Universiteit
Leiden
The Netherlands

Efficient constraint multi-objective optimization with applications in ship design

Winter, R. de

Citation
Winter, R. de. (2024, October 8). *Efficient constraint multi-objective optimization with applications in ship design*. Retrieved from https://hdl.handle.net/1887/4094606

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/4094606

Note: To cite this publication please use the final published version (if applicable).

Efficient Constraint Multi-Objective Optimization with Applications in Ship Design

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op dinsdag 8 oktober 2024
klokke 16.00 uur

door

Roy de Winter

geboren te Nisse, Nederland

in 1994

Promotor:

Prof.dr. T.H.W. Bäck

Co-promotor:

Dr. N. van Stein

Promotiecommissie:

Prof.dr. M.M. Bonsangue
Prof.dr. A. Plaat
Dr. H. Wang
Dr. Y. Fan
Prof.dr. K. Deb (Michigan State University, USA)
Prof.dr. C.A.C. Coello (National Polytechnic Institute, Mexico)

Copyright © 2024 Roy de Winter All rights reserved.

This dissertation was made possible through the support of C-Job Naval Architects which provided me with a full-time research and development position, thereby facilitating my research and studies as a guest PhD candidate at the Leiden Institute Of Advanced Computer Science, Leiden University. This arrangement was essential in enabling the completion of my doctoral studies without grant funding.

Cover by: Jelle van de Ridder www.jellevanderidder.com/

Contents

1 Introduction **1**

1.1 Research Questions . . . 3
1.2 Outline . . . 3
1.3 Publications of this thesis . . . 4
1.4 Other Work by the Author . . . 5

2 Preliminaries **7**

2.1 Acronyms . . . 7
2.1.1 Optimization Acronyms . . . 7
2.1.2 Ship Design Acronyms . . . 8
2.2 Problem Notations . . . 8
2.2.1 Pareto Optimal Solutions . . . 10
2.3 Expensive Black Box Optimization . . . 11
2.3.1 Design of Experiments . . . 11
2.3.2 Surrogate Models . . . 12
2.3.3 Local Search Methodologies For Surrogate Exploration . . . 17
2.4 Benchmark Test Functions . . . 18
2.5 Algorithm Performance Metrics . . . 19
2.5.1 Multi-Objective Performance Metrics . . . 19
2.5.2 Optimization Result Visualizations . . . 21

3 Ship Design Problem Characteristics **29**

3.1 Introduction . . . 29
3.1.1 Empirical Design Method . . . 31
3.1.2 Simulated Design Method . . . 31
3.1.3 Radical Design Choices . . . 32

3.2 Holistic Ship Design Optimization . . . 32
3.2.1 Ship Design and Evaluation Software . . . 33
3.2.2 Optimization Problem Definition Guidelines . . . 37
3.2.3 Optimization Framework . . . 39
3.3 Illustrative Ship Design Optimization Problem . . . 39
3.3.1 Parametric Geometry . . . 40
3.3.2 Ship Design Constraints . . . 42
3.3.3 Ship Design Objectives . . . 45
3.3.4 Optimization of Ship Design Problem . . . 46
3.4 Conclusions and Future Work . . . 46

4 Empirical Design Optimization Approach 49
4.1 Introduction . . . 50
4.2 Data Description for Reference Studies . . . 50
4.2.1 Visualizations . . . 51
4.2.2 Data Pre-processing . . . 52
4.3 New Empirical Design Methodology . . . 54
4.3.1 Setup Design Challenge . . . 54
4.3.2 Random Forest Regression . . . 55
4.3.3 Isolation Forest . . . 56
4.3.4 Design Problem Optimization . . . 57
4.4 Empirical Design Experiments . . . 58
4.4.1 Random Forest Regression Experiment . . . 58
4.4.2 Isolation Forest Experiment . . . 60
4.4.3 NSGA-II Experiment . . . 60
4.5 Discussion . . . 62
4.6 Conclusion and Future Work . . . 63

5 Multi Objective Simulation Based Optimization 65
5.1 Constraint Multi-Objective Optimization . . . 66
5.1.1 Related Work . . . 67
5.1.2 SAMO-COBRA . . . 69
5.1.3 Multi-Objective Optimization Experiments . . . 75
5.1.4 Results . . . 77
5.1.5 Discussion $\mathcal{P}_{hv}$ vs. $\mathcal{S}$-metric Infill Criterion . . . 81
5.1.6 Conclusion and Future Work on Multi-Objective Optimization 82
5.2 Parallel Multi-Objective Optimization . . . 83

