonic distortion of noload back emf ($THDV$)
- Peak of fundamental of back emf ($F-BEMF$)
- Magnet utilization factor (MUF)

For electric machine design, a low value of $THDV$ is desired since it is a direct measure of noise, vibration, and harshness (NVH) during the operation of the electric machine. Similarly, an increase in $F-BEMF$ increases the average torque but reduces the maximum speed that the machine can achieve, thus, presenting a trade-off. On the other hand, a higher value of MUF is desired, where MUF is defined as the ratio of average torque to PM volume. Since PM material is expensive, a higher value of MUF translates to a reduction in the machine cost. Figure 6.19a shows $THDV$ for Pareto-optimal solutions obtained from combined runs of NSGA-II-WR and NSGA-II-WR-SA optimization methods. Although solutions lying in the bottom region of the Pareto-front have the least torque pulsation, they have the highest $THDV$ (more than 30%) and must be avoided during selection. Since the remaining Pareto-optimal solutions have similar $THDV$ (10-14%), it is easier to select solutions based on the other four performance measures. Based on further evaluation, three preferred solutions, 1, 2, and 3, are selected, which are also highlighted in Figure 6.19a. The basis of the selection of these solutions is as follows.

- Solution 1: maximum average torque



(a) Selected solutions with domain specific a posteriori MCDM method.

(b) Selected solutions with trade-off analysis.

Figure 6.19: Objective space highlighting the selected solutions using the a posteriori MCDM method.

- Solution 2: maximum MUF
- Solution 3: minimum pulsation and $F-BEMF$

Trade-Off Calculation Using Objective Functions: Trade-off analysis of the Pareto-optimal set is an effective method to select preferred solutions without domain expertise. In this study, for a particular solution ($\mathbf{x}^{(i)}$), a trade-off is calculated using the following equation on the neighborhood of points ranked according to Euclidean distance (represented by $B(\mathbf{x}^{(i)})$). The term $\sum_{k=1}^M \{1|c_k > d_k\}$ calculates the number of k 's (out of M) for which the condition $c_k > d_k$ is valid. It should be noted that for the trade-off calculation, only two objective functions defined in (6.7) are used, and solutions with high trade-off values are desired.

$$\begin{aligned}
 \text{Avg.Loss}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) &= \frac{\sum_{k=1}^M \max(0, f_k(\mathbf{x}^{(j)}) - f_k(\mathbf{x}^{(i)}))}{\sum_{k=1}^M \{1|f_k(\mathbf{x}^{(j)}) > f_k(\mathbf{x}^{(i)})\}}, \\
 \text{Avg.Gain}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) &= \frac{\sum_{k=1}^M \max(0, f_k(\mathbf{x}^{(i)}) - f_k(\mathbf{x}^{(j)}))}{\sum_{k=1}^M \{1|f_k(\mathbf{x}^{(i)}) > f_k(\mathbf{x}^{(j)})\}}, \\
 \text{Trade-off}(\mathbf{x}^{(i)}) &= \max_{j=1}^{|B(\mathbf{x}^{(i)})|} \frac{\text{Avg.Loss}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})}{\text{Avg.Gain}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})}.
 \end{aligned} \tag{6.8}$$

Table 6.9: Performance comparison of five preferred solutions found using domain specific a posteriori MCDM method and trade-off analysis. Preferred values are highlighted in bold for the five solutions.

Solution	Avg torque (Nm)	Pulsations (Nm)	THDV (%)	MUF (Nm/mm ³)	F-BEMF (V)
1	263.0374	47.4060	14.1263	0.0290	248.2401
2	235.5986	14.2488	11.2982	0.0308	236.7291
3	231.3853	9.9186	11.5016	0.0304	234.4451
4	258.3494	38.2791	12.2894	0.0287	246.4461
5	254.4529	33.7342	10.4722	0.0285	242.1732
Reference	214.7760	36.1846	14.4093	0.0330	209.2622

After performing a trade-off calculation based on the above definition, three solutions with the highest trade-off, Solution 3, 4, and 5, are selected from the combined Pareto-optimal set, as shown in Figure 6.19b. Interestingly, Solution 3 is picked again, with the highest trade-offs among all solutions. The basis of the selection of these solutions is as follows.

- Solution 3: highest trade-off value (114.99)
- Solution 4: 2nd highest trade-off value (50.79)
- Solution 5: 3rd highest trade-off value (35.07)

Performance comparison of selected solutions: Performance details of the five selected solutions along with reference design are given in Table 6.9. Further insights into the performance of these solutions can be gained by analyzing the design space, as shown in Figure 6.20a. Some important observations highlighting the trade-off among selected solutions are as follows.

- Out of the five selected solutions, Solution 4 is dominated in all performance measures by at least one solution.
- Solution 1 provides the maximum average torque but also maximizes the amplitude of pulsations and *F-BEMF*. Both these characteristics can be explained by larger magnet thickness (x_2), slot height (x_7), slot width (x_8), and slot opening height and width (x_9 and x_{10}).

- Solutions 2 and 3 perform quite similarly in all aspects, with slight variations observed in average torque and torque pulsations. This can be explained since both solutions have almost similar values of design variables with only a significant difference in angle between magnets (x_4).
- All selected solutions have larger $F-BEMF$ compared to the reference design, which means that all five solutions will have a smaller speed range. The relation between $F-BEMF$ and the maximum achievable speed can be seen in Figure 6.20b, which shows the torque/speed envelop of solutions 1 and 3 along with reference design. With a further increase in speed, one would observe that torque produced by Solution 1 drops to zero more quickly compared to Solution 3.
- Although Solution 5 has the least $THDV$ and provides high average torque (only 3.26% less than Solution 1), it has the least MUF out of the selected solutions. Additionally, Solution 5 has significantly less pulsation due to smaller slot width (x_8), leading to a smaller slot cross-section compared to Solution 1.
- A comparison of magnetic flux density plots of Solutions 1, 2, and 3 at corresponding rated operating conditions shows that Solution 1 suffers from higher saturation in stator teeth, back iron, and rotor steel close to magnet edges as explained above and is shown in Figure 6.21.

Based on the discussion presented in this work, one should select Solution 1, 4, or 5 for an application with a high average torque requirement. If the focus is more on smooth operation with high-speed range, Solution 2 or 3 should be selected. It is also worth mentioning that while trade-off analysis can pick Solution 3, one would have missed out on Solution 2 with the highest MUF which requires domain expertise. Ultimately, the selection of a single solution out of a Pareto-optimal set will require further preference information.

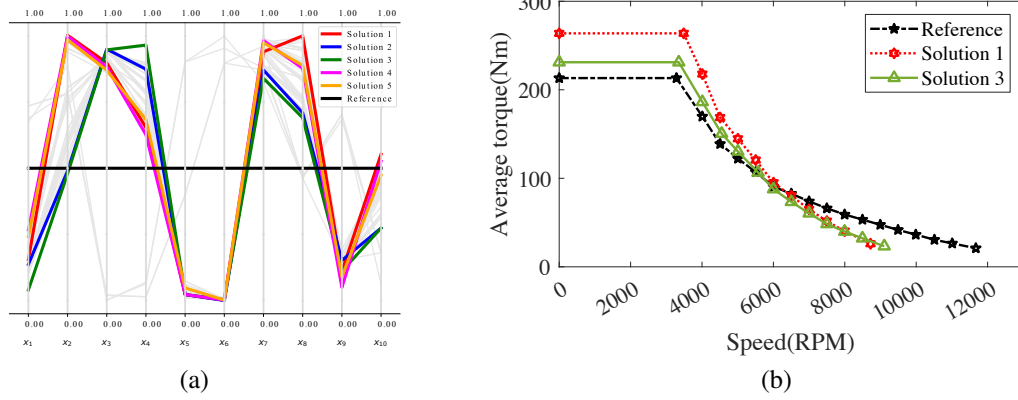


Figure 6.20: Five selected Pareto-optimal solutions highlighted in normalized design space.

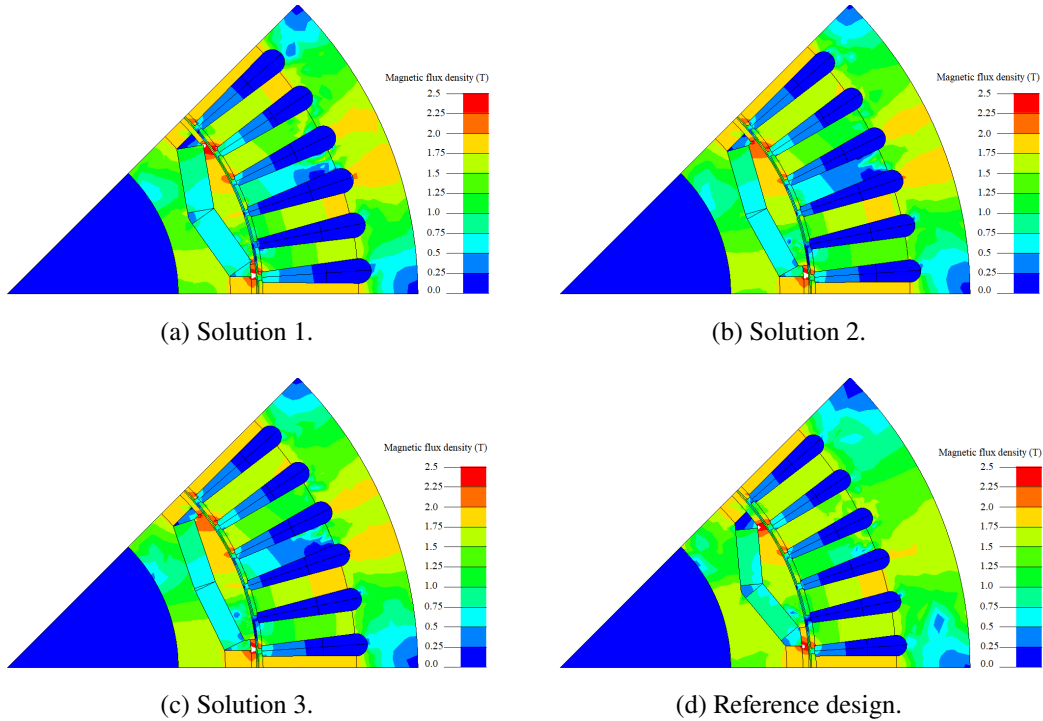


Figure 6.21: Magnetic flux density plots of Solutions 1, 2, 3 and reference design at rated operation.

6.2.5 Summary of Section 6.2

In this section, a repair operator to efficiently handle inexpensive constraints has been presented. The operator's goal is to convert every infeasible solution to a feasible one before the expensive objective function evaluation takes place. The repair operator has been incorporated into NSGA-II, a popular multi-objective optimization algorithm, and its impact has been demonstrated by

solving a bi-objective optimization problem with expensive objectives and inexpensive constraint function evaluations. Results have shown that the optimization with the proposed repair operator (NSGA-II-WR) has led to an improved Pareto-optimal set compared to the baseline optimization method (NSGA-II). Since the optimization of real-world problems is often time-consuming, an extension of the proposed method with surrogate assistance has also been proposed. A sequential parametric study has been performed to identify the optimal values of three parameters; (1) infills in each generation (N), (2) number of generations for exploitation (k), and (3) number of the initial design of experiments (N^{DOE}). Results with a tuned parameter configuration have indicated that optimization with surrogates (NSGA-II-WR-SA) improves the algorithm's convergence further and outperforms NSGA-II-WR significantly. The ultimate goal of an optimization process is to reach an optimal solution that can be implemented successfully. Thus, an a posteriori MCDM approach focused on machine cost, noise, vibration, harshness (NVH), and speed range of the electric machine has been presented to identify three preferred solutions out of the Pareto-optimal set successfully. Additionally, trade-off calculations based on objective functions have also been used to pick three preferred solutions, thereby helping the designer focus on solutions of the most interest.

6.3 Summary of the Chapter

This chapter has presented two case studies of real-world applications with computationally expensive objectives functions. First, we have provided a case study addressing the optimization of a cylinder head water jacket. Results have shown that surrogate incorporation can effectively improve the convergence behavior of optimization algorithms. The second case study has investigated the optimization of the design of an electric machine. The optimization problem was based on computationally expensive objectives but less time-consuming constraints. The proposed method has exploited the discrepancy of expenses by ensuring the feasibility of a design before starting the time-consuming simulation. Both case studies have demonstrated the practical relevance of directly addressing computationally expensive functions in the algorithm design.

CHAPTER 7

OPTIMIZATION IN PRACTICE

The previous chapters have enlightened the optimization of computationally expensive functions from a more technical point of view, focusing on *how* surrogates can be used during optimization. Nevertheless, most applications' essential and challenging aspect is their interdisciplinary character, which has not been addressed yet. For application problems, knowledge and experience in *optimization* and at least one other *domain* is necessary. In practice, this requires collaboration between experts with (preferably) complementary expertise. But the way of carrying out the collaborations and the responsibilities of the domain and optimization experts need to be investigated. One of the optimization expert's tasks is to find a suitable optimization method or develop a prototype. For both purposes, most commonly existing optimization frameworks are either directly used by executing an optimization function or indirectly by importing modules. Thus, as developers of a widely used optimization framework called pymoo, we like to give some insights into its architecture, features, and usage. As collaboration and the usage of frameworks are critical for real-world optimization success, both will be discussed next.

7.1 Collaborative Optimization

7.1.1 Introduction

Interdisciplinarity is a critical component of any applied research nowadays. Multiple branches of knowledge coming together require not only to master each discipline independently but also their intersections. A discipline playing an essential role in various sciences regarding problem-solving tasks is (mathematical) *optimization*. The interdisciplinary character of optimization becomes apparent by studying related literature in various research fields, such as engineering, economics, medicine, and society [172, 174, 200, 204]. While reading different kinds of studies, one will realize that some publications focus on the domain and others more on the optimization method

itself; however, most of the attention is paid to the investigation's outcome and not the collaborative process. Since collaboration is vital for success, this chapter focuses on aspects of collaborative research in the context of optimization, which will be referred to as *collaborative optimization* in the remainder of this study.

Collaborations are essential in an interwoven discipline like optimization, which requires knowledge in optimization itself and one or multiple other domains. Because domain knowledge is the foundation for the algorithm's design, its incorporation requires a fundamental, or even deep, understanding of the domain and the desired method's requirements. Naturally, this demonstrates the need for domain and optimization knowledge, which is realized by initiating a collaboration. The attempt of *separately* solving the domain-specific and optimization-related tasks is likely to fail; however, this is still carried out in practice even today. For instance, such a clear separation of tasks can be realized by having a domain expert formulating the problem statement independently and the optimization expert developing the algorithm from thereon without any further feedback from the domain expert. Even though each task should have a collaborator responsible for taking the lead, communication and agreements are vital for true collaboration and success. Thus, in collaborative optimization, the outcome is more than the sum of its parts, and success is achieved by effectively addressing the fusion of multiple research fields.

This chapter's focus shall lie on the research collaboration in any kind of discipline where optimization is needed and applied. Thus, the different phases of collaboration and all supporting activities are of importance. Furthermore, in this study, collaborative optimization is based on human-human interactions; however, human-machine interaction can be a component of the problem description. Nevertheless, while related works specify collaborative optimization only in the context of multi-disciplinary design optimization [321], this study considers collaboration in a more generic context. Moreover, it is worth mentioning that the term collaborative optimization has also been used to refer to a specific type of algorithms to solve large-scale optimization problems [322, 323], which is also not the focus of this study.

First, we'll discuss work related to different aspects of collaborative optimization. In Sec-

tion 7.1.3, we propose a blueprint for collaborative optimization by describing primary and supporting activities. Illustrative case studies are provided in Section 7.1.4 and conclusions are discussed in Section 7.1.5.

7.1.2 Related Work

Collaboration can be defined as “the situation of two or more people working together to create or achieve the same thing” [324]. Sharing the same goal while working together is essential to understanding the word’s meaning. Another definition emphasizes the existence of conflicting goals, which should be reduced to a common denominator, and the fact that collaboration is more contentious than coordination or cooperation [325]. In the field of optimization, well-studied subjects characterize collaboration by projects and project management, interdisciplinarity, communication, and (applied) research. Even though all of them have to be mastered simultaneously in collaborative optimization, related work considers them independently for now. Later in this study, a more precise definition of collaborative optimization and these aspects’ interactions will be provided.

Projects have been well-studied throughout the literature and are a fundamental part of economics. Techniques to measure the success of a project have especially been of interest. A well-known method for measuring success is the so-called iron triangle, describing success as a trade-off of time, cost, and quality [326]. Whereas most authors agree that the criteria are critical, the model has also been criticized for being too simple. Thus, more sophisticated models have been proposed to measure the success or failure of a project. In general, there is an agreement that for projects in general, measuring success is challenging, not least because of subjective views of stakeholders or the time dependency [327]. During a project, the time is also referred to as the project life cycle, which can be divided into different phases: conceptualization, planning, executing, termination [328]. More modern approaches, however, do not follow the traditional waterfall model; instead, they pursue flexible and iterative project management strategies [329].

Projects with goals regarding more than one discipline have to deal with interdisciplinary chal-

lenges. Interdisciplinary is characterized by a suitable combination of knowledge from different specialties. The purpose of the combination is to exceed the values the sum of all contributions individually [330]. A successful fusion of disciplines requires unifying separate ways of understanding and approaching problems across disciplines [331]. Rooting interdisciplinary research more in society was attempted by promoting work across disciplines on many research universities campuses in the United States in past years. However, the general superiority of interdisciplinary over disciplinary knowledge has also been critically assessed [332].

Collaboration across disciplines has to ensure efficient communication. Unavoidably, communication is a practical discipline and a vital skill for many different sciences [333]. It is a widespread belief that interpersonal and social problems are caused by impaired communication and can be alleviated by good communication [334].

Besides essential aspects of collaboration itself, successful collaborations in optimization are evident by studying literature. Various studies show optimization is almost ubiquitous, for instance, in Agriculture [172], Engineering [174], Medicine [200], or Economics [204]. Different research studies use different kinds of collaborations among different stakeholders. Collaboration is also set up in different ways, for example, in the same laboratory between researchers, across departments and research groups, across research institutes in the same of different countries, or between academia and industry.

7.1.3 SOLVeR: Collaborative Optimization

Collaborative optimization describes a procedure involving at least two stakeholders – a domain-specific and optimization expert – pursuing to solve an optimization problem *interactively*. The domain-specific expert initially provides the problem to be solved with the optimization expert's knowledge and experience. The interaction between both experts is crucial to solve the problem successfully and can occur at different levels of involvement.