5.2.1 Related Work . . . 85
5.2.2 Multi-Point Acquisition function . . . 88
5.2.3 Multi-Point Acqusition Function Experiments . . . 92
5.2.4 Results . . . 93
5.2.5 Discussion on Parallelization . . . 97
5.2.6 Conclusion and Future Work on Parallel Optimization . . . 98
5.3 Expensive and Inexpensive Function Optimization . . . 99
5.3.1 Related Work . . . 100
5.3.2 Inexpensive Function Exploitation . . . 102
5.3.3 IOC-SAMO-COBRA Experiments . . . 104
5.3.4 Results . . . 105
5.3.5 Discussion of inexpensive function use . . . 112
5.3.6 Conclusion and Future Work on Inexpensive Function Exploitation 113
5.4 Overall Conclusions and Future Work . . . 114

6 Real World Applications 115
6.1 Trailing Suction Hopper Dredger . . . 115
6.1.1 Results of Trailing Suction Hopper Dredger Design Experiment 116
6.1.2 Analysis of the TSHD Results . . . 116
6.1.3 Conclusion from Trailing Suction Hopper Dredger Study . . . 117
6.2 Wind Feeder Vessel . . . 118
6.2.1 Results of Wind Feeder Design Experiment . . . 120
6.2.2 Analysis of Wind Feeder Optimization Results . . . 122
6.2.3 Conclusion for Wind Feeder Vessel . . . 122
6.3 Single Hold Cargo Ship Damage Stability Optimization . . . 122
6.3.1 Results of Cargo Vessel Design Experiment . . . 124
6.3.2 Analysis of Cargo Vessel Design Results . . . 125
6.3.3 Conclusion Cargo Vessel . . . 126
6.4 Roll-on/Roll-off Ferry Hull Optimization . . . 126
6.4.1 Results of Ferry Hull Optimization Experiment . . . 128
6.4.2 Analysis of Ferry Hull Results . . . 128
6.4.3 Conclusion on Ferry Hull Optimization . . . 130
6.5 Bulb Optimization Problem . . . 131
6.5.1 Results of Bulb Design Experiment . . . 132
6.5.2 Analysis of Bulb Design Results . . . 133
6.5.3 Conclusion on Bulb Optimization . . . 133

6.6 Real World Optimization Conclusions and Future Work 134

7 Conclusions and Future Work **135**
7.1 Summary 135
7.2 Conclusions 137
7.3 Future Work 138

A Appendix **139**
A.1 Empirical Attainment Difference Functions 139

B Appendix **147**
B.1 Expensive Single Objective Optimization 147
B.2 Modular Optimization Framework 147
B.2.1 Input parameters 147
B.2.2 Design of Experiments 148
B.2.3 Evaluation of the solutions 149
B.2.4 Radial Basis Functions 151
B.2.5 Acquisition Function Optimization 151
B.3 Experiments 152
B.3.1 G-Problem experimental setup 152
B.4 Results 154
B.5 Conclusion and Future Work 155

Bibliography **157**

Summary **173**

Samenvatting **175**

Acknowledgements **177**

About the Author **179**

Chapter 1

Introduction

Multi-objective optimization involves the concurrent optimization of conflicting objectives, aiming to identify a group of solutions that reveal a trade-off among optimal solutions instead of a single optimal solution. Finding these trade-offs is particularly challenging when the evaluation methods of the solutions are computationally expensive. The complexity further increases when constraints are present in the optimization problem. Examples of such constraint multi-objective problems can be found in various engineering design problem domains, including aerospace [33, 34], automotive [116, 157], civil [1, 42], marine [115, 149], and others.

This work shows how feasible optimal solutions in the constraint multi-objective optimization problem domains can be found with optimization algorithms. The primary focus of this work is the development of constraint multi-objective optimization algorithms, with a secondary objective to apply the newly developed algorithms to ship design optimization problems. Before multi-objective algorithms were used, the classical way to optimize multi-objective problems in the maritime industry was to optimize one objective at a time or use traditional mathematical optimization techniques such as the weighted sum method. The pragmatic weighted sum approach combines objectives by multiplying them with predefined weights into one objective. The weighted sum of the objectives is then optimized until an optimal feasible solution is found. However, the weighted sum method has some disadvantages, since a priori weight selection can be difficult, the outcome is sensitive to the weight selection, and if only one weight combination is selected only one solution will be discovered instead of the entire Pareto frontier [99]. Advantages in multi-objective evolutionary algorithms have made multi-objective optimization more common and accessible in the past 30

years [10] and thereby also more usable for the maritime industry.