Even though collaborations are carried out in different manners and have different challenges, they often have analogous phases and supplemental activities. Thus, collaborative optimization

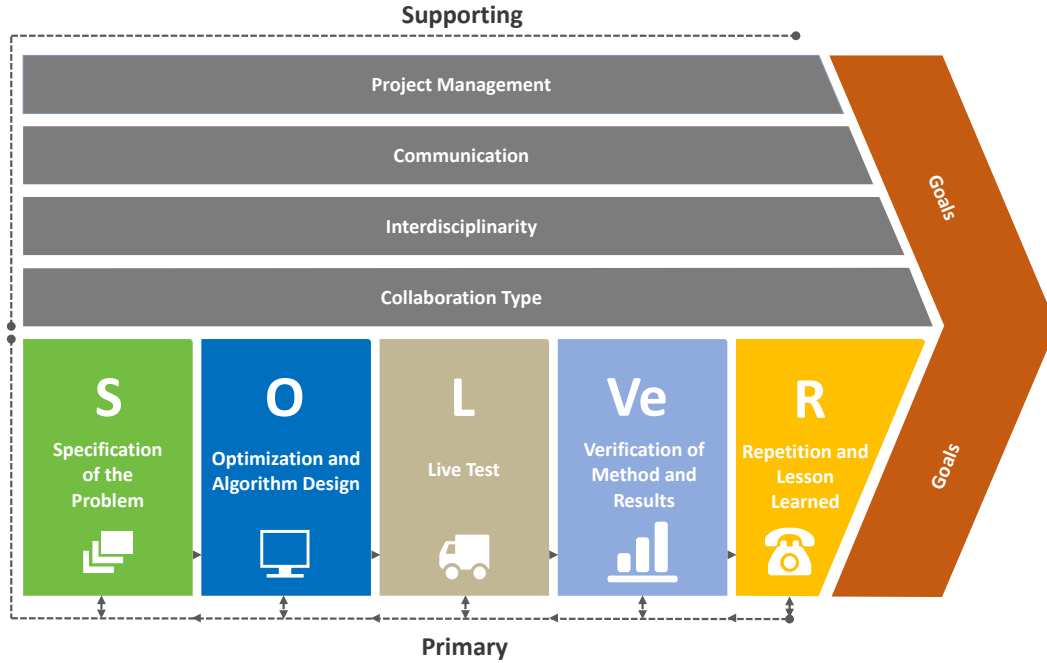


Figure 7.1: Collaborative optimization practice using SOLVeR.

shall be schematized to track the overall progress and highlight important aspects for a successful collaboration. A blueprint for collaborative optimization is shown in Figure 7.1, presenting not only the phases but also the supporting activities. The primary phases follow the *SOLVeR* acronym: Specification of the Problem (‘S’), Optimization and Algorithm Design (‘O’), Live Test (‘L’), Verification of Method and Results (‘Ve’), and Repetitions and Lessons Learned (‘R’). For each phase, the domain and the optimization expert’s roles and responsibilities differ and shall be discussed in detail. Moreover, the arrows between the phases on the bottom indicate that multiple iterations of phases are inevitable in practice and an essential part of a collaboration. Furthermore, the phases are accompanied by supporting activities, such as project management, communication, interdisciplinarity, and the type of collaboration. The blueprint’s split of primary and supporting activities is inspired by the well-known value chain model [335] with similar characteristics. Both the primary and supporting activities are essential to reach the goals. In the following, the five *SOLVeR* phases are discussed, and additionally, an overview of each phase’s characteristics is provided in Figure 7.2. Moreover, all supporting activities are described in detail.

(i) *Specification of the Problem ('S')*: In the first phase, all collaborators need to get a clear understanding of the optimization method's overall goal. For the optimization expert, this often requires understanding the fundamentals of a foreign research field. Thus, the domain expert's responsibility is to communicate efficiently and to define domain-related terminology if necessary. The primary goal is not for all collaborators to understand every little detail but to grasp what the problem is about. Thus, abstraction should be made whenever possible. Moreover, possible requirements and meta-information about the problem should be discussed, for instance, the evaluation time of a single design or the type and number of variables to be considered. After the problem has been defined verbally, it should be stated mathematically, defining the objective(s), constraints, and the underlying search space. With fundamental knowledge about the domain, the optimization expert will often take the lead for the mathematical problem formulation. Nevertheless, the domain expert's feedback is crucial to ensure the formulation fits the specifications and the domain expert's expectations. For instance, a target measure could be either incorporated into the problem formulation as a constraint or an objective. Whereas both options might be legitimate ways of considering this metric, domain knowledge can favor one or the other. Together with the optimization expert's knowledge about each option's benefits and drawbacks in an optimization sense, the domain knowledge demonstrates the benefits of a close collaboration from the beginning.

(ii) *Optimization and Algorithm Design ('O')*: After the problem has been defined mathematically, the design of a suitable algorithm is of most interest. The selection or design of an algorithm requires experience in optimization and can be rather challenging. Before starting with the algorithm's design, all problem-dependent information shall be analyzed. For instance, does the evaluation also provide information about the gradient? Or, how many function evaluations are affordable? However, it's important to note that some characteristics can only be assumed and are not known beforehand. For example, a vital question to ask is the modality of the function's fitness landscape because it determines whether a local or global search might be appropriate. If there is an explicitly defined equality constraint, one of the variables can be replaced in terms of other variables – a process that eliminates one variable, and also every modified solution will

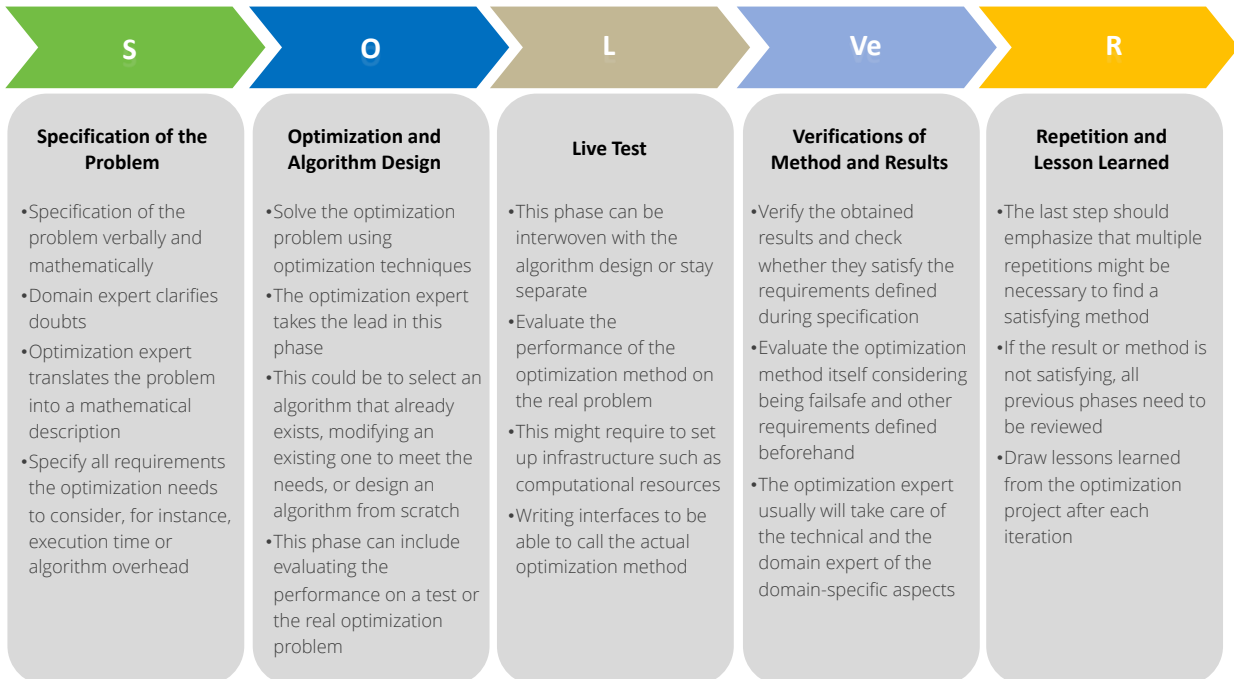


Figure 7.2: Phases and responsibilities.

automatically satisfy the equality constraint. The use of such information to redefine an original problem requires collaboration between optimization and domain-specific experts at the start of the optimization process. A standard optimization algorithm can be modified to suit the supplied problem information. This can happen in modifying different operators of the algorithm. For example, the initial solution(s) can be repaired to satisfy certain constraints so that the search can begin from a good solution(s). The generative operations for creating new solutions can be motivated by the problem information so that new solutions satisfy the supplied problem information.

The fact that the mathematical problem definition and the optimization method are directly linked to each other demonstrates the interdependence of the first two phases and the importance of collaboration. After completing phase two, an algorithm has been developed, possible bugs during development have been fixed, and source code or a binary file for running the method exists.

(iii) *Live Test* ('L'): In the third phase, the developed algorithm is run in a live environment to observe its performance on the real-world optimization problem. The testing phase is crucial to ensure that the algorithm's design is suitable for the original problem. This might require

interfacing between different programming languages or setting up the computational resources to run the method in a live environment. The domain and optimization expert's responsibilities in this phase depend on the type of collaboration and agreements. On the one hand, the algorithm's design can be driven by test problems with similar characteristics as the real-world optimization problem. The development on test problems is also often necessary because of the lack of computational resources or software licenses on the algorithm developer's end or the industrial partner preferring to make the problem not accessible to the outside. On the other hand, the problem's evaluation function might be delivered to the optimization expert – either open or closed source – and may be directly used during the algorithm's design. In some cases, the problem might have been vaguely defined from the beginning, and the developer needs to implement a representative live environment from scratch, for instance, by generating synthetic data with reasonable assumptions. The variety of live tests' realizations shows that different collaboration types require a different amount of collaborative effort in this phase. However, no matter what type has been chosen, this phase's outcome is a method and results that have to be analyzed.

(iv) *Verification of Method and Results ('Ve')*: In the fourth phase, the goals and requirements defined initially need to be critically assessed and verified. The verification is based on the results obtained in the previous phase. Even though the verification procedure will vary from collaboration to collaboration, some tasks employed in practice are to analyze the algorithm's convergence over time and carefully inspect the solutions being found. In some collaborations, the optimization is only performed once, and most attention is paid to the obtained solution(s) itself, and the method plays a minor role. The obtained solutions need to be closely examined and made sure that all requirements are satisfied. The examination often involves checking correctness, feasibility, and the visualization of solutions and results. If discrepancies have been observed, the mathematically defined problem might need to be refined or even entirely redefined, and a reiteration of phase one may be necessary. Other collaborations might focus more on the method itself, primarily when the algorithm is run repetitively, for instance, daily or weekly. Then, a thorough test of the method, including possible boundary scenarios, is of importance. Moreover, for stochastic algorithms, not

a single run's performance, but a statistical analysis of a set of runs needs to be done to address the underlying randomness and ensure the method's robustness. No matter where the priority of the collaboration lies, verification is crucial to measure the overall success.

(v) Repetition and Lesson Learned ('R'): As recommended for projects in general, the last phase consists of reflecting on the collaboration and critically assessing the progress made. Practitioners will agree that no project ends without future work and possible new collaborations. Thus, drawing lessons learned to avoid pitfalls helps improve long-term efficiency and productivity.

Besides primary activities classified into phases, supporting activities are an essential part of collaborative optimization. The supporting activities accompany any of the primary phases and play a different role anytime during the collaboration.

Project Management: A project is characterized by a project schedule with a clearly defined beginning and end. Moreover, the project's outcome is typically defined by milestones and project goals, which should be achieved during or at the end of the project. In practice, goals can also be conflicting, for instance, in a university-industry research collaboration where the researchers prioritize a seminal publication. In contrast, the industry might want to keep the findings confidential to keep a competitor's advantage. Nevertheless, agreeing on the goals initially and keeping track of them is good advice in all collaborations. Moreover, project management includes all matters regarding funding more resources and workforce during the collaboration.

Communication: Efficient communication is essential on many levels. The collaboration is accompanied by communication throughout all phases. The availability of collaborators and the communication frequency can significantly impact the project's outcome. While some collaborators prefer frequent feedback, such as daily or weekly, others favor less frequent meetings, for instance, monthly or biannually. Besides the frequency of regular meetings, the collaboration should define several milestone meetings, consisting of at least a kick-off and final meeting. The type of communication often depends on the geographical distance between collaborators. A relatively small distance and convenient commute shall allow in-person meetings. Often, however,

this is not the case, and mostly online meetings are scheduled. Modern technology that allows to turn on a webcam, share the screen, or even take over screen control can become handy to increase such meetings' productivity. Moreover, consistent e-mail correspondence and a hybrid in-person and online communication style are often carried out in practice. Challenges in communication commonly occur through domain-specific terminology, which is unclear to all collaborators or even language barriers in international collaborations.

Interdisciplinarity: Many collaborations have their origin of a subject being of an interdisciplinary manner. Therefore, an expert for the involved disciplines significantly speeds up the research process or makes meaningful insights possible at all. In collaborative optimization, interdisciplinarity is given by the presence of optimization itself and one other discipline. For some projects, even multiple other disciplines might be involved with possible conflicting objectives. In literature, such a situation related to optimization is also referred to as Multi-Disciplinary Design Optimization (MDO). During the collaboration, especially during the initial problem specification phase, a fundamental understanding of each discipline is essential. Even only rudimentary knowledge helps develop an appreciation for each other's research fields and facilitate meaningful discussions.

Collaboration Type: The type of collaboration has a significant impact on each collaborator's responsibility. With the type of collaboration, we refer to aspects related to the involvement, type, and number of collaborators. In a *light* collaboration, details of the optimization problem regarding complete problem formulation are available to the optimization experts, thereby not requiring much collaboration between the two expert groups. In a *medium* collaboration, besides the details of the problem formulation, further information is required either due to the complexity involved in the problem or due to the nature of the problem. Optimization experts must share intermediate results with domain-specific experts to get further information to improve the optimization method. In a *strong* collaboration, both groups must engage in more collaboration to solve the problem. This can happen if the objective and constraint functions cannot be shared with the optimization experts due to confidentiality issues or the unavailability of computing resources with the optimization group.

7.1.4 Case Studies

The blueprint for collaborative optimization can be put into practice in different ways. We demonstrate two case studies to illustrate.

Case Study 1: Cylinder Head Water Jacket. As a case study, the collaboration with an automobile company regarding the optimization of a Cylinder Head Water Jacket is discussed. A study focusing on the optimization itself has already been published [28]; however, details of the collaborative process itself were not part of the study. Initially, the industrial partner with domain-specific expertise sought an optimization expert to solve an industrial design problem that could not be solved suitably with a commercial solver. Most commercial solvers are generic and not ideal candidate solution methods to find an acceptable solution with a solution evaluation budget. Thus, a collaboration was initiated. The industrial collaborator had a background in engineering and more than a decade of experience in engineering design. The optimization experts are specialized in multi-objective and evolutionary optimization, and the team consisted of one professor and two Ph.D. students. The goal to design an algorithm that can deal with a constrained multi-objective optimization problem where each evaluation requires computationally expensive simulation was defined (phase ‘S’). Due to the time-consuming evaluation function, the overall evaluation budget was limited to 120 simulations per optimization run. However, the algorithmic overhead could be significantly higher and even reach a couple of minutes to find new solutions in each iteration. Secondly, the algorithm was first developed on test problems with similar characteristics but computationally inexpensive functions (phase ‘O’). Even though the algorithm has been designed from scratch, the usage of existing modules and algorithms of pymoo [29] – a Python framework for multi-objective optimization – was handy for prototyping and even sped up the algorithm’s development. Bi-monthly discussions between all collaborators accompanied the research process. Thirdly, multiple runs on the live environment (phase ‘L’) optimizing the Cylinder Head Water Jacket have been employed. Because the optimization experts did not have access to the simulation software, the optimization run was carried out manually by sending engineering designs back and

forth via e-mail. This way, multiple experiments have been run, and at the same time, the results were verified (phase ‘Ve’). Thus, the execution of phases ‘L’ and ‘Ve’ happened simultaneously. As the method has been confirmed to be suitable for the optimization problem, the source code has finally been delivered to the industrial partner. Delivering the source code ensured the algorithm was used in the future for similar problems (phase ‘R’). Moreover, a final meeting discussing the method and assessing the project’s success has taken place between all collaborators and coworkers from related departments.

Case Study 2: Engine Design. In another auto-industry project executed at the COIN Lab, the initial task of the industry designers was to reduce the weight of an automobile engine from its current weight by 10 kg (phase ‘S’). The problem involves 145 discrete variables, which can be varied within specified lower and upper bounds, 146 constraints which all must be satisfied, and six conflicting objectives, which all must be optimized. The objective and constraint functions were not available in explicit form; rather, a black-box executable was supplied. Initial collaborations between the two groups revealed that the functions’ gradients were also available from the executable routine. The availability of gradient information allowed the optimization experts to devise a new operator – a gradient-based local search approach – to improve a solution locally. Another study revealed that when 2.5 million random solutions were evaluated, no single solution was found to be feasible. The majority of the search space being infeasible prompted optimization experts to devise an algorithm to infinitely emphasize every feasible solution. A generic many-objective optimization algorithm (NSGA-III [279]) was modified to develop a customized method (phase ‘O’) to find feasible non-dominated solutions. The customized algorithm was directly applied to solve the engine design problem (phase ‘L’). The developed method resulting from a close collaboration found a new engine, 17 kilograms lighter than the current design, which is 7 kg better than originally desired (phase ‘Ve’). Further information on obtained results can be found from [336]. This study mostly used a light collaboration mode.

The power of collaborative optimization came next from the designers. The multiplicity of designs obtained by customized NSGA-III motivated the designers to set the next goal (phase ‘R’)

to find *multiple* engines with identical weight. This promoted the whole ‘SOLVeR’ procedure to a new specification (phase ‘S’). Optimization experts then introduced the concept of *niche-preservation* – survival of similar solutions as clusters – to develop a new optimization method (phase ‘O’). Niche preservation is a new optimization technique that was possible to be developed only by a collaborative problem-solving approach. The method was applied to the real problem (phase ‘L’), and three different pairs of engines, each having an identical weight, were obtained (phase ‘Ve’). The SOLVeR approach’s ability to reduce the engine weight by more than 10 kg motivated the designers to repeat the process (phase ‘R’) to a third cycle in which they aspired to reduce the weight further by relaxing the constraint bounds.

Relaxation of constraints to improve objective function was dealt with by formulating a two-objective optimization problem (phase ‘O’). One of the objectives was to minimize the amount of constraint violation from the current best solution; the second conflicting objective was to maximize the amount of weight reduction from the current best solution. The bi-objective optimization method found multiple trade-off solutions with different combinations of constraint violations and weight reductions (phase ‘L’). The solutions allowed designers to better understand the trade-off before choosing a final solution for implementation (phase ‘Ve’).

None of these extensions achieved with specific and innovative optimization methods were academic, nor were they standard optimization practices. However, they revealed alternate solutions close to the designers’ interests, so they had a plethora of pertinent solutions before choosing one. Such a design feat was possible only with a collaborative optimization procedure.

7.1.5 Summary of Section 7.1

Optimization is an interdisciplinary research field and a substantial part of various sciences. Thus, collaboration is vital to tackle problem-solving tasks in all kinds of disciplines successfully. Whereas most studies focus on the outcome of such collaborative optimization, this study puts the collaborative process itself as the center of attention. To guide the process of collaboration, we have proposed a blueprint following the *SOLVeR* approach consisting of five phases: Specification

of the Problem, Optimization and Algorithm Design, Live Test, Verification of Method and Results, and Repetitions and Lesson. We have defined the domain and the optimization expert's roles and responsibilities for each phase and highlighted the other supporting activities during collaborative optimization. Moreover, two case studies have illustrated how the blueprint for collaborative optimization was implemented in practice.

This section has demonstrated the importance of performing a collaborative optimization rather than a silo-based optimization without any intermediate interactions from domain-specific experts. Collaborative optimization makes solving challenging problems quicker and opens up new avenues for more flexible and practical optimization studies. Through collaboration, the experts benefit from each other, which results in understanding different facets of the application and gaining insights more efficiently.

7.2 pymoo: Multi-Objective Optimization in Python

Collaborative optimization implies that different kinds of experts are working together. However, one should not assume that everyone can write code or is familiar with the usage of programming languages. Thus, standard software is one way of making research accessible to a larger audience. Nevertheless, such deliverables are limited by definition because the software package is mainly of a black-box nature, and no further modifications can be made. In contrast to standard software, open-source *frameworks* are publicly available and are ideal for customizing optimization methods. For this reason, the usage of frameworks offers a good trade-off between not having to start developing an optimization method from scratch and having access to existing state-of-the-art optimization algorithms. As someone who has made an effort to develop an optimization framework, we would like to share our development's common design principles and features. Our optimization framework pymoo is an open-source (evolutionary) multi-objective optimization framework written in Python, and we are proud to be able to say that it has gained some popularity over the last years [29].

7.2.1 Introduction

Optimization plays an essential role in many scientific areas, such as engineering, data analytics, and deep learning. These fields are fast-growing, and their concepts are employed for various purposes, for instance, gaining insights from large data sets or fitting accurate prediction models. Efficient implementation in a suitable programming language is essential whenever an algorithm must handle a significantly large amount of data. Python [337] has become the programming language of choice over the last few years for the research areas mentioned above because it is not only easy to use but good community support also exists. Python is a high-level, cross-platform, and interpreted programming language that focuses on code readability. A large number of high-quality libraries are available and support for any kind of scientific computation is ensured. These characteristics make Python an appropriate tool for many research and industry projects where the investigations can be complex.

A fundamental principle of research is to ensure the reproducibility of studies and provide access to the research materials whenever possible. In computer science, this translates to a sketch of an algorithm and the implementation itself. However, the implementation of optimization algorithms can be challenging, and, specifically, benchmarking is time-consuming. Having access to either a good collection of different source codes or a comprehensive library is time-saving and reduces the probability of an error-prone implementation from scratch.

To address this need for multi-objective optimization in Python, we introduce pymoo. The goal of our framework is not only to provide state-of-the-art optimization algorithms but also to cover different aspects related to the optimization process itself. We have implemented single-, multi-, and many-objective test problems, which can be used as a test-bed for algorithms. In addition to the objective and constraint values of test problems, gradient information can be retrieved through automatic differentiation [338]. Moreover, a parallel evaluation of solutions can be implemented through vectorized computations, multi-threaded execution, and distributed computing. Further, pymoo provides implementations of performance indicators to measure the quality of results obtained by a multi-objective optimization algorithm. Tools for an explorative analysis through

visualization of lower and higher-dimensional data are available, and multi-criteria decision-making methods guide selecting a single solution from a solution set based on preferences.

Our framework is designed to be extendable through its modular implementation. For instance, a genetic algorithm is assembled in a plug-and-play manner by making use of specific sub-modules, such as initial sampling, mating selection, crossover, mutation, and survival selection. Each sub-module takes care of an aspect independently, and, therefore, variants of algorithms can be initiated by passing different combinations of sub-modules. This concept allows end-users to incorporate domain knowledge through custom implementations. For example, in an evolutionary algorithm, a biased initial sampling module created with the knowledge of domain experts can guide the initial search.

Furthermore, we like to mention that our framework is well-documented, with a large number of available code snippets. We created a starter's guide for users to become familiar with our framework and demonstrate its capabilities. As an example, it shows the optimization results of a bi-objective optimization problem with two constraints. An extract from the guide will be presented in this chapter. Moreover, we explain each algorithm and code needed to run it on a suitable optimization problem in our software documentation. Additionally, we show a definition of test problems and provide a plot of their fitness landscapes. The framework documentation is built using Sphinx [339], and the correctness of modules is ensured by automatic unit testing [340]. Most algorithms have been developed in collaboration with the second author and benchmarked extensively against the original implementations.

7.2.2 Related Work

In the last decades, various optimization frameworks in diverse programming languages have been developed. However, some of them only partially cover multi-objective optimization. In general, the choice of a suitable framework for an optimization task is a multi-objective problem itself. Moreover, some criteria are rather subjective, for instance, the usability and extendibility of a framework. Therefore, the assessment regarding criteria and the decision-making process differ

from user to user. For example, one might have decided on a programming language first, either because of personal preference or a project constraint and then searched for a suitable framework. One might give more importance to the overall features of a framework, for example, parallelization or visualization, over the programming language itself. An overview of some existing multi-objective optimization frameworks in Python is listed in Table 7.1, each of which is described in the following.

Recently, the well-known multi-objective optimization framework jMetal [341] developed in Java [342] has been ported to a Python version, namely jMetalPy [343]. The authors aim to further extend it and to make use of the full feature set of Python, for instance, data analysis and data visualization. In addition to traditional optimization algorithms, jMetalPy also offers methods for dynamic optimization. Moreover, the post-analysis of performance metrics of an experiment with several independent runs is automated.

Parallel Global Multiobjective Optimizer, PyGMO [344], is an optimization library for the easy distribution of massive optimization tasks over multiple CPUs. It uses the generalized island-model paradigm for the coarse-grained parallelization of optimization algorithms and, therefore, allows users to develop asynchronous and distributed algorithms.

Platypus [345] is a multi-objective optimization framework that offers implementations of state-of-the-art algorithms. It enables users to generate an experiment with various algorithms and provides post-analysis methods based on metrics and visualization.

A Distributed Evolutionary Algorithms in Python (DEAP) [346] is a novel evolutionary computation framework for rapid prototyping and testing of ideas. Even though DEAP does not focus on multi-objective optimization, due to the modularity and extendibility of the framework, multi-objective algorithms can be developed. Moreover, parallelization and load-balancing tasks are supported out of the box.

Inspyred [347] is a framework for creating bio-inspired computational intelligence algorithms in Python, which is not focused on multi-objective algorithms directly, but on evolutionary computation in general. However, an example for NSGA-II [10] is provided, and other multi-objective

Table 7.1: Multi-objective optimization frameworks in Python.

Name	License	Focus on multi-objective	Pure Python	Visualization	Decision Making
jMetalPy	MIT	✓	✓	✓	✗
PyGMO	GPL-3.0	✓	✗	✗	✗
Platypus	GPL-3.0	✓	✓	✗	✗
DEAP	LGPL-3.0	✗	✓	✗	✗
Inspyred	MIT	✗	✓	✗	✗
pymoo	Apache 2.0	✓	✓	✓	✓

algorithms can be implemented through the modular implementation of the framework.

If the search for frameworks is not limited to Python, other popular frameworks should be considered: PlatEMO [245] in Matlab, MOEA [348] and jMetal [343] in Java, jMetalCpp [349] and PaGMO [344] in C++. Of course, this is not an exhaustive list and readers may search for other available options.

7.2.3 Architecture

Software architecture is fundamentally important to keep source code organized. On the one hand, it helps developers and users to get an overview of existing classes, and on the other hand, it allows flexibility and extendibility by adding new modules. Figure 7.3 visualizes the architecture of pymoo. The first level of abstraction consists of optimization problems, algorithms, and analytics. Each of the modules can be categorized into more detail and consists of multiple sub-modules.

- (i) *Problems*: Optimization problems in our framework are categorized into single-, multi-, and many-objective test problems. Gradients are available through automatic differentiation, and parallelization can be implemented by using a variety of techniques.
- (ii) *Optimization*: Since most of the algorithms are based on evolutionary computations, operators such as sampling, mating selection, crossover, and mutation have to be chosen or

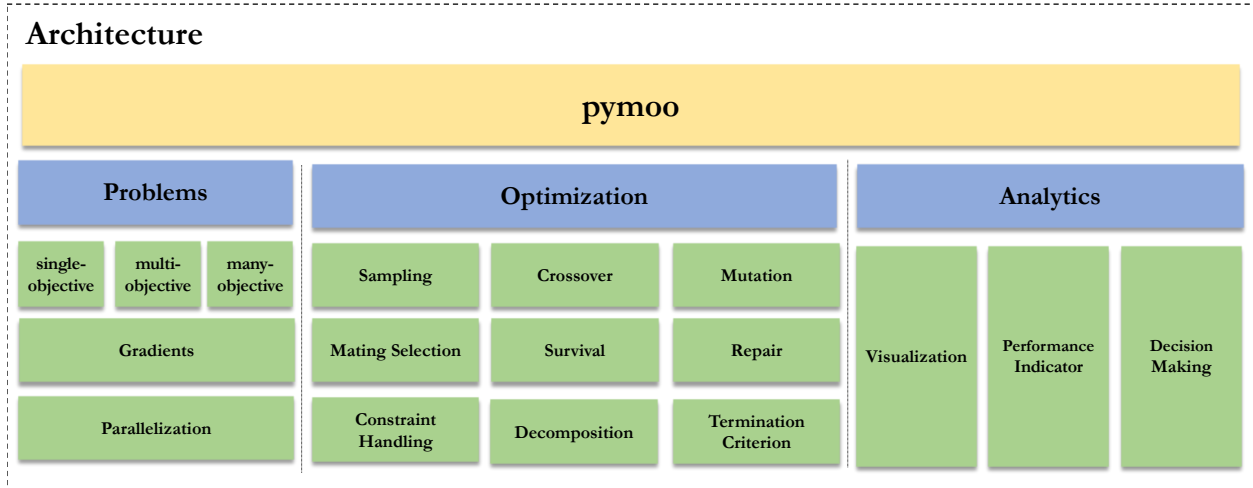


Figure 7.3: Software architecture of pymoo.

implemented. Furthermore, because many problems in practice have one or more constraints, a methodology for handling those must be incorporated. Some algorithms are based on decomposition, which splits the multi-objective problem into many single-objective problems. Moreover, when the algorithm is used to solve the problem, a termination criterion must be defined either explicitly or implicitly by the implementation of the algorithm.

- (iii) *Analytics*: During and after an optimization run, analytics support the understanding of data. First, intuitively the design space, objective space, or other metrics can be explored through visualization. Moreover, to measure the convergence and/or diversity of a Pareto-optimal set, performance indicators can be used. For real-parameter problems, the recently proposed theoretical KKT proximity metric [350, 351] computation procedure is included in pymoo to compute the proximity of a solution to the true Pareto-optimal front, despite not knowing its exact location. In order to support the decision-making process, either through finding points close to the area of interest in the objective space or high trade-off solutions. This can be applied either during an optimization run to mimic interactive optimization or as a post-analysis.

In the remainder of the chapter, we will discuss each of the modules mentioned in more detail.

7.2.4 Problems

It is common practice for researchers to evaluate the performance of algorithms on a variety of test problems. Since we know no single-best algorithm for all arbitrary optimization problems exist [352], this helps to identify problem classes where the algorithm is suitable. Therefore, a collection of test problems with different numbers of variables, objectives, or constraints and alternating complexity becomes handy for algorithm development. Moreover, in a multi-objective context, test problems with different Pareto front shapes or varying variable densities close to the optimal region are of interest.

7.2.4.1 Implementations

In our framework, we categorize test problems regarding the number of objectives: single-objective (1 objective), multi-objective (2 or 3 objectives), and many-objective (more than 3 objectives). Test problems implemented in pymoo are listed in Table 7.2. For each problem, the number of variables, objectives, and constraints are indicated. If the test problem is scalable to any of the parameters, we label the problem with (*s*). If the problem is scalable, but a default number was originally proposed, we indicate that with surrounding brackets. In case the category does not apply, for example, because we refer to a test problem family with several functions, we use (\cdot).

The implementations in pymoo let end-users define what values of the corresponding problem should be returned. On an implementation level, the `evaluate` function of a `Problem` instance takes a list `return_value_of` which contains the type of values being returned. By default the objective values "F" and if the problem has constraints the constraint violation "CV" are included. The constraint function values can be returned independently by adding "G". This gives developers the flexibility to receive the values that are needed for their methods.

7.2.4.2 Parallelization

If evaluation functions are computationally expensive, a serialized evaluation of a set of solutions can become the bottleneck of the overall optimization procedure. For this reason, parallelization

Table 7.2: Multi-objective optimization test problems.

Problem	Variables	Objectives	Constraints
Single-Objective			
Ackley	(s)	1	-
Cantilevered Beams	4	1	2
Griewank	(s)	1	-
Himmelblau	2	1	-
Knapsack	(s)	1	1
Pressure Vessel	4	1	4
Rastrigin	(s)	1	-
Rosenbrock	(s)	1	-
Schwefel	(s)	1	-
Sphere	(s)	1	-
Zakharov	(s)	1	-
G1-9	(·)	(·)	(·)
Multi-Objective			
BNH	2	2	2
Carside	7	3	10
Kursawe	3	2	-
OSY	6	2	6
TNK	2	2	2
Truss2D	3	2	1
Welded Beam	4	2	4
CTP1-8	(s)	2	(s)
ZDT1-3	(30)	2	-
ZDT4	(10)	2	-
ZDT5	(80)	2	-
ZDT6	(10)	2	-
Many-Objective			
DTLZ 1-7	(s)	(s)	-
CDTLZ	(s)	(s)	-
DTLZ1 ⁻¹	(s)	(s)	-
SDTLZ	(s)	(s)	-
WFG	(s)	(s)	-

is desired for utilizing existing computational resources more efficiently and the distribution of long-running calculations. In pymoo, the evaluation function receives a set of solutions if the algorithm uses a population. This empowers the user to implement any kind of parallelization as long as the objective values for all solutions are written as an output when the evaluation function terminates. In our framework, a couple of possibilities to implement parallelization exist:

- (i) *Vectorized Evaluation:* A common technique to parallelize evaluations is to use matrices where each row represents a solution. Therefore, a vectorized evaluation refers to a column that includes the variables of all solutions. By using vectors, the objective values of all solutions are calculated at once. To run calculations on a GPU, implementing support for PyTorch [353] tensors can be done with little overhead given suitable hardware and correctly installed drivers.
- (ii) *Threaded Loop-wise Evaluation:* If the function evaluation should occur independently, a `for` loop can be used to set the values. By default, the evaluation is serialized, and no calculations occur in parallel. By providing a keyword to the evaluation function, pymoo spawns a thread for each evaluation and manages those by using the default thread pool implementation in Python. This behavior can be implemented out of the box, and the number of parallel threads can be modified.
- (iii) *Distributed Evaluation:* If the evaluation should not be limited to a single machine, the evaluation itself can be distributed to several workers or a whole cluster. We recommend using Dask [354] which enables distributed computations on different levels. For instance, the matrix operation itself can be distributed, or a whole function can be outsourced. Similar to the loop-wise evaluation, each individual can be evaluated element-wise by sending it to a worker.

7.2.5 Optimization Module

The optimization module provides different kinds of sub-modules to be used in algorithms. Some of them are more of a generic nature, such as decomposition and termination criterion, and others are more related to evolutionary computing. By assembling those modules together, algorithms are built.

7.2.5.1 Algorithms

Available algorithm implementations in pymoo are listed in Table 7.3. Compared to other optimization frameworks, the list of algorithms may look rather short; however, each algorithm is customizable, and variants can be initialized with different parameters. For instance, a Steady-State NSGA-II [355] can be initialized by setting the number of offspring to one. This can be achieved by supplying this as a parameter in the initialization method. Moreover, it is worth mentioning that many-objective algorithms, such as NSGA-III or MOEAD, require reference directions to be provided. The reference directions are commonly desired to be uniform or to have a bias toward a region of interest. Our framework offers an implementation of the Das and Dennis method [280] for a fixed number of points (fixed with respect to a parameter often referred to as *partition number*) and a recently proposed Riesz-Energy based method which creates a well-spaced point set for an *arbitrary* number of points and is capable of introducing a bias towards preferred regions in the objective space [276].

7.2.5.2 Operators

The following evolutionary operators are available:

- (i) *Sampling*: The initial population is mostly based on random sampling. In some cases, it might be based on domain knowledge or a set of existing solutions whose performance has already been assessed. Otherwise, it can be sampled randomly for real, integer, or binary variables. Additionally, Latin-Hypercube Sampling [234] can be used for real variables.

Table 7.3: Multi-objective optimization algorithms.

Algorithm	Reference
GA	[1, 32]
BRKGA	[356]
DE	[107]
Nelder-Mead	[357]
CMA-ES	[112, 113]
NSGA-II	[10]
RNSGA-II	[358]
NSGA-III	[279, 258, 12]
UNSGA-III	[359]
RNSGA-III	[360]
MOEAD	[11]

- (ii) *Crossover*: A variety of crossover operators for different type of variables are implemented. In Figure 7.4 some of them are presented. Figures 7.4a to 7.4d help to visualize the information exchange in a crossover with two parents being involved. Each row represents an offspring and each column a variable. The corresponding boxes indicate whether the values of the offspring are inherited from the first or from the second parent. For one- and two-point crossovers, it can be observed that either one or two cuts in the variable sequence exist. Contrarily, the Uniform Crossover (UX) does not have any clear pattern because each variable is chosen randomly either from the first or from the second parent. For the Half Uniform Crossover (HUX), half of the variables, which are different, are exchanged. For the purpose of illustration, we have created two parents that have different values in 10 different positions. For real variables, Simulated Binary Crossover [361] mimics the combination of binary encoded variables. In Figure 7.4e, the probability distribution when the parents $x - 1 = 0.2$ and $x - 2 = 0.8$ where $x - i \in [0, 1]$ with $\eta = 0.8$ are recombined is shown. Analogously, in case of integer variables we subtract 0.5 from the lower and add $(0.5 - \epsilon)$ to the upper bound before applying the crossover and round to the nearest integer afterwards (see Figure 7.4f).
- (iii) *Mutation*: For real and integer variables Polynomial Mutation [362, 9] and for binary variables Bitflip mutation [1] is provided.

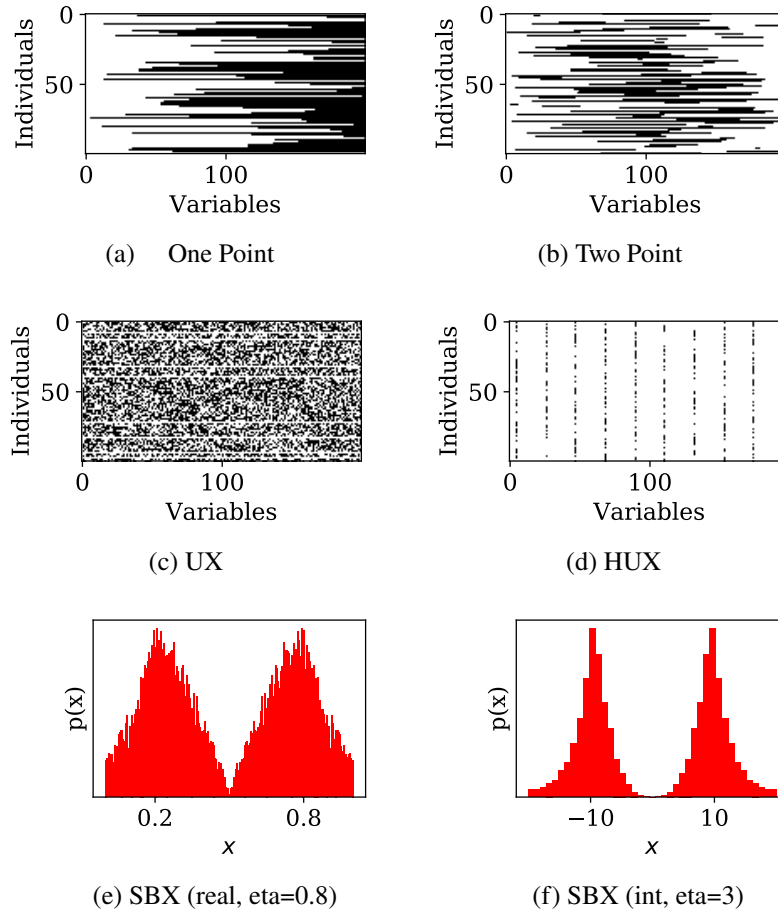


Figure 7.4: Illustration of some crossover operators for different variables types.

Different problems require different types of operators. In practice, if a problem is supposed to be solved repeatedly and routinely, it makes sense to *customize* the evolutionary operators to improve the convergence of the algorithm. Moreover, for custom variable types, for instance trees or mixed variables [363], custom operators [6] can be implemented easily and called by algorithm class. Our software documentation contains examples for custom modules, operators, and variable types.

7.2.5.3 Termination Criterion

For every algorithm, it must be determined when it should terminate a run. This can be simply based on a predefined number of function evaluations, iterations, or a more advanced criterion, such

as the change of a performance metric over time. For example, we have implemented a termination criterion based on the variable and objective space difference between generations. In order to make the termination criterion more robust, the last k generations are considered. The largest movement from a solution to its closest neighbor is tracked across generations, and whenever it is below a certain threshold, the algorithm is considered to have converged. Analogously, the movement in the objective space can also be used. In the objective space, however, normalization is more challenging and has to be addressed carefully. The default termination criterion for multi-objective problems in pymoo keeps track of the boundary points in the objective space and uses them, when they have settled down, for normalization. More details about the proposed termination criterion can be found in [364].