In ship design, multi-objective optimization can be applied in different design phases. Ship design typically consists of three phases, the preliminary design stage, the contract design stage, and the detailed design stage. According to Taggart [139], early design stages occupy the smallest amount of time. In the first design phases, the operational requirements are translated into phyisical and technical characteristics [89]. This is done by finding the balance between the need of the customer and the available budget, resulting in one or more possible design solutions [54]. Despite the limited amount of time spent in the early design stages, it is estimated that 60 to 80 percent of the total life cycle cost is already locked in after the preliminary design stage [127]. Depending on the experience of the involved Naval Architects, this can be quite risky. The design decisions are typically hard to reverse and are also often made with limited design problem knowledge. It is because of the large room for improvement and the relatively large decision freedom in the early design stage that this work focuses on applying multi-objective algorithms in the preliminary design stage. This is done by investigating what optimization algorithms are needed when using two different design methods:

1. In the empirical design method the main dimensions and ship characteristics are determined by investigating and learning from similar-built vessels.

2. The simulated design method uses 3-D models connected to simulation software that evaluate the ship designs created by naval architects.

The constraint multi-objective problems arising from these two design methodologies exhibit significant differences, particularly when accounting for the required computational effort. Where the empirical design method is computationally inexpensive and relatively simplistic, the simulated design method is computationally more demanding and can lead to really challenging conditions for identifying promising feasible Pareto efficient solutions. More details about the design methodologies are described in Chapter 3.

The availability of design data and more exact and computationally demanding simulation software has led to the need for more advanced optimization algorithms and techniques. The constraint multi-objective optimization algorithms developed in this doctoral dissertation aim to fill this gap. The main scientific contribution therefore is also the newly developed algorithms and methodologies for constraint multi-objective optimization. The proposed methodologies are thoroughly described, tested, and compared to state-of-the-art constraint multi-objective optimization algorithms. The new

algorithms have extended the horizons of knowledge in the field of multi-objective optimization and now give engineers from different domains the tools needed to more efficiently deal with computationally demanding constraint multi-objective optimization problems.

1.1 Research Questions

Finding feasible Pareto efficient solutions for computationally demanding optimization problems is not a straightforward task. The primary research question is therefore as follows:

How to identify the Pareto frontier of constraint multi-objective optimization problems with only a few function evaluations?

The secondary objective is to apply the new algorithms to ship design optimization problems. The research question is therefore separated into the following sub-research questions:

RQ1: What are typical ship design optimization problem characteristics?

RQ2: How can data be used to find feasible Pareto efficient ship design solutions?

RQ3: How to find the Pareto frontier of computationally expensive problems?

RQ3.1: How to deal with expensive multi-objective problems?

RQ3.2: How to efficiently satisfy constraints in multi-objective optimization?

RQ3.3: How to propose multiple solutions for evaluation in parallel?

RQ3.4: How to deal with a mix of expensive and inexpensive functions?

RQ3.5: How do the proposed algorithms compare to state-of-the-art algorithms?

RQ4: What is the performance of the proposed algorithms in real-world scenarios?

1.2 Outline

The research questions are answered in different chapters of this dissertation. The remainder of this thesis is organized as follows: In Chapter 2 the problem notations and relevant preliminaries are given for efficient constraint multi-objective optimization.

Chapter 3 describes ship design optimization problem characteristics and answers research question 1. In Chapter 4 research question 2 is answered and it is shown how multi-objective optimization algorithms are used in combination with machine learning in an empirical ship design method. The largest scientific contribution of this research is described in Chapter 5. In this chapter, the main research question is answered, and multi-objective optimization algorithm configurations are proposed for solving parallel simulation-based optimization problems with mixed expensive constraints and objectives. In Chapter 6 research question 4 is answered and the algorithms from the previous chapter are used to optimize real-world ship design optimization problems. Finally, conclusions are drawn in Chapter 7.

1.3 Publications of this thesis

The majority of the work in this thesis has been published before in academic journals and conference proceedings. The contents of this thesis consist of the following papers.

[146] **Roy de Winter**. Parallel constrained multi-objective optimization for ship design damage stability problem with (in)expensive function evaluations. In *Marine 2023*, page 1. Marine 2023, Scipedia, 2023.