7.2.5.4 Decomposition

Decomposition transforms multi-objective problems into many single-objective optimization problems [365]. Such a technique can be either embedded in a multi-objective algorithm and solved simultaneously or independently using a single-objective optimizer. Some decomposition methods are based on the l_p -metrics with different p values. For instance, a naive but frequently used decomposition approach is the Weighted-Sum Method ($p = 1$), which is known to be incapable of converging to the non-convex part of a Pareto front [9]. Moreover, instead of summing values, Tchebysheff Method ($p = \infty$) considers only the maximum value of the difference between the ideal point and a solution. Similarly, the Achievement Scalarization Function (ASF) [366] and a modified version Augmented Achievement Scalarization Function (AASF) [367] use the maximum of all differences. Furthermore, Penalty Boundary Intersection (PBI) [11] is calculated by a weighted sum of the norm of the projection of a point onto the reference direction and the perpendicular distance. Also, it is worth noting that normalization is essential for any kind of decomposition. All decomposition techniques mentioned above are implemented in pymoo.

7.2.6 Analytics

7.2.6.1 Performance Indicators

The comparison regarding performance is rather simple for single-objective optimization algorithms because each optimization run results in a single best solution. In multi-objective optimization, however, each run returns a non-dominated set of solutions. In order to compare sets of solutions, various performance indicators have been proposed in the past [368]. In pymoo most commonly used performance indicators are described:

- (i) *GD/IGD*: Given the Pareto front PF the deviation between the non-dominated set S found by the algorithm and the optimum can be measured. Following this principle, Generational Distance (GD) indicator [369] calculates the average Euclidean distance in the objective space from each solution in S to the closest solution in PF. This measures the convergence of S, but does not indicate whether a good diversity on the Pareto front has been reached. Similarly, Inverted Generational Distance (IGD) indicator [240] measures the average Euclidean distance in the objective space from each solution in PF to the closest solution in S. The Pareto front as a whole needs to be covered by solutions from S to minimize the performance metric. Thus, the lower the GD and IGD values, the better is the set. However, IGD is known to be not Pareto-compliant [370].
- (ii) *GD+/IGD+*: A variation of GD and IGD has been proposed in [370]. The Euclidean distance is replaced by a distance measure that takes the dominance relation into account. The authors show that IGD+ is weakly Pareto compliant.
- (iii) *Hypervolume*: Moreover, the dominated portion of the objective space can be used to measure the quality of non-dominated solutions [371]. The higher the hypervolume, the better is the set. Instead of the Pareto front, a reference point needs to be provided. It has been shown that Hypervolume is Pareto compliant [241]. Because the performance metric becomes computationally expensive in higher dimensional spaces, the exact measure

becomes intractable. However, we plan to include some proposed approximation methods in the near future.

Performance indicators are used to compare existing algorithms. Moreover, the development of new algorithms can be driven by the goodness of different metrics themselves.

7.2.6.2 Visualization

The visualization of intermediate steps or the final result is inevitable. In multi and many-objective optimization, visualization of the objective space is of interest so that trade-off information among solutions can be easily experienced from the plots. Depending on the dimension of the objective space, different types of plots are suitable to represent a single or a set of solutions. In pymoo the implemented visualizations wrap around the well-known plotting library in Python Matplotlib [372]. Keyword arguments provided by Matplotlib itself are still available, which allows modifying, for instance, the color, thickness, opacity of lines, points, or other shapes. Therefore, all visualization techniques are customizable and extendable.

For two or three objectives, scatter plots (see Figure 7.5a and 7.5b) can give a good intuition about the solution set. Trade-offs can be observed by considering the distance between two points. It might be desired to normalize each objective to make sure a comparison between values is based on relative and not absolute values. Pairwise Scatter Plots (see Figure 7.5c) visualize more than three objectives by showing each pair of axes independently. The diagonal is used to label the corresponding objectives.

Also, high-dimensional data can be illustrated by Parallel Coordinate Plots (PCP) as shown in Figure 7.5d. All axes are plotted vertically and represent an objective. Each solution is illustrated by a line from left to right. The intersection of a line and an axis indicate the value of the solution regarding the corresponding objective. For the purpose of comparison, solution(s) can be highlighted by varying color and opacity.

Moreover, a common practice is to project the higher dimensional objective values onto the 2D plane using a transformation function. Radviz (Figure 7.5e) visualizes all points in a circle,

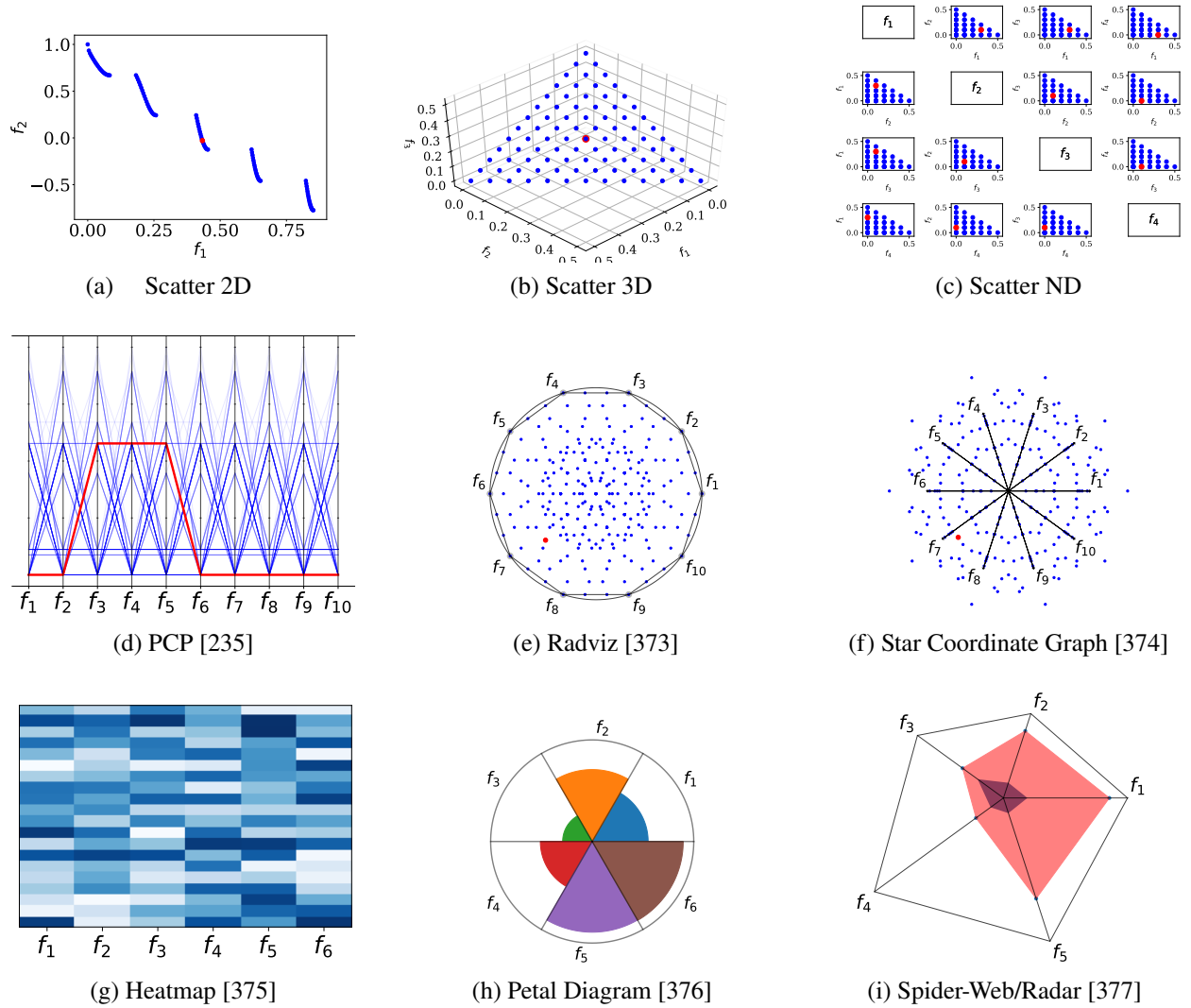


Figure 7.5: Different visualization methods coded in pymoo.

and the objective axes are uniformly positioned around the perimeter. Considering a minimization problem and a set of non-dominated solutions, an extreme point very close to an axis represents the worst solution for that corresponding objective but is comparably "good" in one or many other objectives. Similarly, Star Coordinate Plots (see Figure 7.5f) illustrate the objective space, except that the transformation function allows solutions outside of the circle.

Heatmaps (see Figure 7.5g) are used to represent the goodness of solutions through colors. Each row represents a solution and each column a variable. We leave the choice to the end-user of what color map to use and whether light or dark colors illustrate better or worse solutions. Also,

solutions can be sorted lexicographically by their corresponding objective values.

Instead of visualizing a set of solutions, one solution can be illustrated at a time. The Petal Diagram (Figure 7.5h) is a pie diagram where the objective value is represented by each piece's diameter. Colors are used to further distinguish the pieces. Finally, the Spider-Web or Radar Diagram (Figure 7.5i) shows the values of the objective as a point on an axis. The ideal and nadir point [9] is represented by the inner and outer polygon. By definition, the solution lies in between those two extremes. If the objective space ranges are scaled differently, normalization for the purpose of plotting can be enabled, and the diagram becomes symmetric. New and emerging methods for visualizing more than efficient three-dimensional solutions, such as 2.5-dimensional PaletteViz plots [378], would be implemented in the future.

7.2.6.3 Decision Making

In practice, after obtaining a set of non-dominated solutions, a single solution has to be chosen for implementation. Pymoo provides a few “a posteriori” approaches for decision making [9].

- (i) *Compromise Programming*: One way of making a decision is to compute the value of a scalarized and aggregated function and select one solution based on the minimum or maximum value of the function. In pymoo, a number of scalarization functions described in Section 7.2.5.4 can be used to come to a decision regarding desired weights of objectives.
- (ii) *Pseudo-Weights*: However, a more intuitive way to choose a solution out of a Pareto front is the pseudo-weight vector approach proposed in [9]. The pseudo-weight w_i for the i -th objective function is calculated by:

$$w_i = \frac{(f_i^{\max} - f_i(x)) / (f_i^{\max} - f_i^{\min})}{\sum_{m=1}^M (f_m^{\max} - f_m(x)) / (f_m^{\max} - f_m^{\min})}. \quad (7.1)$$

The normalized distance to the worst solution regarding each objective i is calculated. It is interesting to note that for non-convex Pareto fronts, the pseudo-weight does not correspond to the result of an optimization using the weighted-sum method. A solution having the

closest pseudo-weight to a target preference vector of objectives (f_1 being preferred twice as important as f_2 results in a target preference vector of (0.667, 0.333)) can be chosen as the preferred solution from the efficient set.

- (iii) *High Trade-Off Solutions*: Furthermore, high trade-off solutions are usually of interest but not straightforward to identify in higher-dimensional objective spaces. We have implemented the procedure proposed in [379]. It was described to be embedded in an algorithm to guide the search; we, however, use it for post-processing. The metric for each solution pair x_i and x_j in a non-dominated set is given by:

$$T(x_i, x_j) = \frac{\sum_{i=1}^M \max[0, f_m(x_j) - f_m(x_i)]}{\sum_{i=1}^M \max[0, f_m(x_i) - f_m(x_j)]}, \quad (7.2)$$

where the numerator represents the aggregated sacrifice and the denominator represents the aggregated gain. The trade-off measure $\mu(x_i, S)$ for each solution x_i with respect to a set of neighboring solutions S is obtained by:

$$\mu(x_i, S) = \min_{x_j \in S} T(x_i, x_j). \quad (7.3)$$

It finds the minimum $T(x_i, x_j)$ from x_i to all other solutions $x_j \in S$. Instead of calculating the metric with respect to all others, we provide the option to only consider the k closest neighbors in the objective space to reduce the computational complexity. Based on circumstances, the ‘min’ operator can be replaced with ‘average’, or ‘max’, or any other suitable operator. Thereafter, the solution having the maximum μ can be chosen as the preferred solution, meaning that this solution causes a maximum sacrifice in one of the objective values for a unit gain in another objective value for it be the most valuable solution for implementation.

The above methods are algorithmic but require user interaction to choose a single preferred solution. However, in real practice, a more problem-specific decision-making method must be used, such as an interaction EMO method suggested elsewhere [380]. We like to emphasize that multi-objective frameworks should include multi-criteria decision-making methods and support end-users in choosing a solution out of a trade-off solution set.

7.2.7 Summary of Section 7.2

This chapter has introduced pymoo, a multi-objective optimization framework in Python. We have presented the overall architecture of the framework consisting of three core modules: Problems, Optimization, and Analytics. Each module has been described in-depth and illustrative examples have been provided. We have shown that our framework covers various aspects of multi-objective optimization, including the visualization of high-dimensional spaces and multi-criteria decision-making to finally select a solution out of the obtained solution set. One distinguishing feature of our framework with other existing ones is that we have provided a few options for various key aspects of a multi-objective optimization task, providing standard evolutionary operators for optimization, standard performance metrics for evaluating a run, standard visualization techniques for showcasing obtained trade-off solutions, and a few approaches for decision-making.

However, the framework can be extended to make it more comprehensive, and we are constantly adding new capabilities based on practicalities learned from our collaboration with industries. In the future, we plan to implement more optimization algorithms and test problems to provide end-users with more choices. Also, we aim to implement some methods from classical literature on single-objective optimization, which can also be used for multi-objective optimization through decomposition or embedded as a local search. So far, we have provided a few basic performance metrics. We plan to extend this by creating a module that automatically runs a list of algorithms on test problems and provides statistics of different performance indicators.

Furthermore, we like to mention that any kind of contribution is more than welcome. We see our framework as a collaborative collection from and to the multi-objective optimization community. By adding a method or algorithm to pymoo the community can benefit from a growing comprehensive framework, and it can help researchers to advertise their methods. Interested researchers are welcome to contact the authors. In general, different kinds of contributions are possible, and more information can be found online. Moreover, we would like to mention that even though we try to keep our framework as bug-free as possible, in case of exceptions during the execution or doubt of correctness, please contact us directly or use our issue tracker.

7.3 Summary of the Chapter

This chapter has provided a blueprint for optimization in practice. We have proposed a methodology following the SOLVeR acronym. While collaboratively solving an optimization problem, each phase is accompanied by supporting activities such as project management or interdisciplinary communication. The application of the SOLVeR method has been shown in two case studies. Additionally, we have provided insights into the software architecture and usage of pymoo, a well-known optimization framework used in academia and industry. The introduction into the framework has given a fundamental understanding of the different software components and has been helpful to get end-users started solving their optimization problem using a standard toolkit.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Summary of Contributions

This thesis has focused on computational expense – an important problem characteristic in practice. In Chapter 1, we have discussed the origin of time-consuming evaluation functions and why they need to be addressed when solving real-world optimization problems. Moreover, we have presented the idea of using surrogates, which is the predominant approach in academia and industry. In Chapter 2, some more basics and definitions about optimization, different well-known models, and a standard surrogate-based method were provided. Related literature, including a categorization of existing surrogate-based approaches, was discussed in Chapter 3. Besides a theme-wise overview of previous works, we have identified current trends of surrogate-assisted optimization and open issues to be addressed. Chapter 4 first have presented a method that incorporates surrogate models into different kinds of well-known metaheuristics proposed for single-objective optimization. By keeping the capability of generalization in mind, this idea was extended to multi-objective optimization algorithms, as this needs to be commonly solved in practice. In Chapter 5, the optimization of multiple independently computable and mixed computationally expensive functions was investigated. This often occurs in practice when the performance assessment (constraints and objectives) of a design requires running multiple third-party software packages or calculating closed-form equations. We have proposed a method when all objectives are computationally expensive and the constraints are inexpensive. In such a case, one wants to ensure that only feasible designs are sent to the time-consuming evaluation function. Moreover, we have developed an evolutionary algorithm for mixed computationally expensive functions evaluating the target functions in a specific order that maximizes the information gain in each iteration. In Chapter 6 we have presented two case studies. First, the optimization of a bi-objective head water jacket design problem with computationally expensive functions. Second, a constrained bi-objective application exploring the electric machine

design with computationally expensive objectives and inexpensive constraints. Some studies and projects in this thesis required collaborating with other – sometimes less optimization-literate – researchers. Thus, we have presented a blueprint for collaborative optimization in Chapter 7. Besides the collaboration, implementing the optimization code itself can be quite challenging. As founders and maintainers of an open-source package (which also was used for the majority of this thesis), we have described the architecture and usage of our framework focusing on multi-objective optimization.

8.2 Future Directions

Throughout this dissertation, all kinds of questions have been approached in different ways. In the very end, we would now like to identify possible future research directions, including matters that deserve to be investigated in more detail.

We proposed a generic methodology for surrogate-assisted optimization applied to unconstrained and constrained, single- and multi-objective problems in Chapter 4. Even though we tested our method for a variety of metaheuristics, some additional studies shall broaden the scope of this work further. Furthermore, the constrained handling for multi-objective optimization problems has demonstrated competitive results to another algorithm; however, it has room for improvement. Other interesting future research directions are the behavior of the proposed method with an increasing amount of variables (20 or more) and variation of the evaluation budget (imitating application evaluations of mediocre or high expense). This raises questions about the time a surrogate needs to be fitted and the evaluation itself. Throughout this thesis, we assumed that the expensive evaluation of a solution justified any additional computational burden (for instance, running a whole surrogate selection procedure for each objective and constraint). However, for mediocre expensive functions, let us say for one minute, one has to revise if surrogate incorporation is still beneficial.

Heterogeneously expensive evaluations are a practical matter that needs to be paid attention to when solving a real-world problem. First, our focus was on dealing with different evaluation expenses (for objectives and constraints) but not the surrogate incorporation itself in Chapter 5.

Thus, a future research direction should be the holistic consideration of both aspects at the same time attempting to solve a real-world optimization problem directly. For instance, suitable candidates are multi-physics design problems often requiring multiple software packages to be executed. Moreover, we have assumed that the evaluation time for each solution's target is known and is approximately the same for all designs. However, for some simulations, the evaluation time may vary. For instance, let us assume the configuration of a car controller is optimized where the simulation takes place until a time threshold has been made or an accident happens. For some solutions, the evaluation will be relatively quick (an accident happens early). For others, the simulation is time-intensive (simulating until the time limit is met; no accident happens). Thus, one cannot generally order targets by their expensiveness (or as we have proposed, by information gain) but needs to implement a more sophisticated procedure. Possible research directions are dynamically updating the target's evaluation time using a book-keeping approach or another external predictor. One aspect of handling heterogeneous computing time problems is that you do not know beforehand how much computational time a solution will take to evaluate a solution. Usually, the time to compute a function may vary with a distribution. Thus, the allocation of objectives and constraints for the next evaluation may need to be done online with a quick decision at the end of each evaluation is complete. Such practicalities in scheduling will make the whole approach even more pragmatic.

Optimization is characterized by interdisciplinarity. In Chapter 2, we analyzed the literature and already provided a list of a number of application problems in different sciences with the necessity to address expensive functions. Additionally, we provided two case studies in Chapter 6. Nevertheless, this is a good starting point more, and more application problems need to be investigated. This future research direction shall help pin down commonality across research disciplines and further refine the requirements of surrogate-assisted optimization. Furthermore, because addressing expensive functions is necessary for solving real-world optimization problems, research and industry can benefit from each other pursuing collaborative optimization.

Lastly, some closing thoughts on our contributions directly to the community. Pymoo has

become a widely used tool in academia and industry. Over the last years, the framework has become a robust toolkit for (evolutionary) single- and multi-objective optimization. Many extensions and features are possible in the future, each addressing a specific problem characteristic. Moreover, the long-term view will require building a self-sustainable community to incorporate new research approaches and provide a well-maintained framework to end-users. The community should include internationally well-known optimization researchers and a group of Python developers taking care of technical matters. A network and an active community are important factors for keeping an open-source product alive. As pymoo has become one of the standard tools in Python for multi-objective optimization, we are planning to release another toolkit focusing on the optimization of expensive functions. The new development shall be based on the findings in this thesis and depend on pymoo across its optimization components. The new toolkit will directly serve as an extension for researchers and practitioners to solve computationally expensive optimization problems efficiently.

APPENDIX

SCOPUS QUERIES FOR LITERATURE ANALYSIS

In Section 3.2 we performed an analysis of literature related to surrogate-assisted algorithms. We distinguished between publications with a focus on the Problem, Method, and Goal. Their corresponding Scopus queries are listed in Algorithms A.1, A.2, A.3, respectively.

Algorithm A.1: Scopus Query: Problem

```
1 TITLE ( ( "simulation optimization" OR "simulation-based optimization"
2 OR "expensive black-box" OR "data-driven optimization") AND optimization )
3 AND ( LIMIT-TO ( LANGUAGE,"English" ) )
```

Algorithm A.2: Scopus Query: Method

```
1 TITLE ( ( "simulation optimization" OR "simulation-based optimization"
2 OR "expensive black-box" OR "data-driven optimization" OR bayesian OR "model-based"
3 OR "Kriging assisted") AND optimization )
4 AND ( LIMIT-TO ( LANGUAGE,"English" ) )
```

Algorithm A.3: Scopus Query: Goal

```
1 TITLE-ABS-KEY ( ( "Efficient Global Optimization" OR "Efficient Global Optimizer"
2 OR "anytime optimization" OR "anytime algorithm" OR "EGO") AND optimization )
3 AND ( LIMIT-TO ( LANGUAGE,"English" ) )
```

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*. USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1989.
- [2] T. A. Burrell, S. L. Campbell, C. Coomer, C. W. Ayers, A. A. Wereszczak, J. P. Cunningham, L. D. Marlino, L. E. Seiber, and H.-T. Lin, "Evaluation of the 2010 toyota prius hybrid synergy drive system," Tech. Rep. March, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States), Mar. 2011.
- [3] Y. Pochet and L. A. Wolsey, *Production planning by mixed integer programming*. Springer Science & Business Media, 2006.
- [4] M. R. Bonyadi, Z. Michalewicz, and L. Barone, "The travelling thief problem: The first step in the transition from theoretical problems to realistic problems," in *2013 IEEE congress on evolutionary computation*, pp. 1037–1044, 2013.
- [5] J. Kiefer, "Sequential minimax search for a maximum," *Proceedings of the American Mathematical Society*, vol. 4, no. 3, pp. 502–506, 1953.
- [6] K. Deb and C. Myburgh, "A Population-Based Fast Algorithm for a Billion-Dimensional Resource Allocation Problem with Integer Variables," *European Journal of Operational Research*, vol. 261, no. 2, pp. 460–474, 2017.
- [7] Y. Chauvin and D. E. Rumelhart, eds., *Backpropagation: Theory, architectures, and applications*. USA: L. Erlbaum Associates Inc., 1995.
- [8] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11 – 26, 2017.
- [9] K. Deb, *Multi-objective optimization using evolutionary algorithms*. USA: John Wiley & Sons, Inc., 2001.
- [10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Trans. Evol. Comp.*, vol. 6, pp. 182–197, Apr. 2002.
- [11] Q. Zhang and H. Li, "A multi-objective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation, Accepted*, vol. 2007, 2007.
- [12] J. Blank, K. Deb, and P. Roy, "Investigating the normalization procedure of NSGA-III," in *Evolutionary multi-criterion optimization* (K. Deb, E. Goodman, C. A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P. Reed, eds.), (Cham), pp. 229–240, Springer International Publishing, 2019.
- [13] T. Tušar and B. Filipič, "Visualization of pareto front approximations in evolutionary multiobjective optimization: A critical review and the prosecution method," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 225–245, 2015.

- [14] M. Kaisa, *Nonlinear multiobjective optimization*, vol. 12 of *International series in operations research & management science*. Boston, USA: Kluwer Academic Publishers, 1999.
- [15] D. P. Bertsekas, *Constrained optimization and lagrange multiplier methods (optimization and neural computation series)*. Athena Scientific, 1 ed., 1996.
- [16] K. Deb, “An efficient constraint handling method for genetic algorithms,” *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311–338, 2000.
- [17] J. A. Snyman and D. N. Wilke, *Practical mathematical optimization: basic optimization theory and gradient-based algorithms; 2nd ed.* Springer optimization and its applications, Cham: Springer, 2018.
- [18] C. Audet and W. Hare, *Derivative-free and blackbox optimization*. Jan. 2017.
- [19] R. Horst and P. Pardalos, *Handbook of global optimization*. Nonconvex optimization and its applications, Springer US, 2013.
- [20] J. J. Schneider and S. Kirkpatrick, *Stochastic optimization*. Berlin Heidelberg, Germany: Springer-Ver, 2006.
- [21] J. D. Anderson and J. Wendt, *Computational fluid dynamics*, vol. 206. Springer, 1995.
- [22] B. Szabó and I. Babuška, *Finite element analysis*. John Wiley & Sons, 1991.
- [23] K. Deb, R. Hussein, P. C. Roy, and G. Toscano-Pulido, “A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 104–116, 2019.
- [24] J. Blank and K. Deb, “PSAF: A Probabilistic Surrogate-Assisted Framework for Single-Objective Optimization,” in *GECCO ’21: Proceedings of the genetic and evolutionary computation conference companion*, (New York, NY, USA), ACM, 2021. place: New York, NY, USA.
- [25] J. Blank and K. Deb, “Constrained bi-objective surrogate-assisted optimization of problems with heterogeneous evaluation times: Expensive objectives and inexpensive constraints,” in *Evolutionary multi-criterion optimization* (H. Ishibuchi, Q. Zhang, R. Cheng, K. Li, H. Li, H. Wang, and A. Zhou, eds.), pp. 257–269, Springer International Publishing, 2021.
- [26] J. Blank and K. Deb, “Handling constrained multi-objective optimization problems with heterogeneous evaluation times: proof-of-principle results,” *Memetic Computing*, Mar. 2022.
- [27] J. Blank and K. Deb, “SOLVeR: A Blueprint for Collaborative Optimization in Practice,” 2021. tex.eventtitle: International Multi-Conference on Complexity, Informatics and Cybernetics: IMCIC 2021.
- [28] A. Ahrari, J. Blank, K. Deb, and X. Li, “A proximity-based surrogate-assisted method for simulation-based design optimization of a cylinder head water jacket,” *Engineering Optimization*, pp. 1–19, 2020.

- [29] J. Blank and K. Deb, “pymoo: Multi-objective Optimization in Python,” *IEEE Access*, vol. 8, pp. 89497–89509, 2020.
- [30] J. Nocedal and S. J. Wright, *Numerical optimization*. New York, NY, USA: Springer, 2 ed., 2006.
- [31] J. H. Holland, “Genetic Algorithms: Computer programs that “evolve” in ways that resemble natural selection can solve complex problems even their creators do not fully understand,” 1960.
- [32] J. H. Holland, *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press, 1975.
- [33] K. Deb and C. Myburgh, “Breaking the billion-variable barrier in real-world optimization using a customized evolutionary algorithm,” in *Proceedings of the genetic and evolutionary computation conference 2016*, GECCO ’16, (New York, NY, USA), pp. 653–660, Association for Computing Machinery, 2016.
- [34] R. L. Hardy, “Multiquadric equations of topography and other irregular surfaces,” *Journal of Geophysical Research (1896-1977)*, vol. 76, pp. 1905–1915, Mar. 1971.
- [35] D. G. Krige, “A statistical approach to some basic mine valuation problems on the Witwatersrand, by D.G. Krige, published in the Journal, December 1951 : introduction by the author,” 1951.
- [36] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning (adaptive computation and machine learning)*. The MIT Press, 2005.
- [37] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” *J. of Global Optimization*, 1998.
- [38] J. Stork, M. Friese, M. Zaefferer, T. Bartz-Beielstein, A. Fischbach, B. Breiderhoff, B. Naujoks, and T. Tušar, “Open issues in surrogate-assisted optimization,” in *High-performance simulation-based optimization* (T. Bartz-Beielstein, B. Filipič, P. Korošec, and E.-G. Talbi, eds.), pp. 225–244, Cham: Springer International Publishing, 2020.
- [39] “Scopus: Expertly curated abstract and citation database.”
- [40] *GECCO ’20: Proceedings of the 2020 genetic and evolutionary computation conference companion*. Cancún, Mexico: Association for Computing Machinery, 2020.
- [41] J. Zhang, Z. Zhan, Y. Lin, N. Chen, Y. Gong, J. Zhong, H. S. H. Chung, Y. Li, and Y. Shi, “Evolutionary computation meets machine learning: A survey,” *IEEE Computational Intelligence Magazine*, vol. 6, no. 4, pp. 68–75, 2011.
- [42] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, “Data-Driven Evolutionary Optimization: An Overview and Case Studies,” *IEEE Transactions on Evolutionary Computation*, vol. 23, pp. 442–458, June 2019.

- [43] T. W. Simpson, T. M. Mauery, J. J. Korte, and F. Mistree, “Kriging models for global approximation in simulation-based multidisciplinary design optimization,” *AIAA Journal*, vol. 39, no. 12, pp. 2233–2241, 2001.
- [44] Z. Zhou, Y.-S. Ong, M. H. Nguyen, and D. Lim, “A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm,” *2005 IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2832–2839 Vol. 3, 2005.
- [45] Y. Zhang, N. H. Kim, C. Park, and R. T. Haftka, “Multifidelity surrogate based on single linear regression,” *AIAA Journal*, vol. 56, no. 12, pp. 4944–4952, 2018.
- [46] I. Loshchilov, M. Schoenauer, and M. Sebag, “Comparison-based optimizers need comparison-based surrogates,” in *Parallel problem solving from nature, PPSN XI* (R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, eds.), (Berlin, Heidelberg), pp. 364–373, Springer Berlin Heidelberg, 2010.
- [47] A. Rosales-Pérez, J. A. Gonzalez, C. A. Coello Coello, H. J. Escalante, and C. A. Reyes-Garcia, “Surrogate-assisted multi-objective model selection for support vector machines,” *Neurocomputing*, vol. 150, pp. 163 – 172, 2015.
- [48] J. A. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.
- [49] D. Buche, N. N. Schraudolph, and P. Koumoutsakos, “Accelerating evolutionary algorithms with gaussian process fitness function models,” *Trans. Sys. Man Cyber Part C*, vol. 35, pp. 183–194, May 2005.
- [50] B. Liu, Q. Zhang, and G. G. E. Gielen, “A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 180–192, 2014.
- [51] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, “Combining global and local surrogate models to accelerate evolutionary optimization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 1, pp. 66–76, 2007.
- [52] R. Hussein and K. Deb, “A generative kriging surrogate model for constrained and unconstrained multi-objective optimization,” in *Proceedings of the genetic and evolutionary computation conference 2016, GECCO ’16*, (New York, NY, USA), pp. 573–580, Association for Computing Machinery, 2016.
- [53] A. Sinha, S. Bedi, and K. Deb, “Bilevel optimization based on kriging approximations of lower level optimal value function,” in *2018 IEEE congress on evolutionary computation (CEC)*, pp. 1–8, 2018.
- [54] J. Müller and C. A. Shoemaker, “Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems,” *Journal of Global Optimization*, vol. 60, no. 2, pp. 123–144, 2014.

- [55] S. Lophaven, H. B. Nielsen, and J. Søndergaard, “DACE – a MATLAB kriging toolbox,” 2002.
- [56] Yew-Soon Ong, P. B. Nair, and K. Y. Lum, “Max-min surrogate-assisted evolutionary algorithm for robust design,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 392–404, 2006.
- [57] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong, “Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems,” in *Knowledge Incorporation in Evolutionary Computation* (Y. Jin, ed.), pp. 307–331, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.
- [58] Y. Tenne and S. W. Armfield, “A framework for memetic optimization using variable global and local surrogate models,” *Soft Computing*, vol. 13, no. 8, p. 781, 2008.
- [59] R. G. Regis and C. A. Shoemaker, “A stochastic radial basis function method for the global optimization of expensive functions,” *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 497–509, 2007.
- [60] Z. Yang, H. Qiu, L. Gao, X. Cai, C. Jiang, and L. Chen, “Surrogate-assisted classification-collaboration differential evolution for expensive constrained optimization problems,” *Information Sciences*, vol. 508, pp. 50 – 63, 2020.
- [61] M. Farina, “A neural network based generalized response surface multiobjective evolutionary algorithm,” in *Proceedings of the 2002 congress on evolutionary computation. CEC’02 (cat. No.02TH8600)*, vol. 1, pp. 956–961 vol.1, 2002.
- [62] M. Hüsken, Y. Jin, and B. Sendhoff, “Structure optimization of neural networks for evolutionary design optimization,” *Soft Computing*, vol. 9, pp. 21–28, Jan. 2005.
- [63] Y. Jin, M. Hüsken, M. Olhofer, and B. Sendhoff, “Neural Networks for Fitness Approximation in Evolutionary Optimization,” in *Knowledge Incorporation in Evolutionary Computation* (Y. Jin, ed.), pp. 281–306, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.
- [64] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, and Y. Jin, “A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 74–88, 2019.
- [65] M. Holeňa, D. Linke, U. Rodemerck, and L. Bajer, “Neural Networks as Surrogate Models for Measurements in Optimization Algorithms,” in *Analytical and Stochastic Modeling Techniques and Applications* (K. Al-Begain, D. Fiems, and W. J. Knottenbelt, eds.), (Berlin, Heidelberg), pp. 351–366, Springer Berlin Heidelberg, 2010.
- [66] L. Niles, H. Silverman, G. Tajchman, and M. Bush, “How limited training data can allow a neural network to outperform an optimal statistical classifier,” in *International conference on acoustics, speech, and signal processing*, pp. 17–20, 1989.
- [67] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” in *Learning and intelligent optimization* (C. A. C. Coello, ed.), (Berlin, Heidelberg), pp. 507–523, Springer Berlin Heidelberg, 2011.

- [68] K. Eggensperger, M. Lindauer, H. H. Hoos, F. Hutter, and K. Leyton-Brown, “Efficient benchmarking of algorithm configurators via model-based surrogates,” *Machine Learning*, vol. 107, no. 1, pp. 15–41, 2018.
- [69] E. O. Nsoesie, R. J. Beckman, S. Shashaani, K. S. Nagaraj, and M. V. Marathe, “A simulation optimization approach to epidemic forecasting,” *PLOS ONE*, vol. 8, pp. 1–10, June 2013.
- [70] H. Wang and Y. Jin, “A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems,” *IEEE Transactions on Cybernetics*, vol. 50, no. 2, pp. 536–549, 2020.
- [71] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, “Surrogate-Assisted Evolutionary Deep Learning Using an End-to-End Random Forest-Based Performance Predictor,” *IEEE Transactions on Evolutionary Computation*, vol. 24, pp. 350–364, Apr. 2020.
- [72] L. Bajer and M. Holeňa, “Surrogate model for continuous and discrete genetic optimization based on RBF networks,” in *Intelligent data engineering and automated learning – IDEAL 2010* (C. Fyfe, P. Tino, D. Charles, C. Garcia-Osorio, and H. Yin, eds.), (Berlin, Heidelberg), pp. 251–258, Springer Berlin Heidelberg, 2010.
- [73] L. P. Swiler, P. D. Hough, P. Qian, X. Xu, C. Storlie, and H. Lee, “Surrogate models for mixed discrete-continuous variables,” in *Constraint programming and decision making* (M. Ceberio and V. Kreinovich, eds.), pp. 181–202, Cham: Springer International Publishing, 2014.
- [74] J. L. Walsh, “A closed set of normal orthogonal functions,” *American Journal of Mathematics*, vol. 45, p. 5, 1923.
- [75] S. Verel, B. Derbel, A. Liefooghe, H. Aguirre, and K. Tanaka, “A Surrogate Model Based on Walsh Decomposition for Pseudo-Boolean Functions,” in *Parallel Problem Solving from Nature – PPSN XV* (A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, eds.), (Cham), pp. 181–193, Springer International Publishing, 2018.
- [76] G. Pruvost, B. Derbel, A. Liefooghe, S. Verel, and Q. Zhang, “Surrogate-assisted multi-objective combinatorial optimization based on decomposition and walsh basis,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 542–550, Association for Computing Machinery, 2020.
- [77] F. Chicano, D. Whitley, G. Ochoa, and R. Tinós, “Optimizing one million variable NK landscapes by hybridizing deterministic recombination and local search,” in *Proceedings of the genetic and evolutionary computation conference*, GECCO ’17, (New York, NY, USA), pp. 753–760, Association for Computing Machinery, 2017.
- [78] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Proceedings of the 24th international conference on neural information processing systems*, NIPS’11, (Red Hook, NY, USA), pp. 2546–2554, Curran Associates Inc., 2011. [tex.ids= 2011-bergstra-hyper-param-opt.](#)

- [79] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi, “Multiobjective tree-structured parzen estimator for computationally expensive optimization problems,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 533–541, Association for Computing Machinery, 2020.
- [80] M. N. Le, Y. S. Ong, S. Menzel, Y. Jin, and B. Sendhoff, “Evolution by adapting surrogates,” *Evolutionary Computation*, vol. 21, no. 2, pp. 313–340, 2013.
- [81] R. Jin, W. Chen, and T. Simpson, “Comparative studies of metamodelling techniques under multiple modelling criteria,” *Structural and Multidisciplinary Optimization*, vol. 23, pp. 1–13, Dec. 2001.
- [82] D. Lim, Y.-S. Ong, Y. Jin, and B. Sendhoff, “A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation,” in *Proceedings of the 9th annual conference on genetic and evolutionary computation*, GECCO ’07, (New York, NY, USA), pp. 1288–1295, Association for Computing Machinery, 2007.
- [83] D. Lim, Y. Jin, Y. Ong, and B. Sendhoff, “Generalizing surrogate-assisted evolutionary computation,” *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 329–355, 2010.
- [84] K. S. Bhattacharjee, H. K. Singh, T. Ray, and J. Branke, “Multiple surrogate assisted multiobjective optimization using improved pre-selection,” in *2016 IEEE congress on evolutionary computation (CEC)*, 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 4328–4335, 2016.
- [85] B. S. Saini, M. Lopez-Ibanez, and K. Miettinen, “Automatic surrogate modelling technique selection based on features of optimization problems,” in *GECCO ’19: Proceedings of the genetic and evolutionary computation conference companion*, (New York, NY, USA), pp. 1765–1772, ACM, 2019.
- [86] A. Bhosekar and M. Ierapetritou, “Advances in surrogate based modeling, feasibility analysis, and optimization: A review,” *Computers & Chemical Engineering*, vol. 108, pp. 250 – 267, 2018.
- [87] T. Simpson, J. Poplinski, P. N. Koch, and J. Allen, “Metamodels for Computer-based Engineering Design: Survey and recommendations,” *Engineering with Computers*, vol. 17, pp. 129–150, July 2001.
- [88] A. I. Khuri and S. Mukhopadhyay, “Response surface methodology,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 2, pp. 128–149, 2010.
- [89] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook, *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons, 2016.
- [90] S. Kitayama, M. Arakawa, and K. Yamazaki, “Sequential approximate optimization using radial basis function network for engineering optimization,” *Optimization and Engineering*, vol. 12, no. 4, pp. 535–557, 2011.