[147] **Roy de Winter**, Thomas Bäck, and Niki van Stein. Modular optimization framework for mixed expensive and inexpensive real-world problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2024.

[148] **Roy de Winter**, Philip Bronkhorst, Bas van Stein, and Thomas Bäck. Constrained multi-objective optimization with a limited budget of function evaluations. *Memetic Computing*, 14:151–164, 2022.

[149] **Roy de Winter**, Jan Furustam, Thomas Bäck, and Thijs Muller. Optimizing ships using the holistic accelerated concept design methodology. In Tetsuo Okada, Katsuyuki Suzuki, and Yasumi Kawamura, editors, *Practical Design of Ships and Other Floating Structures (PRADS)*, pages 38–50, Singapore, 2021. Springer.

[152] **Roy de Winter**, Bas Milatz, Julian Blank, Niki van Stein, Thomas Bäck, and Kalyanmoy Deb. Parallel multi-objective optimization for expensive and inexpensive objectives and constraints. *Swarm and Evolutionary Computation*, 86:101508, 2024.

[153] **Roy de Winter**, Bas van Stein, and Thomas Bäck. SAMO-COBRA: A fast surrogate assisted constrained multi-objective optimization algorithm. In Hisao

Ishibuchi, Qingfu Zhang, Ran Cheng, Ke Li, Hui Li, Handing Wang, and Aimin Zhou, editors, *International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, pages 270–282. Springer, 2021.

[154] **Roy de Winter**, Bas van Stein, Matthys Dijkman, and Thomas Bäck. Designing ships using constrained multi-objective efficient global optimization. In Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, Renato Umeton, and Vincenzo Sciacca, editors, *International Conference on Machine Learning, Optimization, and Data Science*, pages 191–203. Springer, 2018.

[156] **Roy de Winter**, Bas van Stein, and Thomas Bäck. Multi-point acquisition function for constraint parallel efficient multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 511–519, 2022.

[155] **Roy de Winter**, Bas van Stein, and Thomas Bäck. Ship design performance and cost optimization with machine learning. In *20st Conference on Computer and IT Applications in the Maritime Industries (COMPIT)*, pages 185–196. Hamburg University of Technology, 2020.

1.4 Other Work by the Author

Besides the papers that form the content of this thesis, the author has also been involved in the conceptualization, writing, and review process of the following papers:

[10] Thomas Bäck, Anna V Kononova, Bas van Stein, Hao Wang, Kirill Antonov, Roman Kalkreuth, Jacob de Nobel, Diederick Vermetten, **Roy de Winter**, and Furong Ye. Evolutionary algorithms for parameter optimization—thirty years later. *Evolutionary Computation*, 31(2):81–122, 2023.

[27] Philip Bronkhorst, **Roy de Winter**, Thijs Velner, and Austin A Kana. Enhancing offshore service vessel concept design by involving seakeeping-developing a framework to efficiently design high-performance offshore service vessel concepts. In Volker Bertram, editor, *22nd Conference on Computer and IT Applications in the Maritime Industries (COMPIT)*, pages 273–287. Hamburg University of Technology, Schriftenreihe Schiffbau, 2022.

[75] Gideon Hanse, **Roy de Winter**, Bas van Stein, and Thomas Bäck. Optimally weighted ensembles for efficient multi-objective optimization. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 144–156. Springer, 2021.

[80] Qi Huang, **Roy de Winter**, Bas van Stein, Thomas Bäck, and Anna V Kononova. Multi-surrogate assisted efficient global optimization for discrete problems. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1650–1658. IEEE, 2022.

[103] Bas Milatz, **Roy de Winter**, Jelle DJ van de Ridder, Martijn van Engeland, Francesco Mauro, and Austin A Kana. Parameter space exploration for the probabilistic damage stability method for dry cargo ships. *International Journal of Naval Architecture and Ocean Engineering*, page 100549, 2023.

[150] **Roy de Winter**, Fu Xing Long, Andre Thomaser, Niki van Stein, de, Thomas Bäck, and Anna V Kononova. Landscape analysis based vs. domain-specific optimization algorithm selection for engineering design applications: A clear case. In *2024 IEEE Conference on Artificial Intelligence (CAI)*. IEEE, 2023.

[151] **Roy de Winter**, Bas Milatz, Julian Blank, Niki van Stein, Thomas Bäck, and Kalyanmoy Deb. Hot off the press: Parallel multi-objective optimization for expensive and inexpensive objectives and constraints. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2024.

[162] Lucas Van Rooij, **Roy de Winter**, Anna V Kononova, and Bas van Stein. Explainable AI for ship design analysis with AIS and static ship data. In *15th International Symposium on Practical Design of Ships and Other Floating Structures (PRADS)*, pages 1521–1535, 2022.

[163] Niki van Stein, de **Roy de Winter**, Thomas Bäck, and Anna V Kononova. AI for Expensive Optimization Problems in Industry. In *2023 IEEE Conference on Artificial Intelligence (CAI)*, pages 251–254. IEEE, 2023.

Chapter 2

Preliminaries

This chapter introduces the most important acronyms, the problem notations, the basics of expensive black-box optimization, the benchmark functions used to test the performance of the proposed constraint multi-objective algorithms, and the relevant performance metrics for multi-objective optimization.

2.1 Acronyms

Two separate lists of the most important acronyms of this dissertation are made. One list consists of acronyms that are frequently used when discussing optimization algorithms. The second acronym list consists of acronyms used in the ship design domain.

2.1.1 Optimization Acronyms

COBYLA Constraint Optimization BY Linear Approximation algorithm
DoE Design of Experiments
EAF Empirical Attainment difference Functions
ECDF Empirical Cumulative Distribution Function
HV Hypervolume
IGD+ Inverted Generational Distance+ metric
LHS Latin Hypercube Sample
NSGA-II Non-dominated Sorting Genetic Algorithm II
PF Pareto Frontier

RBF Radial Basis Function
SAMO-COBRA Self-Adaptive Multi-Objective Constrained Optimization by using Radial Basis Function Approximations algorithm

2.1.2 Ship Design Acronyms

ρ Water Density
ACD Accelerated Concept Design
B Breadth
Boa Breadth Overall
Cb Block Coefficient
DWT Dead Weight Tonnage
FFD Free Form Deformation
KPI Key Performance Indicators
L Length
LBP Length Between Perpendiculars
LSW Light Ship Weight
MCR Maximum Continuous Rating
RANSE Reynolds Averaged Navier-Stokes Equation
T Draft
TEU Twenty-Foot Equivalent Unit
TSHD Trailing Suction Hopper Dredger
V Service Speed

2.2 Problem Notations

In this work, the following notations are used to define optimization problems. The focus of this work is mainly on optimization problems with two or more objectives, one or more constraints, and continuous decision variables. These problems can mathematically be formulated as follows:

$$\begin{aligned} \text{minimize: } & \mathbf{f} : \Omega \to \mathbb{R}^k \ , \ \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))^\top \\ \text{subject to: } & g_i(\mathbf{x}) \leq 0 \ \forall i \in \{1, \ldots, m\} \\ & \mathbf{x} \in \Omega \subset \mathbb{R}^d. \end{aligned} \tag{2.1}$$

In the following list, the elements and notations are explained in more detail.

- A solution with d design variables in the continuous domain is defined as $(x_1, \ldots, x_d) = \mathbf{x} \subset \mathbb{R}^d$, here d is called dimensionality.
- Multiple decision vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are denoted as a matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$.
- The design space that consists of all solutions is denoted by $\Omega \subset \mathbb{R}^d$. The Design space in this work is bounded by a lower bound and an upper bound $\mathbf{x_{lb}} \leq \Omega \leq \mathbf{x_{ub}}$. Here $\mathbf{x_{lb}} \in \mathbb{R}^d$, and $\mathbf{x_{ub}} \in \mathbb{R}^d$.
- An objective function that evaluates a solution and returns a continuous objective function value is formulated as $f : \Omega \to \mathbb{R}$.
- Unless otherwise specified all objectives are to be minimized: $\min_{\mathbf{x} \in \mathbb{R}^d}(f(\mathbf{x}))$.
- Objectives to be maximized are without loss of generality transformed into objectives to be minimized: $\min_{\mathbf{x} \in \mathbb{R}^d}(f(\mathbf{x})) = -1 \cdot \max_{\mathbf{x} \in \mathbb{R}^d}(f(\mathbf{x}))$.
- In multi-objective problems, k denotes the number of objectives, and the set of k objectives that evaluate a solution is combined into a vector $\mathbf{f} : \Omega \to \mathbb{R}^k$, $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))^\top$.
- A constraint function that evaluates a solution and returns the constraint violation score is formulated as $g : \Omega \to \mathbb{R}$.
- Unless otherwise specified, the constraint feasibility boundary is 0. Constraint values $g(\mathbf{x}) \leq 0$ are defined as feasible while constraint values $g(\mathbf{x}) > 0$ are infeasible. The output of the constraint function $g(\mathbf{x})$ is therefore a value that represents the constraint violation.
- Constraints with a constraint boundary $g(\mathbf{x}) \leq c$ can be rewritten as $g(\mathbf{x}) - c \leq 0$.
- Constraints where $g(\mathbf{x}) \geq 0$ is defined feasible are refactored to $g(\mathbf{x}) \geq 0 \ \rightarrow -1 \cdot g(\mathbf{x}) \leq 0$.
- An equality constraint $h(\mathbf{x}) = 0$ can be replaced with two inequality constraints. A good practice is to add a small margin ϵ around the constraint boundary so that in case there are tiny numerical instabilities this does not influence the feasibility of the solutions. $h(\mathbf{x}) = 0 \ \rightarrow \ g(\mathbf{x}) - 0.5\epsilon \leq 0$, and $g(\mathbf{x}) + 0.5\epsilon \geq 0$.
- In optimization problems m denotes the number of constraints. The set of m constraints can be combined into a vector $\mathbf{g} : \Omega \to \mathbb{R}^m$, $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_m(\mathbf{x}))^\top$.