- [91] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Advances in neural information processing systems*, pp. 2951–2959, 2012.
- [92] D. R. Jones, “A taxonomy of global optimization methods based on response surfaces,” *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.
- [93] F. Rehbach, M. Zaefferer, B. Naujoks, and T. Bartz-Beielstein, “Expected improvement versus predicted value in surrogate-based optimization,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 868–876, Association for Computing Machinery, 2020.
- [94] R. G. Regis and C. A. Shoemaker, “Constrained global optimization of expensive black box functions using radial basis functions,” *Journal of Global Optimization*, vol. 31, no. 1, pp. 153–171, 2005.
- [95] L. Wang, S. Shan, and G. Gary Wang, “Mode-pursuing sampling method for global optimization on expensive black-box functions,” *Engineering Optimization*, vol. 36, no. 4, pp. 419–438, 2004.
- [96] F. Viana and R. Haftka, “Surrogate-based optimization with parallel simulations using the probability of improvement,” in *13th AIAA/ISSMO multidisciplinary analysis optimization conference*, 2010.
- [97] A. Chaudhuri, R. T. Haftka, P. Ifju, K. Chang, C. Tyler, and T. Schmitz, “Experimental flapping wing optimization and uncertainty quantification using limited samples,” *Structural and Multidisciplinary Optimization*, pp. 957–970, Apr. 2015.
- [98] L. Yaohui, “A Kriging-based global optimization method using multi-points infill search criterion,” *Journal of Algorithms & Computational Technology*, pp. 366–377, Dec. 2017.
- [99] P. Beaucaire, C. Beauthier, and C. Sainvitu, “Multi-point infill sampling strategies exploiting multiple surrogate models,” in *GECCO ’19: Proceedings of the genetic and evolutionary computation conference companion*, (New York, NY, USA), pp. 1559–1567, ACM, 2019.
- [100] N. Berveglieri, B. Derbel, A. Liefoghe, H. Aguirre, Q. Zhang, and K. Tanaka, “Designing parallelism in surrogate-assisted multiobjective optimization based on decomposition,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 462–470, Association for Computing Machinery, 2020.
- [101] J. J. Grefenstette and J. M. Fitzpatrick, *Genetic search with approximate function evaluations*. Proceedings of an International Conference on Genetic Algorithms and Their Applications, 1985.
- [102] A. Ratle, *Accelerating the convergence of evolutionary algorithms by fitness landscape approximation*. International Conference on Parallel Problem Solving from Nature, Springer, 1998.

- [103] Y. Jin, “A comprehensive survey of fitness approximation in evolutionary computation,” *Soft Computing*, vol. 9, pp. 3–12, Jan. 2005.
- [104] Y. Jin, “Surrogate-assisted evolutionary computation: Recent advances and future challenges,” *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61 – 70, 2011.
- [105] Yaochu Jin, M. Olhofer, and B. Sendhoff, “A framework for evolutionary optimization with approximate fitness functions,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, 2002.
- [106] F. Neri and C. Cotta, “Memetic algorithms and memetic computing optimization: A literature review,” *Swarm and Evolutionary Computation*, pp. 1–14, Feb. 2012.
- [107] R. Storn and K. Price, “Differential Evolution –A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,” *Journal of Global Optimization*, pp. 341–359, Dec. 1997.
- [108] A. K. Qin, V. L. Huang, and P. N. Suganthan, “Differential evolution algorithm with strategy adaptation for global numerical optimization,” *IEEE transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2008.
- [109] X. Lu, K. Tang, and X. Yao, “Classification-assisted Differential Evolution for computationally expensive problems,” in *2011 IEEE congress of evolutionary computation (CEC)*, pp. 1986–1993, 2011.
- [110] E. Krempser, H. Bernardino, H. Barbosa, and A. Lemonge, “Differential evolution assisted by surrogate models for structural optimization problems,” vol. 100 of *Civil-Comp Proceedings*, Jan. 2012.
- [111] Y. Wang, D. Yin, S. Yang, and G. Sun, “Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints,” *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1642–1656, 2019.
- [112] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary Computation*, vol. 9, pp. 159–195, June 2001.
- [113] N. Hansen, “The CMA Evolution Strategy: A Comparing Review,” in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms* (J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, eds.), pp. 75–102, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.
- [114] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally weighted learning,” *Artificial Intelligence Review*, vol. 11, pp. 11–73, Feb. 1997.
- [115] S. Kern, N. Hansen, and P. Koumoutsakos, “Local meta-models for optimization using evolution strategies,” in *Parallel problem solving from nature - PPSN IX* (T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, eds.), (Berlin, Heidelberg), pp. 939–948, Springer Berlin Heidelberg, 2006.

- [116] I. Loshchilov, M. Schoenauer, and M. Sebag, “Intensive surrogate model exploitation in self-adaptive surrogate-assisted cma-es (saacm-es),” in *Proceedings of the 15th annual conference on genetic and evolutionary computation*, GECCO ’13, (New York, NY, USA), pp. 439–446, Association for Computing Machinery, 2013.
- [117] L. Bajer, Z. Pitra, J. Repický, and M. Holeňa, “Gaussian process surrogate models for the CMA-ES,” in *GECCO ’19: Proceedings of the genetic and evolutionary computation conference companion*, (New York, NY, USA), pp. 17–18, ACM, 2019.
- [118] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - international conference on neural networks*, vol. 4, pp. 1942–1948 vol.4, 1995.
- [119] Z. Lv, L. Wang, Z. Han, J. Zhao, and W. Wang, “Surrogate-assisted particle swarm optimization algorithm with Pareto active learning for expensive multi-objective optimization,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 838–849, 2019.
- [120] H. Wang, Y. Jin, and J. Doherty, “Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems,” *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2664–2677, 2017.
- [121] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen, “A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms,” *Soft Computing*, vol. 23, no. 9, pp. 3137–3166, 2019.
- [122] R. Allmendinger, M. T. M. Emmerich, J. Hakanen, Y. Jin, and E. Rigoni, “Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case,” *Journal of Multi-Criteria Decision Analysis*, vol. 24, no. 1-2, pp. 5–24, 2017.
- [123] J. Zhang, A. Zhou, and G. Zhang, “A classification and Pareto domination based multiobjective evolutionary algorithm,” *2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2883–2890, 2015.
- [124] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, “A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 129–142, 2018.
- [125] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, “A reference vector guided evolutionary algorithm for many-objective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.
- [126] J. Knowles, “ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.
- [127] R. E. Steuer and E.-U. Choo, “An interactive weighted Tchebycheff procedure for multiple objective programming,” *Mathematical Programming*, vol. 26, no. 3, pp. 326–344, 1983.

- [128] C. M. Cristescu and J. Knowles, “Surrogate-based multiobjective optimization : ParEGO update and test,” 2015.
- [129] M. Li, G. Li, and S. Azarm, “A kriging metamodel assisted multi-objective genetic algorithm for design optimization,” *Journal of Mechanical Design*, vol. 130, Feb. 2008.
- [130] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, “Multiobjective optimization on a limited budget of evaluations using model-assisted \mathcal{S} -Metric selection,” in *Parallel problem solving from nature –PPSN x* (G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, eds.), (Berlin, Heidelberg), pp. 784–794, Springer Berlin Heidelberg, 2008.
- [131] E. Zitzler and L. Thiele, “Multiobjective optimization using evolutionary algorithms — A comparative case study,” in *Parallel problem solving from nature — PPSN v* (A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, eds.), (Berlin, Heidelberg), pp. 292–301, Springer Berlin Heidelberg, 1998.
- [132] V. Picheny, “Multiobjective optimization using Gaussian process emulators via stepwise uncertainty reduction,” *Statistics and Computing*, vol. 25, no. 6, pp. 1265–1280, 2015.
- [133] A. A. M. Rahat, R. M. Everson, and J. E. Fieldsend, “Alternative infill strategies for expensive multi-objective optimisation,” in *Proceedings of the genetic and evolutionary computation conference, GECCO ’17*, (New York, NY, USA), pp. 873–880, Association for Computing Machinery, 2017.
- [134] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, “Expensive multiobjective optimization by MOEA/D with gaussian process model,” *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2010.
- [135] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. USA: Kluwer Academic Publishers, 1981.
- [136] A. Habib, H. K. Singh, T. Chugh, T. Ray, and K. Miettinen, “A multiple surrogate assisted decomposition-based evolutionary algorithm for expensive Multi/Many-Objective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 1000–1014, 2019.
- [137] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, “An adaptive Bayesian approach to surrogate-assisted evolutionary multi-objective optimization,” *Information Sciences*, vol. 519, pp. 317–331, 2020.
- [138] S. Bagheri, W. Konen, R. Allmendinger, J. Branke, K. Deb, J. Fieldsend, D. Quagliarella, and K. Sindhya, “Constraint handling in efficient global optimization,” in *Proceedings of the genetic and evolutionary computation conference, GECCO ’17*, (New York, NY, USA), pp. 673–680, Association for Computing Machinery, 2017.
- [139] T. Chugh, K. Sindhya, K. Miettinen, J. Hakanen, and Y. Jin, “On constraint handling in surrogate-assisted evolutionary many-objective optimization,” in *Parallel problem solving from nature – PPSN XIV* (J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, eds.), (Cham), pp. 214–224, Springer International Publishing, 2016.

- [140] R. Allmendinger and J. Knowles, “‘Hang on a minute’: Investigations on the effects of delayed objective functions in multiobjective optimization,” in *Evolutionary multi-criterion optimization* (R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, eds.), (Berlin, Heidelberg), pp. 6–20, Springer Berlin Heidelberg, 2013.
- [141] R. Allmendinger, J. Handl, and J. Knowles, “Multiobjective optimization: When objectives exhibit non-uniform latencies,” *European Journal of Operational Research*, vol. 243, no. 2, pp. 497 – 513, 2015.
- [142] T. Chugh, R. Allmendinger, V. Ojalehto, and K. Miettinen, “Surrogate-assisted evolutionary biobjective optimization for objectives with non-uniform latencies,” in *Proceedings of the genetic and evolutionary computation conference*, GECCO ’18, (New York, NY, USA), pp. 609–616, Association for Computing Machinery, 2018.
- [143] J. Thomann and G. Eichfelder, “A trust-region algorithm for heterogeneous multiobjective optimization,” *SIAM Journal on Optimization*, vol. 29, no. 2, pp. 1017–1047, 2019.
- [144] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, “Transfer learning for gaussian process assisted evolutionary bi-objective optimization for objectives with different evaluation times,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 587–594, Association for Computing Machinery, 2020.
- [145] X. Ruan, K. Li, B. Derbel, and A. Liefooghe, “Surrogate assisted evolutionary algorithm for medium scale multi-objective optimisation problems,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 560–568, Association for Computing Machinery, 2020.
- [146] F. Rehbach, L. Gentile, and T. Bartz-Beielstein, “Feature selection for surrogate model-based optimization,” in *GECCO ’19: Proceedings of the genetic and evolutionary computation conference companion*, (New York, NY, USA), pp. 399–400, ACM, 2019.
- [147] F. Rehbach, L. Gentile, and T. Bartz-Beielstein, “Variable reduction for surrogate-based optimization,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 1177–1185, Association for Computing Machinery, 2020.
- [148] J.-A. Mejía-de Dios and E. Mezura-Montes, “A surrogate-assisted metaheuristic for bilevel optimization,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 629–635, Association for Computing Machinery, 2020.
- [149] Z. Wang and M. Ierapetritou, “A Novel Surrogate-Based Optimization Method for Black-Box Simulation with Heteroscedastic Noise,” *Industrial & Engineering Chemistry Research*, vol. 56, pp. 10720–10732, Sept. 2017.
- [150] A. I. Forrester, A. Sóbester, and A. J. Keane, “Multi-fidelity optimization via surrogate modelling,” *Proceedings of the royal society a: mathematical, physical and engineering sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007.

- [151] H. Wang, Y. Jin, and J. Doherty, “A generic test suite for evolutionary multifidelity optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 6, pp. 836–850, 2018.
- [152] J. Richter, J. Shi, J.-J. Chen, J. Rahnenführer, and M. Lang, “Model-based optimization with concept drifts,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 877–885, Association for Computing Machinery, 2020.
- [153] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, “NSGA-Net: Neural architecture search using multi-objective genetic algorithm,” in *Proceedings of the genetic and evolutionary computation conference*, GECCO ’19, (New York, NY, USA), pp. 419–427, Association for Computing Machinery, 2019.
- [154] A. Fabisch, “Empirical evaluation of contextual policy search with a comparison-based surrogate model and active covariance matrix adaptation,” in *GECCO ’19: Proceedings of the genetic and evolutionary computation conference companion*, (New York, NY, USA), pp. 251–252, ACM, 2019.
- [155] O. Francon, S. Gonzalez, B. Hodjat, E. Meyerson, R. Mäkeläinen, X. Qiu, and H. Shahrzad, “Effective reinforcement learning through evolutionary surrogate-assisted prescription,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 814–822, Association for Computing Machinery, 2020.
- [156] M. Rochoux, S. Ricci, D. Lucor, B. Cuenot, and A. Trouvé, “Towards predictive data-driven simulations of wildfire spread – Part I: Reduced-cost Ensemble Kalman Filter based on a Polynomial Chaos surrogate model for parameter estimation,” *Natural Hazards and Earth System Sciences*, vol. 14, no. 11, pp. 2951–2973, 2014.
- [157] I. Gidaris, A. A. Taflanidis, and G. P. Mavroeidis, “Kriging metamodeling in seismic risk assessment based on stochastic ground motion models,” *Earthquake Engineering & Structural Dynamics*, vol. 44, pp. 2377–2399, Nov. 2015.
- [158] I. Halachmi, J. Metz, A. Van’t Land, S. Halachmi, and J. Kleijnen, “Case study: Optimal facility allocation in a robotic milking barn,” *Transactions of the American Society of Agricultural Engineers*, vol. 45, no. 5, pp. 1539–1546, 2002.
- [159] D. Broad, H. Maier, and G. Dandy, “Optimal operation of complex water distribution systems using metamodels,” *Journal of Water Resources Planning and Management*, vol. 136, no. 4, pp. 433–443, 2010.
- [160] B. Horowitz, S. M. B. Afonso, and C. V. P. de Mendonça, “Surrogate based optimal water-flooding management,” *Journal of Petroleum Science and Engineering*, vol. 112, pp. 206 – 219, 2013.
- [161] B. Ataie-Ashtiani, H. Ketabchi, and M. Rajabi, “Optimal management of a freshwater lens in a small Island using surrogate models and evolutionary algorithms,” *Journal of Hydrologic Engineering*, vol. 19, no. 2, pp. 339–354, 2014.

- [162] M. J. Asher, B. F. W. Croke, A. J. Jakeman, and L. J. M. Peeters, “A review of surrogate models and their application to groundwater modeling,” *Water Resources Research*, vol. 51, pp. 5957–5973, Aug. 2015.
- [163] C. Zheng, *Surrogate-Assisted Evolutionary Algorithms for Wind Farm Layout Optimisation Problem*. PhD thesis, University of Waikato, 2016.
- [164] H. Klie, “Physics-Based and Data-Driven Surrogates for Production Forecasting,” in *SPE-173206-MS*, (SPE), p. 18, Society of Petroleum Engineers, Feb. 2015.
- [165] S. Agada, S. Geiger, A. Elsheikh, and S. Oladyskhin, “Data-driven surrogates for rapid simulation and optimization of WAG injection in fractured carbonate reservoirs,” *Petroleum Geoscience*, vol. 23, p. 270, May 2017.
- [166] M. Hussain, A. Javadi, A. Ahangar-Asr, and R. Farmani, “A surrogate model for simulation-optimization of aquifer systems subjected to seawater intrusion,” *Journal of Hydrology*, vol. 523, pp. 542–554, 2015.
- [167] R. Sayyadi and A. Awasthi, “A simulation-based optimisation approach for identifying key determinants for sustainable transportation planning,” *International Journal of Systems Science: Operations and Logistics*, vol. 5, no. 2, pp. 161–174, 2018.
- [168] S. Deshpande, L. T. Watson, J. Shu, F. A. Kamke, and N. Ramakrishnan, “Data driven surrogate-based optimization in the problem solving environment WBCSim,” *Engineering with Computers*, vol. 27, no. 3, pp. 211–223, 2011.
- [169] V. Drouet, S. Verel, and J.-M. Do, “Surrogate-assisted asynchronous multiobjective algorithm for nuclear power plant operations,” in *Proceedings of the 2020 genetic and evolutionary computation conference*, GECCO ’20, (New York, NY, USA), pp. 1073–1081, Association for Computing Machinery, 2020.
- [170] N. O. Nikitin, P. Vychuzhanin, A. Hvatov, I. Deeva, A. V. Kalyuzhnaya, and S. V. Kovalchuk, “Deadline-driven approach for multi-fidelity surrogate-assisted environmental model calibration: SWAN wind wave model case study,” in *GECCO ’19: Proceedings of the genetic and evolutionary computation conference companion*, (New York, NY, USA), pp. 1583–1591, ACM, 2019.
- [171] F. S. Royce, J. W. Jones, and J. W. Hansen, “MODEL-BASED OPTIMIZATION OF CROP MANAGEMENT FOR CLIMATE FORECAST APPLICATIONS,” *Transactions of the ASAE*, vol. 44, no. 5, p. 1319, 2001.
- [172] P. C. Roy, A. Guber, M. Abouali, A. P. Nejadhashemi, K. Deb, and A. J. M. Smucker, “Simulation Optimization of Water Usage and Crop Yield Using Precision Irrigation,” in *Evolutionary Multi-Criterion Optimization* (K. Deb, E. Goodman, C. A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P. Reed, eds.), (Cham), pp. 695–706, Springer International Publishing, 2019.

- [173] S. Grihon, E. Burnaev, M. Belyaev, and P. Prikhodko, “Surrogate modeling of stability constraints for optimization of composite structures,” in *Surrogate-based modeling and optimization: Applications in engineering* (S. Koziel and L. Leifsson, eds.), pp. 359–391, New York, NY: Springer New York, 2013.
- [174] L. Leifsson, S. Koziel, E. Jonsson, and S. Ogurtsov, “Aerodynamic shape optimization by space mapping,” in *Surrogate-based modeling and optimization: Applications in engineering* (S. Koziel and L. Leifsson, eds.), pp. 213–245, New York, NY: Springer New York, 2013.
- [175] S. Ulaganathan and N. Asproulis, “Surrogate models for aerodynamic shape optimisation,” in *Surrogate-based modeling and optimization: Applications in engineering* (S. Koziel and L. Leifsson, eds.), pp. 285–312, New York, NY: Springer New York, 2013.
- [176] A. Amrit and L. Leifsson, “Applications of surrogate-assisted and multi-fidelity multi-objective optimization algorithms to simulation-based aerodynamic design,” *Engineering Computations (Swansea, Wales)*, vol. 37, no. 2, pp. 430–457, 2019.
- [177] A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, and V. Torczon, “Optimization Using Surrogate Objectives on a Helicopter Test Example,” in *Computational Methods for Optimal Design and Control: Proceedings of the AFOSR Workshop on Optimal Design and Control Arlington, Virginia 30 September–3 October, 1997* (J. Borggaard, J. Burns, E. Cliff, and S. Schreck, eds.), pp. 49–58, Boston, MA: Birkhäuser Boston, 1998.
- [178] F. Duchaine, T. Morel, and L. Gicquel, “Computational-fluid-dynamics-based kriging optimization tool for aeronautical combustion chambers,” *AIAA Journal*, vol. 47, no. 3, pp. 631–645, 2009.
- [179] H. Yin, H. Fang, G. Wen, Q. Wang, and Y. Xiao, “An adaptive RBF-based multi-objective optimization method for crashworthiness design of functionally graded multi-cell tube,” *Structural and Multidisciplinary Optimization*, vol. 53, no. 1, pp. 129–144, 2016.
- [180] P. Zhu, F. Pan, W. Chen, and S. Zhang, “Use of support vector regression in structural optimization: Application to vehicle crashworthiness design,” *Mathematics and Computers in Simulation*, vol. 86, pp. 21–31, 2012.
- [181] P. Zhu, Y. Zhang, and G.-L. Chen, “Metamodel-based lightweight design of an automotive front-body structure using robust optimization,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 223, no. 9, pp. 1133–1147, 2009.
- [182] A. Parnianifard, A. S. Azfanizam, M. K. A. Ariffin, and M. I. S. Ismail, “Kriging-assisted robust black-box simulation optimization in direct speed control of DC motor under uncertainty,” *IEEE Transactions on Magnetics*, vol. 54, no. 7, pp. 1–10, 2018.
- [183] E. Gengembre, B. Ladevie, O. Fudym, and A. Thuillier, “A Kriging constrained efficient global optimization approach applied to low-energy building design problems,” *Inverse Problems in Science and Engineering*, vol. 20, no. 7, pp. 1101–1114, 2012.