2.2.1 Pareto Optimal Solutions

Now that the notations are clear, a definition can be given for optimal solutions. First, the definitions are given for a local and a global optimum and later the definition of Pareto optimal and Pareto dominance is given to define which solutions are preferred over other solutions for constraint multi-objective problems.

Definition 2.1 (Local Optimal Solution)**.** A local optimal solution is a solution that is the best in its local surroundings. This solution can be improved by inspecting other regions of the design space.

Definition 2.2 (Global Optimal Solution)**.** A global optimum is the best possible solution for the entire design space. Unlike a local solution, this solution is the absolute best solution and it can not be improved anymore.

Definition 2.3 (Feasible solution)**.** A solution $\mathbf{x} \in \Omega$ is *feasible* if and only if $g_i(\mathbf{x}) \leq 0 \; \forall i \in \{1, \ldots, m\}$.

Definition 2.4 (Dominance)**.** Solution $\mathbf{y} = \mathbf{f}(\mathbf{x})$ *dominates* solution $\mathbf{y}' = \mathbf{f}(\mathbf{x}')$ if and only if $\forall_{i \in \{1,\ldots,k\}} : y_i \leq y'_i$ and $\exists_{i \in \{1,\ldots,k\}} : y_i < y'_i$, in symbols $\mathbf{y} \prec \mathbf{y}'$.

Definition 2.5 (Incomparability)**.** Solution $\mathbf{y} = \mathbf{f}(\mathbf{x})$ is *incomparable* to solution $\mathbf{y}' = \mathbf{f}(\mathbf{x}')$ if and only if $\exists_{i \in \{1,\ldots,k\}} : y_i < y'_i$ and $\exists_{i \in \{1,\ldots,k\}} : y_i > y'_i$, in symbols $\mathbf{y} \| \mathbf{y}'$.

Definition 2.6 (Indifference)**.** Solution $\mathbf{y} = \mathbf{f}(\mathbf{x})$ is *indifferent* to solution $\mathbf{y}' = \mathbf{f}(\mathbf{x}')$ if and only if $\forall_{i \in \{1,\ldots,k\}} : y_i = y'_i$, in symbols $\mathbf{y} \sim \mathbf{y}'$. Note that indifference is equivalent to equality in the objective space. It is not guaranteed to also be equivalent in the design space: $\mathbf{f}(\mathbf{x}) \sim \mathbf{f}(\mathbf{x}') \implies \mathbf{x} = \mathbf{x}'$

Definition 2.7 (Pareto-optimal solution)**.** Solution $\mathbf{x} \in \Omega$ is *Pareto-optimal* if and only if there is no solution $\mathbf{x}' \in \Omega$ for which $f(\mathbf{x}') = \mathbf{y}' \prec \mathbf{y} = f(\mathbf{x})$.

Definition 2.8 (Feasible Pareto-optimal solution)**.** Solution $\mathbf{x} \in \Omega$ is *feasible Pareto-optimal* if and only if there is no solution $\mathbf{x}' \in \Omega$ for which $f(\mathbf{x}') = \mathbf{y}' \prec \mathbf{y} = f(\mathbf{x})$ where $g_i(\mathbf{x}) \leq 0$ and $g_i(\mathbf{x}') \leq 0 \; \forall i \in \{1, \ldots, m\}$.

Definition 2.9 (Pareto optimal set)**.** The set of solutions $\mathbf{X}$ that form the *Pareto optimal set* is the set of feasible incomparable and indifferent solutions that are not dominated by any other solutions $\mathbf{X} = \{\mathbf{x}' \in \mathbf{X} \mid \nexists \mathbf{x} \in \Omega : \mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}')\}$.

2.3 Expensive Black Box Optimization

When solving expensive black-box optimization problems, often Bayesian optimization is chosen as the method to find the optimal solutions. An argument for using Bayesian optimization algorithms is that these algorithms typically require a small evaluation budget to find (a set of) optimal solutions. This makes Bayesian optimization a good choice when the evaluation functions are computationally expensive [53, 90].

Bayesian optimization consists of 4 steps:

1. Bayesian optimization starts with a Design of Experiments (DoE). The solutions from the DoE are evaluated with the objective and constraint functions and put in an archive of evaluated solutions.

2. Surrogate models are fitted with all data from the archive.

3. With an infill criterion the promising solution(s) are selected based on the surrogate(s) predictions. The infill criterion is optimized with an optimization algorithm.

4. The new solutions are evaluated with the objective and constraint functions and added to the archive.

Finally, the algorithm terminates if a stopping criterion is met, otherwise the algorithm goes back to step 2. See Figure 2.1 for a graphical representation.

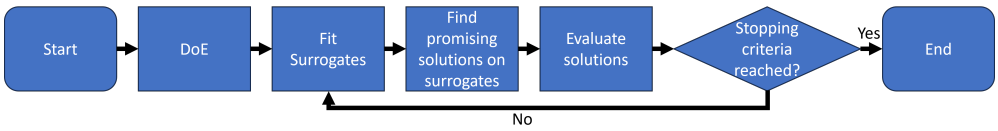


Figure 2.1: Flowchart of Bayesian optimization.

Choosing the optimal DoE strategy, surrogate models, and a local search method to find promising solutions on the surrogates is still a challenge. In the following subsections, a few DoE strategies, surrogate models, and local search methodologies are discussed that are used and referred to in this work.

2.3.1 Design of Experiments

In a design of experiments, the first set of solutions is generated and evaluated. The location in the domain where these first solutions are placed (also sometimes referred

to as initial sampling) is dependent on the DoE strategy. Several possible choices for a DoE exist, including uniform random sampling, full factorial design [17], Latin Hypercube Sampling [138], Halton sampling [73], and Sobol sampling [134]. Each of these methods has its own strengths, however, a large empirical comparison was presented by Bossek et. al. [23]. This empirical study concludes that spending as few evaluations on the DoE as possible is often beneficial because this leaves more room for evaluations proposed by the optimization algorithm [23]. This empirical finding also works best for most constraint multi-objective problems [148].

A recently proposed new sampling method is the Riesz s-energy-based sampling method [77]. The Riesz s-energy-based sampling method iteratively improves and proposes an arbitrary number of well-spaced points in the design space [21]. This method has been modified for constraint search spaces [19] so that it samples solutions only in the feasible area of the design space. This however is only practically applicable if the constraint functions are inexpensive to evaluate.

In industrial settings, solutions can often be evaluated in parallel. The number of solutions that can be evaluated in parallel are denoted with a p in this work. The number of solutions that can be evaluated in parallel is often dependent on the available computational resources and, if applicable, the number of commercial simulator licenses. The resources for parallel evaluations can be used to evaluate the initial DoE. The size of the DoE however now should be chosen based on the size of p. It is advised to choose a DoE size of $\lceil DoE_{min}/p \rceil \cdot p$, where p is the maximum possible number of simultaneous parallel evaluations and DoE_{min} the smallest DoE size required for training the first surrogate models. This way, no wall clock time is wasted, and the maximum amount of information from the objectives and constraints is gathered.