- [184] L. Van Gelder, P. Das, H. Janssen, and S. Roels, “Comparative study of metamodeling techniques in building energy simulation: Guidelines for practitioners,” *Simulation Modelling Practice and Theory*, vol. 49, pp. 245–257, 2014.
- [185] A.-T. Nguyen, S. Reiter, and P. Rigo, “A review on simulation-based optimization methods applied to building performance analysis,” *Applied Energy*, vol. 113, pp. 1043 – 1058, 2014.
- [186] S. Tseranidis, N. C. Brown, and C. T. Mueller, “Data-driven approximation algorithms for rapid performance evaluation and optimization of civil structures,” *Automation in Construction*, vol. 72, pp. 279–293, Dec. 2016.
- [187] Dan Guo, T. Chai, Jinliang Ding, and Y. Jin, “Small data driven evolutionary multi-objective optimization of fused magnesium furnaces,” in *2016 IEEE symposium series on computational intelligence (SSCI)*, pp. 1–8, 2016.
- [188] T. Chugh, N. Chakraborti, K. Sindhya, and Y. Jin, “A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem,” *Materials and Manufacturing Processes*, vol. 32, no. 10, pp. 1172–1178, 2017.
- [189] J. A. Easum, J. Nagar, and D. H. Werner, “Multi-objective surrogate-assisted optimization applied to patch antenna design,” in *2017 IEEE international symposium on antennas and propagation & USNC/URSI national radio science meeting*, 2017 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting, pp. 339–340, 2017.
- [190] I. Danjuma, M. Akinsolu, C. See, R. Abd-Alhameed, and B. Liu, “Design and optimization of a slotted monopole antenna for ultra-wide band body centric imaging applications,” *IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology*, vol. 4, no. 2, pp. 140–147, 2020.
- [191] J. P. Jacobs, S. Koziel, and L. Leifsson, “Bayesian support vector regression modeling of microwave structures for design applications,” in *Surrogate-based modeling and optimization: Applications in engineering* (S. Koziel and L. Leifsson, eds.), pp. 121–145, New York, NY: Springer New York, 2013.
- [192] S. Koziel, Q. S. Cheng, and J. W. Bandler, “Fast EM modeling exploiting shape-preserving response prediction and space mapping,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, no. 3, pp. 399–407, 2014.
- [193] S. Koziel, L. Leifsson, and S. Ogurtsov, “Space mapping for electromagnetic-simulation-driven design optimization,” in *Surrogate-based modeling and optimization: Applications in engineering* (S. Koziel and L. Leifsson, eds.), pp. 1–25, New York, NY: Springer New York, 2013.
- [194] A.-K. S. O. Hassan and A. S. A. Mohamed, “Surrogate-based circuit design centering,” in *Surrogate-based modeling and optimization: Applications in engineering* (S. Koziel and L. Leifsson, eds.), pp. 27–49, New York, NY: Springer New York, 2013.

- [195] J. Caballero and I. Grossmann, “An algorithm for the use of surrogate models in modular flowsheet optimization,” *AIChE Journal*, vol. 54, no. 10, pp. 2633–2650, 2008.
- [196] S. Lucidi, M. Maurici, L. Paulon, F. Rinaldi, and M. Roma, “A simulation-based multiobjective optimization approach for health care service management,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1480–1491, 2016.
- [197] H. Guo, S. Gao, K. Tsui, and T. Niu, “Simulation optimization for medical staff configuration at emergency department in hong kong,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1655–1665, 2017.
- [198] F. Zeinali, M. Mahootchi, and M. Sepehri, “Resource planning in the emergency departments: A simulation-based metamodeling approach,” *Simulation Modelling Practice and Theory*, vol. 53, pp. 123–138, 2015.
- [199] G. Deng, D. G. Fryback, and V. Kuruchittham, “Breast cancer epidemiology : calibrating simulations via optimization,” 2005.
- [200] S. Bhattarai, S. Klimov, M. Aleskandarany, H. Burrell, A. Wormall, A. Green, P. Rida, I. Ellis, R. Osan, E. Rakha, and R. Aneja, “Machine learning-based prediction of breast cancer growth rate in vivo,” *British Journal of Cancer*, vol. 121, no. 6, pp. 497–504, 2019.
- [201] C. Spanakis, E. Mathioudakis, M. Tsiknakis, N. Kampanis, and K. Marias, “Function approximation for medical image registration,” in *2018 41st international conference on telecommunications and signal processing (TSP)*, 2018 41st International Conference on Telecommunications and Signal Processing (TSP), pp. 1–5, 2018.
- [202] P. A. Romero, A. Krause, and F. H. Arnold, “Navigating the protein fitness landscape with Gaussian processes,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 110, pp. E193–E201, Jan. 2013.
- [203] L. Y. Hsieh, E. Huang, and C. Chen, “Equipment utilization enhancement in photolithography area through a dynamic system control using multi-fidelity simulation optimization with big data technique,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 2, pp. 166–175, 2017.
- [204] C. Almeder, M. Preusser, and R. F. Hartl, “Simulation and optimization of supply chains: alternative or complementary approaches?,” *OR Spectrum. Quantitative Approaches in Management*, vol. 31, no. 1, pp. 95–119, 2009.
- [205] C. Osorio and M. Bierlaire, “A surrogate model for traffic optimization of congested networks: an analytic queueing network approach,” Jan. 2009.
- [206] C. Osorio and M. Bierlaire, “A simulation-based optimization framework for urban traffic control,” tech. rep., 2010.
- [207] Y. Zhao, P. A. Ioannou, and M. M. Dessouky, “Dynamic multimodal freight routing using a co-simulation optimization approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 7, pp. 2657–2667, 2019.

- [208] W. E. Biles, J. P. C. Kleijnen, W. C. M. van Beers, and I. van Nieuwenhuyse, “Kriging metamodeling in constrained simulation optimization: an explorative study,” in *2007 Winter Simulation Conference*, pp. 355–362, Dec. 2007.
- [209] W. Ye and F. You, “A computationally efficient simulation-based optimization method with region-wise surrogate modeling for stochastic inventory management of supply chains with general network structures,” *Computers & Chemical Engineering*, vol. 87, pp. 164–179, 2016.
- [210] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, “Selection schemes in surrogate-assisted genetic programming for job shop scheduling,” in *Simulated evolution and learning* (G. Dick, W. N. Browne, P. Whigham, M. Zhang, L. T. Bui, H. Ishibuchi, Y. Jin, X. Li, Y. Shi, P. Singh, K. C. Tan, and K. Tang, eds.), (Cham), pp. 656–667, Springer International Publishing, 2014.
- [211] F. Amiri, B. Shirazi, and A. Tajdin, “Multi-objective simulation optimization for uncertain resource assignment and job sequence in automated flexible job shop,” *Applied Soft Computing Journal*, vol. 75, pp. 190–202, 2019.
- [212] M.-E. Iacob, D. Quartel, and H. Jonkers, “Capturing business strategy and value in enterprise architecture to support portfolio valuation,” pp. 11–20, 2012.
- [213] María G. Villarreal-Marroquín, Joshua D. Svenson, Fangfang Sun, Thomas J. Santner, Angela Dean, and José M. Castro, “A comparison of two metamodel-based methodologies for multiple criteria simulation optimization using an injection molding case study,” *Journal of Polymer Engineering*, vol. 33, no. 3, pp. 193–209, 2013.
- [214] T. Gabor and P. Altmann, “Benchmarking surrogate-assisted genetic recommender systems,” in *GECCO ’19: Proceedings of the genetic and evolutionary computation conference companion*, (New York, NY, USA), pp. 1568–1575, ACM, 2019.
- [215] A. Antonakis and K. Giannakoglou, “Optimisation of military aircraft engine maintenance subject to engine part shortages using asynchronous metamodel-assisted particle swarm optimisation and Monte-Carlo simulations,” *International Journal of Systems Science: Operations and Logistics*, vol. 5, no. 3, pp. 239–252, 2018.
- [216] J. Blank and K. Deb, “GPSAF: A Generalized Probabilistic Surrogate-Assisted Framework for Constrained Single- and Multi-objective Optimization,” Tech. Rep. 2022004, Computational Optimization and Innovation Laboratory, Michigan State University, East Lansing, MI-48824, USA, 2022.
- [217] W. Luo, R. Yi, B. Yang, and P. Xu, “Surrogate-assisted evolutionary framework for data-driven dynamic optimization,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 2, pp. 137–150, 2019.
- [218] S. Olafsson and J. Kim, “Simulation optimization,” in *Proceedings of the winter simulation conference*, vol. 1, pp. 79–84, 2002.

- [219] R. T. Haftka, D. Villanueva, and A. Chaudhuri, “Parallel surrogate-assisted global optimization with expensive functions –a survey,” *Structural and Multidisciplinary Optimization*, vol. 54, no. 1, pp. 3–13, 2016.
- [220] R. Storn, “On the usage of differential evolution for function optimization,” in *Proceedings of north american fuzzy information processing*, pp. 519–523, 1996.
- [221] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou, “Meta-model—Assisted evolution strategies,” in *Parallel problem solving from nature — PPSN VII* (J. J. M. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañes, eds.), (Berlin, Heidelberg), pp. 361–370, Springer Berlin Heidelberg, 2002. tex.ids= 2002-emmerich-metamodel-assisted.
- [222] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, “Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems,” *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 644–660, 2017.
- [223] X. Cai, L. Gao, and X. Li, “Efficient generalized surrogate-assisted evolutionary algorithm for high-dimensional expensive problems,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 365–379, 2020.
- [224] Y. S. Ong, P. B. Nair, and A. J. Keane, “Evolutionary optimization of computationally expensive problems via surrogate modeling,” *AIAA Journal*, vol. 41, no. 4, pp. 687–696, 2003.
- [225] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, *K-rvea: A kriging-assisted evolutionary algorithm for many-objective optimization*. Mar. 2016.
- [226] S. Bagheri, W. Konen, M. Emmerich, and T. Bäck, “Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets,” *Applied Soft Computing*, vol. 61, pp. 377 – 393, 2017.
- [227] M. M. Islam, H. K. Singh, and T. Ray, “A surrogate assisted approach for single-objective bilevel optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 681–696, 2017.
- [228] J. Müller, “MISO: mixed-integer surrogate optimization framework,” *Optimization and Engineering*, vol. 17, pp. 177–203, Mar. 2016.
- [229] J. Müller and J. D. Woodbury, “GOSAC: global optimization with surrogate approximation of constraints,” *Journal of Global Optimization*, vol. 69, pp. 117–136, Sept. 2017.
- [230] W. J. Roux, N. Stander, and R. T. Haftka, “Response surface approximations for structural optimization,” *International Journal for Numerical Methods in Engineering*, vol. 42, no. 3, pp. 517–534, 1998.
- [231] R. H. Myers, A. I. Khuri, and Walter H. Carter, “Response surface methodology: 1966–1988,” *Technometrics*, vol. 31, no. 2, pp. 137–157, 1989.

- [232] N. Hansen, “A global surrogate assisted CMA-ES,” in *Proceedings of the genetic and evolutionary computation conference*, GECCO '19, (New York, NY, USA), pp. 664–672, Association for Computing Machinery, 2019.
- [233] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, “COCO: A platform for comparing continuous optimizers in a black-box setting,” *Optimization Methods and Software*, 2020.
- [234] M. D. McKay, R. J. Beckman, and W. J. Conover, “Comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.
- [235] E. Wegman, “Hyperdimensional data analysis using parallel coordinates,” *Journal of the American Statistical Association*, vol. 85, pp. 664–675, Sept. 1990.
- [236] Z. Zhan, J. Zhang, Y. Li, and H. S. Chung, “Adaptive particle swarm optimization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.
- [237] T. G. authors, “GPyOpt: A bayesian optimization framework in python,” 2016.
- [238] G. De Ath, R. M. Everson, J. E. Fieldsend, and A. A. M. Rahat, “ ϵ -shotgun,” *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, June 2020.
- [239] M. G. Kendall, “A new measure of rank correlation,” *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.
- [240] C. A. Coello Coello and M. Reyes Sierra, “A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm,” in *MICAI 2004: Advances in artificial intelligence* (R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, and H. Sossa, eds.), (Berlin, Heidelberg), pp. 688–697, Springer Berlin Heidelberg, 2004.
- [241] E. Zitzler, D. Brockhoff, and L. Thiele, “The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators via Weighted Integration,” in *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*, EMO'07, (Berlin, Heidelberg), pp. 862–876, Springer-Verlag, 2007.
- [242] J. Blanchard, C. Beauthier, and T. Carletti, “A surrogate-assisted cooperative co-evolutionary algorithm using recursive differential grouping as decomposition strategy,” in *2019 IEEE congress on evolutionary computation (CEC)*, pp. 689–696, 2019.
- [243] G. Chen, K. Zhang, X. Xue, L. Zhang, J. Yao, H. Sun, L. Fan, and Y. Yang, “Surrogate-assisted evolutionary algorithm with dimensionality reduction method for water flooding production optimization,” *Journal of Petroleum Science and Engineering*, vol. 185, p. 106633, 2020.
- [244] F. Li, X. Cai, L. Gao, and W. Shen, “A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems,” *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1390–1402, 2021.

- [245] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, “PlatEMO: A MATLAB platform for evolutionary multi-objective optimization,” *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.
- [246] Z. Michalewicz and M. Schoenauer, “Evolutionary algorithms for constrained parameter optimization problems,” *Evolutionary Computation*, vol. 4, pp. 1–32, Mar. 1996.
- [247] C. A. Floudas and P. M. Pardalos, “A collection of test problems for constrained global optimization algorithms,” in *Lecture notes in computer science*, 1990.
- [248] *IEEE international conference on evolutionary computation, CEC 2006, part of WCCI 2006, vancouver, BC, canada, 16-21 july 2006*. IEEE, 2006.
- [249] J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. Coello, and K. Deb, “Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization,” *Nanyang Technological University, Singapore, Tech. Rep*, vol. 41, Jan. 2006.
- [250] T. Runarsson and X. Yao, “Search biases in constrained evolutionary optimization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 2, pp. 233–243, 2005.
- [251] T. Runarsson and X. Yao, “Stochastic ranking for constrained evolutionary optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.
- [252] R Core Team, “R: A language and environment for statistical computing,” manual, Vienna, Austria, 2020.
- [253] E. Zitzler, K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: Empirical results,” *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.
- [254] S. Huband, L. Barone, L. While, and P. Hingston, “A Scalable Multi-objective Test Problem Toolkit,” in *Evolutionary Multi-Criterion Optimization* (C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, eds.), (Berlin, Heidelberg), pp. 280–295, Springer Berlin Heidelberg, 2005.
- [255] N. Beume, B. Naujoks, and M. Emmerich, “SMS-EMOA: Multiobjective selection based on dominated hypervolume,” *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.
- [256] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength pareto evolutionary algorithm,” 2001.
- [257] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable test problems for evolutionary multiobjective optimization,” in *Evolutionary multiobjective optimization: Theoretical advances and applications* (A. Abraham, L. Jain, and R. Goldberg, eds.), pp. 105–145, London: Springer London, 2005.

- [258] H. Jain and K. Deb, “An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach,” *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 602–622, Aug. 2014.
- [259] T. T. Binh and U. Korn, “MOBES: A multiobjective evolution strategy for constrained optimization problems,” in *In Proceedings of the Third International Conference on Genetic Algorithms*, pp. 176–182, 1997.
- [260] N. Srinivas and K. Deb, “Multi-Objective function optimization using non-dominated sorting genetic algorithms,” *Evolutionary Computation Journal*, vol. 2, no. 3, pp. 221–248, 1994.
- [261] M. Tanaka, “GA-based decision support system for multi-criteria optimization,” in *Proceedings of the international conference on systems, man and cybernetics*, vol. 2, pp. 1556–1561, 1995.
- [262] A. Osyczka and S. Kundu, “A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm,” *Structural optimization*, vol. 10, pp. 94–99, Oct. 1995.
- [263] E. Mezura-Montes and C. A. C. Coello, “Constraint-handling in nature-inspired numerical optimization: past, present and future,” *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.
- [264] A. I. Forrester and A. J. Keane, “Recent advances in surrogate-based optimization,” *Progress in Aerospace Sciences*, vol. 45, no. 1, pp. 50 – 79, 2009.
- [265] J. Thomann and G. Eichfelder, “Representation of the Pareto Front for heterogeneous multi-objective optimization,” vol. 1, pp. 293–323, Dec. 2019.
- [266] X. Wang, Y. Jin, S. Schmitt, M. Olhofer, and R. Allmendinger, “Transfer learning based surrogate assisted evolutionary bi-objective optimization for objectives with different evaluation times,” *Knowledge-Based Systems*, vol. 227, p. 107190, 2021.
- [267] R. Allmendinger and J. Knowles, “Heterogeneous objectives: State-of-the-art and future research,” 2021.
- [268] K. H. Rahi, H. K. Singh, and T. Ray, “Investigating the use of sequencing and infeasibility driven strategies for constrained optimization,” in *2019 IEEE congress on evolutionary computation (CEC)*, pp. 1642–1649, 2019.
- [269] K. H. Rahi, H. K. Singh, and T. Ray, “Feasibility-ratio based sequencing for computationally efficient constrained optimization,” *Swarm and Evolutionary Computation*, vol. 62, p. 100850, 2021.
- [270] K. H. Rahi, H. K. Singh, and T. Ray, “Partial evaluation strategies for expensive evolutionary constrained optimization,” *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2021.

- [271] A. I. Forrester, A. Sóbester, and A. J. Keane, “Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007.
- [272] D. Brockhoff and E. Zitzler, “Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization,” in *Parallel problem solving from nature - PPSN IX* (T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, eds.), (Berlin, Heidelberg), pp. 533–542, Springer Berlin Heidelberg, 2006.
- [273] G. Batista and Maria Carolina Monard, “An analysis of four missing data treatment methods for supervised learning,” *Applied Artificial Intelligence*, vol. 17, no. 5-6, pp. 519–533, 2003.
- [274] A. Pétrowski, “A Clearing Procedure as a Niching Method for Genetic Algorithms,” in *IEEE 3rd ICEC’96*, pp. 798–803, 1996.
- [275] D. Hardin and E. Saff, “Minimal Riesz energy point configurations for rectifiable d-dimensional manifolds,” *Advances in Mathematics*, vol. 193, no. 1, pp. 174 – 204, 2005.
- [276] J. Blank, K. Deb, Y. Dhebar, S. Bandaru, and H. Seada, “Generating Well-Spaced Points on a Unit Simplex for Evolutionary Many-Objective Optimization,” *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2020.
- [277] D. Eriksson, D. Bindel, and C. A. Shoemaker, “pySOT and POAP: An event-driven asynchronous framework for surrogate optimization,” *arXiv preprint arXiv:1908.00420*, 2019.
- [278] K. Deb, A. Pratap, and T. Meyarivan, “Constrained test problems for multi-objective evolutionary optimization,” in *Evolutionary multi-criterion optimization* (E. Zitzler, L. Thiele, K. Deb, C. A. Coello Coello, and D. Corne, eds.), (Berlin, Heidelberg), pp. 284–298, Springer Berlin Heidelberg, 2001.
- [279] K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints,” *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.
- [280] I. Das and J. E. Dennis, “Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems,” *SIAM J. on Optimization*, vol. 8, pp. 631–657, Mar. 1998.
- [281] K. Deb and M. Goyal, “Optimizing engineering designs using a combined genetic search,” in *PROCEEDINGS OF THE SIXTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS*, pp. 521–528, Morgan Kauffman Publishers, 1995.
- [282] B. Khoshoo, J. Blank, T. Pham, K. Deb, and S. Foster, “Optimized electric machine design solutions with efficient handling of constraints,” in *2021 IEEE symposium series on computational intelligence (SSCI)*, pp. 1–8, 2021.
- [283] N. V. Sahinidis, “Optimization under uncertainty: state-of-the-art and opportunities,” *Computers & Chemical Engineering*, vol. 28, no. 6-7, pp. 971–983, 2004.

- [284] X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht, “Seeking multiple solutions: an updated survey on niching methods and their applications,” *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 518–538, 2016.
- [285] T. Bartz-Beielstein, M. Preuss, K. Schmitt, and H.-P. Schwefel, “Challenges for contemporary evolutionary algorithms,” 2010.
- [286] N. Hansen, A. Auger, S. Finck, and R. Ros, “Real-parameter black-box optimization benchmarking 2010: Experimental setup,” Research report RR-7215, INRIA, Mar. 2010.
- [287] S. Koziel, L. Leifsson, and X.-S. Yang, *Solving computationally expensive engineering problems: methods and applications*, vol. 97. Springer, 2014.
- [288] V. R. Joseph, “Space-filling designs for computer experiments: A review,” *Quality Engineering*, vol. 28, no. 1, pp. 28–35, 2016.
- [289] A. Ahrari, K. Deb, and M. Preuss, “Multimodal optimization by covariance matrix self-adaptation evolution strategy with repelling subpopulations,” *Evolutionary computation*, vol. 25, no. 3, pp. 439–471, 2017.
- [290] A. Sóbester, S. J. Leary, and A. J. Keane, “A parallel updating scheme for approximating and optimizing high fidelity computer simulations,” *Structural and multidisciplinary optimization*, vol. 27, no. 5, pp. 371–383, 2004.
- [291] D. Zhan, J. Qian, and Y. Cheng, “Balancing global and local search in parallel efficient global optimization algorithms,” *Journal of Global Optimization*, vol. 67, no. 4, pp. 873–892, 2017.
- [292] N. M. Alexandrov, J. Dennis, R. M. Lewis, and V. Torczon, “A trust-region framework for managing the use of approximation models in optimization,” *Structural optimization*, vol. 15, no. 1, pp. 16–23, 1998.
- [293] Y. Jin, M. Hüsken, and B. Sendhoff, “Quality measures for approximate models in evolutionary computation,” in *GECCO*, pp. 170–173, 2003.
- [294] A. Auger and N. Hansen, “A restart CMA evolution strategy with increasing population size,” in *2005 IEEE congress on evolutionary computation*, vol. 2, pp. 1769–1776 Vol. 2, 2005.
- [295] H. Ishibuchi, R. Imada, N. Masuyama, and Y. Nojima, “Comparison of hypervolume, IGD and IGD+ from the viewpoint of optimal distributions of solutions,” in *International conference on evolutionary multi-criterion optimization*, pp. 332–345, 2019.
- [296] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, “How to specify a reference point in hypervolume calculation for fair performance comparison,” *Evolutionary computation*, vol. 26, no. 3, pp. 411–440, 2018.
- [297] J. Müller, “SOCOMO: Surrogate optimization of computationally expensive multiobjective problems,” *INFORMS Journal on Computing*, vol. 29, no. 4, pp. 581–596, 2017.
- [298] J. Müller, “Codes for Surrogate Model Based Optimization,” 2020.

- [299] R. Ramarathnam, B. G. Desai, and V. S. Rao, "A comparative study of minimization techniques for optimization of induction motor design," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-92, no. 5, pp. 1448–1454, 1973.
- [300] B. Singh, B. P. Singh, S. S. Murthy, and C. S. Jha, "Experience in design optimization of induction motor using 'SUMT' algorithm," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-102, no. 10, pp. 3379–3384, 1983.
- [301] N. Bianchi and S. Bolognani, "Design optimisation of electric motors by genetic algorithms," *IEE Proceedings - Electric Power Applications*, vol. 145, pp. 475–483(8), Sept. 1998. tex.copyright: © IEE.
- [302] B. Mirzaeian, M. Moallem, V. Tahani, and C. Lucas, "Multiobjective optimization method based on a genetic algorithm for switched reluctance motor design," *IEEE Transactions on Magnetics*, vol. 38, no. 3, pp. 1524–1527, 2002.
- [303] S. D. Sudhoff, J. Cale, B. Cassimere, and M. Swinney, "Genetic algorithm based design of a permanent magnet synchronous machine," *2005 IEEE International Conference on Electric Machines and Drives*, pp. 1011–1019, 2005.
- [304] D. Zarko, D. Ban, and T. Lipo, "Design optimization of interior permanent magnet (IPM) motors with maximized torque output in the entire speed range," in *2005 european conference on power electronics and applications*, pp. 10 pp.–P.10, 2005.
- [305] Y. Duan, R. G. Harley, and T. G. Habetler, "Comparison of particle swarm optimization and genetic algorithm in the design of permanent magnet motors," *2009 IEEE 6th International Power Electronics and Motion Control Conference, IPEMC '09*, vol. 3, pp. 822–825, 2009.
- [306] Y. Duan and D. M. Ionel, "A review of recent developments in electrical machine design optimization methods with a permanent-magnet synchronous motor benchmark study," *IEEE Transactions on Industry Applications*, vol. 49, no. 3, pp. 1268–1275, 2013.
- [307] P. Zhang, G. Y. Sizov, D. M. Ionel, and N. A. Demerdash, "Design optimization of spoke-type ferrite magnet machines by combined design of experiments and differential evolution algorithms," in *2013 international electric machines drives conference*, pp. 892–898, 2013.
- [308] G. Pellegrino and F. Cupertino, "FEA-based multi-objective optimization of IPM motor design including rotor losses," in *2010 IEEE energy conversion congress and exposition*, pp. 3659–3666, 2010.
- [309] G. Pellegrino and F. Cupertino, "IPM motor rotor design by means of FEA-based multi-objective optimization," in *2010 IEEE international symposium on industrial electronics*, pp. 1340–1346, 2010.
- [310] L. Jolly, M. Jabbar, and L. Qinghua, "Design optimization of permanent magnet motors using response surface methodology and genetic algorithms," *IEEE Transactions on Magnetics*, vol. 41, no. 10, pp. 3928–3930, 2005.

- [311] D. M. Ionel and M. Popescu, “Finite element surrogate model for electric machines with revolving field — application to IPM motors,” in *2009 IEEE energy conversion congress and exposition*, pp. 178–186, 2009.
- [312] G. Y. Sizov, D. M. Ionel, and N. A. O. Demerdash, “Modeling and parametric design of permanent-magnet AC machines using computationally efficient finite-element analysis,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 6, pp. 2403–2413, 2012.
- [313] N. Taran, D. M. Ionel, and D. G. Dorrell, “Two-level surrogate-assisted differential evolution multi-objective optimization of electric machines using 3-D FEA,” *IEEE Transactions on Magnetics*, vol. 54, no. 11, pp. 1–5, 2018.
- [314] J. Song, J. Zhao, F. Dong, J. Zhao, Z. Qian, and Q. Zhang, “A novel regression modeling method for PMSLM structural design optimization using a Distance-Weighted KNN Algorithm,” *IEEE Transactions on Industry Applications*, vol. 54, no. 5, pp. 4198–4206, 2018.
- [315] G. Pellegrino, F. Cupertino, and C. Gerada, “Automatic design of synchronous reluctance motors focusing on barrier shape optimization,” *IEEE Transactions on Industry Applications*, vol. 51, no. 2, pp. 1465–1474, 2015.
- [316] G. Bramerdorfer, J. A. Tapia, J. J. Pyrhönen, and A. Cavagnino, “Modern electrical machine design optimization: Techniques, Trends, and Best Practices,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 7672–7684, 2018.
- [317] S. Stipetic, W. Miebach, and D. Zarko, “Optimization in design of electric machines: Methodology and workflow,” in *2015 intl aegean conference on electrical machines power electronics (ACEMP), 2015 intl conference on optimization of electrical electronic equipment (OPTIM) 2015 intl symposium on advanced electromechanical motion systems (ELECTRO-MOTION)*, pp. 441–448, 2015.
- [318] F. Gillon and P. Brochet, “Screening and response surface method applied to the numerical optimization of electromagnetic devices,” *IEEE Transactions on Magnetics*, vol. 36, no. 4, pp. 1163–1167, 2000.
- [319] Altair, “Altair FluxMotor (version 2019.1.1),” manual, Altair Engineering Inc., 2019.
- [320] J. Pyrhönen, T. Jokinen, and V. Hrabovcová, “Design of rotating electrical machines,” in *Design of rotating electrical machines*, pp. 1–512, 2008.
- [321] E. J. Cramer, J. E. Dennis, Jr, P. D. Frank, R. M. Lewis, and G. R. Shubin, “Problem formulation for multidisciplinary optimization,” *SIAM Journal on Optimization*, vol. 4, no. 4, pp. 754–776, 1994.
- [322] Y. Parte, D. Auroux, J. Clément, M. Masmoudi, and J. Hermetz, “Collaborative optimization,” *Multidisciplinary design optimization in computational mechanics*, pp. 321–368, Wiley, 2010.

- [323] R. Braun, A. Moore, and I. Kroo, “Use of the collaborative optimization architecture for launch vehicle design,” in *6th symposium on multidisciplinary analysis and optimization*, 1996.
- [324] Collaboration, *The cambridge dictionary of philosophy*. Cambridge University Press, 1999.
- [325] “Why is collaboration so difficult?,” 2017.
- [326] R. Atkinson, “Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria,” *International journal of project management*, vol. 17, no. 6, pp. 337–342, 1999.
- [327] A. De Wit, “Measurement of project success,” *International journal of project management*, vol. 6, no. 3, pp. 164–170, 1988.
- [328] J. R. Adams and S. E. Barnd, “Behavioral Implications of the Project Life Cycle,” in *Project Management Handbook*, pp. 206–230, John Wiley & Sons, Ltd, 1997.
- [329] J. Highsmith, *Agile project management: creating innovative products*. Pearson education, 2009.
- [330] G. D. Brewer, “The challenges of interdisciplinarity,” *Policy sciences*, vol. 32, no. 4, pp. 327–337, 1999.
- [331] S. Lélé and R. B. Norgaard, “Practicing interdisciplinarity,” *BioScience*, vol. 55, no. 11, pp. 967–975, 2005.
- [332] J. A. Jacobs and S. Frickel, “Interdisciplinarity: A Critical Assessment,” *Annual Review of Sociology*, vol. 35, no. 1, pp. 43–65, 2009.
- [333] R. T. Craig, “Communication in the Conversation of Disciplines,” *Russian Journal of Communication*, vol. 1, no. 1, pp. 7–23, 2008.
- [334] J. D. Peters, *Speaking into the air : a history of the idea of communication*. Chicago : University of Chicago Press, 1999., 1999.
- [335] E. Curry, “The big data value chain: Definitions, concepts, and theoretical approaches,” in *New horizons for a data-driven economy* (J. M. Cavanillas, E. Curry, and W. Wahlster, eds.), pp. 29–37, Springer, 2016.
- [336] A. Gaur, A. K. Talukder, K. Deb, S. Tiwari, S. Xu, and D. Jones, “Unconventional optimization for achieving well-informed design solutions for the automobile industry,” *Engineering Optimization*, vol. 52, no. 9, pp. 1542–1560, 2020.
- [337] G. Rossum, “Python Reference Manual,” tech. rep., CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, The Netherlands, 1995.
- [338] M. Bücker, G. Corliss, P. Hovland, U. Naumann, and B. Norris, *Automatic Differentiation: Applications, Theory, and Implementations (Lecture Notes in Computational Science and Engineering)*. Berlin, Heidelberg: Springer-Verlag, 2006.

- [339] R. Lehmann, *Sphinx Documentation*. Universitaet Potsdam, 2019.
- [340] A. Pajankar, *Python Unit Test Automation: Practical Techniques for Python Developers and Testers*. Berkely, CA, USA: Apress, 1st ed., 2017.
- [341] J. J. Durillo and A. J. Nebro, “jMetal: A Java framework for multi-objective optimization,” *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011.
- [342] J. Gosling, B. Joy, G. L. Steele, G. Bracha, and A. Buckley, *The Java Language Specification, Java SE 8 Edition*. Addison-Wesley Professional, 1st ed., 2014.
- [343] A. Benítez-Hidalgo, A. J. Nebro, J. García-Nieto, I. Oregi, and J. D. Ser, “jMetalPy: A Python framework for multi-objective optimization with metaheuristics,” *Swarm and Evolutionary Computation*, vol. 51, p. 100598, 2019.
- [344] D. Izzo, “PyGMO and PyKEP: open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization),” in *5th International Conference on Astrodynamics Tools and Techniques (ICATT 2012)*, 2012.
- [345] D. Hadka, *Platypus: Multiobjective Optimization in Python*.
- [346] F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, “DEAP: Evolutionary Algorithms Made Easy,” *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, July 2012.
- [347] A. Garrett, *inspyred: Python library for bio-inspired computational intelligence*.
- [348] D. Hadka, *MOEA Framework: A free and open source Java framework for multiobjective optimization*.
- [349] E. López-Camacho, M. J. García-Godoy, A. J. Nebro, and J. F. A. Montes, “jMetalCpp: optimizing molecular docking problems with a C++ metaheuristic framework,” *Bioinformatics*, vol. 30, no. 3, pp. 437–438, 2014.
- [350] K. Deb and M. Abouhawwash, “An optimality theory-based proximity measure for set-based multiobjective optimization,” *IEEE Trans. Evolutionary Computation*, vol. 20, no. 4, pp. 515–528, 2016.
- [351] K. Deb, M. Abouhawwash, and H. Seada, “A computationally fast convergence measure and implementation for single-, multiple-, and many-objective optimization,” *IEEE Trans. Emerging Topics in Comput. Intellig.*, vol. 1, no. 4, pp. 280–293, 2017.
- [352] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *Trans. Evol. Comp.*, vol. 1, pp. 67–82, Apr. 1997.
- [353] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” 2017.
- [354] Dask Development Team, *Dask: Library for dynamic task scheduling*. 2016.

- [355] S. Mishra, S. Mondal, and S. Saha, “Fast implementation of steady-state NSGA-II,” in *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 3777–3784, July 2016.
- [356] J. C. Bean, “Genetic Algorithms and Random Keys for Sequencing and Optimization,” *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, 1994.
- [357] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *Computer Journal*, vol. 7, pp. 308–313, 1965.
- [358] K. Deb and J. Sundar, “Reference point based multi-objective optimization using evolutionary algorithms,” in *Proceedings of the 8th annual conference on genetic and evolutionary computation, GECCO '06*, (New York, NY, USA), pp. 635–642, ACM, 2006.
- [359] H. Seada and K. Deb, “A unified evolutionary optimization procedure for single, multiple, and many objectives,” *IEEE Transactions on Evolutionary Computation*, vol. 20, pp. 358–369, June 2016.
- [360] Y. Vesikar, K. Deb, and J. Blank, “Reference point based NSGA-III for preferred solutions,” in *2018 IEEE symposium series on computational intelligence (SSCI)*, pp. 1587–1594, Nov. 2018.
- [361] K. Deb, K. Sindhya, and T. Okabe, “Self-adaptive simulated binary crossover for real-parameter optimization,” in *Proceedings of the 9th annual conference on genetic and evolutionary computation, GECCO '07*, (New York, NY, USA), pp. 1187–1194, ACM, 2007.
- [362] K. Deb and D. Deb, “Analysing mutation schemes for real-parameter genetic algorithms,” *International Journal of Artificial Intelligence and Soft Computing*, vol. 4, no. 1, pp. 1–28, 2014.
- [363] K. Deb and M. Goyal, “A robust optimization procedure for mechanical component design based on genetic adaptive search,” *Transactions of the ASME: Journal of Mechanical Design*, vol. 120, no. 2, pp. 162–164, 1998.
- [364] J. Blank and K. Deb, “A Running Performance Metric and Termination Criterion for Evaluating Evolutionary Multi- and Many-objective Optimization Algorithms,” in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.
- [365] A. Santiago, H. J. F. Huacuja, B. Dorronsoro, J. E. Pecero, C. G. Santillan, J. J. G. Barbosa, and J. C. S. Monterrubio, “A Survey of Decomposition Methods for Multi-objective Optimization,” in *Recent Advances on Hybrid Approaches for Designing Intelligent Systems* (O. Castillo, P. Melin, W. Pedrycz, and J. Kacprzyk, eds.), pp. 453–465, Cham: Springer International Publishing, 2014.
- [366] A. P. Wierzbicki, “The use of reference objectives in multiobjective optimization,” in *Multiple criteria decision making theory and application*, pp. 468–486, Springer, 1980.
- [367] A. P. Wierzbicki, “A mathematical basis for satisficing decision making,” *Mathematical Modelling*, vol. 3, no. 5, pp. 391 – 405, 1982.

- [368] J. Knowles and D. Corne, “On Metrics for Comparing Non-Dominated Sets,” in *Proceedings of the 2002 Congress on Evolutionary Computation Conference (CEC02)*, (United States), pp. 711–716, Institute of Electrical and Electronics Engineers, 2002.
- [369] D. A. V. Veldhuizen, “Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations,” tech. rep., Evolutionary Computation, 1999.
- [370] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, “Modified distance calculation in generational distance and inverted generational distance,” in *Evolutionary multi-criterion optimization* (A. Gaspar-Cunha, C. Henggeler Antunes, and C. C. Coello, eds.), (Cham), pp. 110–125, Springer International Publishing, 2015.
- [371] E. Zitzler and L. Thiele, “Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study,” in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, PPSN V, (London, UK, UK), pp. 292–304, Springer-Verlag, 1998.
- [372] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [373] P. Hoffman, G. Grinstein, and D. Pinkney, “Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations,” in *Proc. Workshop on new Paradigms in Information Visualization and Manipulation in conjunction with the ACM International Conference on Information and Knowledge Management (NPIVM99)*, (New York, NY, USA), pp. 9–16, ACM, 1999.
- [374] E. Kandogan, “Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions,” in *In proceedings of the IEEE information visualization symposium, late breaking hot topics*, pp. 9–12, 2000.
- [375] A. Pryke, S. Mostaghim, and A. Nazemi, “Heatmap visualization of population based multi objective algorithms,” in *Evolutionary multi-criterion optimization* (S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, eds.), (Berlin, Heidelberg), pp. 361–375, Springer Berlin Heidelberg, 2007.
- [376] Y. S. Tan and N. M. Fraser, “The modified star graph and the petal diagram: two new visual aids for discrete alternative multicriteria decision making,” *Journal of Multi-Criteria Decision Analysis*, vol. 7, no. 1, pp. 20–33, 1998.
- [377] E. Kasanen, R. Östermark, and M. Zeleny, “Gestalt system of holistic graphics: New management support view of MCDM,” *Computers & OR*, vol. 18, no. 2, pp. 233–239, 1991.
- [378] A. K. A. Talukder and K. Deb, “PaletteViz: A Visualization Method for Functional Understanding of High-Dimensional Pareto-Optimal Data-Sets to Aid Multi-Criteria Decision Making,” *IEEE Computational Intelligence Magazine*, vol. 15, no. 2, pp. 36–48, 2020.
- [379] L. Rachmawati and D. Srinivasan, “Multiobjective evolutionary algorithm with controllable focus on the knees of the pareto front,” *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 810–824, Aug. 2009.

- [380] K. Deb, A. Sinha, P. Korhonen, and J. Wallenius, “An Interactive Evolutionary Multi-Objective Optimization Method Based on Progressively Approximated Value Functions,” *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 723–739, 2010.