2.3.2 Surrogate Models

For the algorithm proposed in this research Radial Basis Functions (RBFs) and Kriging (also known as Gaussian process regression) are considered surrogate models. RBF and Kriging surrogates are fundamentally very similar, however, RBFs have many advantages: 1) RBFs require smaller sample sizes to fit a surrogate, 2) they are generally faster (also for larger input spaces), 3) have fewer assumptions on the underlying data, 4) deliver in many cases equal or better accuracy, 5) and with a newly developed uncertainty quantification method RBFs can now also be used in infill criteria that require this [12, 57]. Besides these fundamental arguments, an empirical comparison showed faster convergence and better results for algorithms with RBF surrogates compared to

the algorithm with Kriging surrogates [148]. For these reasons, the algorithm proposed in this work uses RBFs as surrogate models.

Radial Basis Function Interpolation

Radial Basis Functions (RBFs) are a type of mathematical function used for approximating the relationship between input and output variables [29]. The input variables are often the decision variables ($\mathbf{x}$), and the output variables are the objective ($\mathbf{f}$) or constraint ($\mathbf{g}$) value of the evaluated solutions. RBF interpolation approximates a function by fitting a linear weighted combination of RBFs [13]. The challenge is to find correct weights ($\boldsymbol{\theta}$) and a good RBF kernel $\varphi(\|\mathbf{x} - \mathbf{c}\|)$. An RBF is only dependent on the distance between the input point $\mathbf{x}$ to the center $\mathbf{c}$. The RBFs used in this work take each evaluated point as the centroid of the function, and the weighted linear combination of RBFs always produces a perfect fit through the training points. Besides the perfect fit on the training points, the linear combination of the RBFs can also give a reasonable approximation of the unknown area.

Any function which is only dependent on the distance from a specific point to another point belongs to the group of RBFs. The RBF kernels (φ) considered in this work are the *cubic* with $\varphi(r) = r^3$, *Gaussian* with $\varphi(r) = \exp\left(-(\epsilon \cdot r)^2\right)$, *multiquadric* with $\varphi(r) = \sqrt{1 + (\epsilon \cdot r)^2}$, *inverse quadratic* with $\varphi(r) = (1 + (\epsilon \cdot r)^2)^{-1}$, *inverse multiquadric* with $\varphi(r) = (\sqrt{1 + (\epsilon \cdot r)^2})^{-1}$, and *thin plate spline* with $\varphi(r) = r^2 \log r$. Note that the shape/width parameter ϵ for every individual RBF is kept constant as proposed by Urquhart et al. [159]. Moreover, all shape parameters are fixed to $\epsilon = 1$.

Finding suitable linear weighted combinations $\boldsymbol{\theta}$ of the RBFs can be done by inverting $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n}$ where $\boldsymbol{\Phi}_{i,j} = \varphi(\|\mathbf{x}_i - \mathbf{x}_j\|)$:

$$\boldsymbol{\theta} = \boldsymbol{\Phi}^{-1} \cdot \mathbf{f} \tag{2.2}$$

Here $\mathbf{f}$ is a vector of length n with the function values of one of the objectives or constraints. Because $\boldsymbol{\Phi}$ is not always invertible, Micchelli introduced RBFs with a polynomial tail, better known as augmented RBFs [101]. In this work, augmented RBFs are used with a second-order polynomial tail. The polynomial tail is created by extending the original matrix $\boldsymbol{\Phi}$ with $\mathbf{P} = (1, x_{i1}, \ldots, x_{id}, x_{i,1}^2, \ldots, x_{id}^2)$, in its ith row, where x_{ij} is the j-th component of vector $\mathbf{x}_i$, for $i = 1, \ldots, n$ and $j = 1, \ldots, d$, $\mathbf{P}^\top$,

and zeros $\mathbf{0}_{(2d+1)\times(2d+1)}$, leading to $1+2d$ more weights $\boldsymbol{\mu}$ to learn.

$$\begin{bmatrix} \boldsymbol{\Phi} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{0}_{(2d+1)\times(2d+1)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0}_{2d+1} \end{bmatrix} \tag{2.3}$$

Now that the weights $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ can be computed with Eq. 2.2, for each unseen input $\mathbf{x}'$ the function value (f') can be interpolated/predicted by using Eq. (2.4).

$$\begin{aligned} f' &= \boldsymbol{\Phi}' \cdot \left[\boldsymbol{\theta}\boldsymbol{\mu}\right] \\ f' &= \sum_{i=1}^{n} \theta_i \varphi(\|\mathbf{x}' - \mathbf{x}_i\|) + \mu_0 + \sum_{l=1}^{d} \mu_l \mathbf{x}'_l + \sum_{l=1}^{d} \mu_l \mathbf{x}'^2_l, \\ \mathbf{x} &\in \mathbb{R}^d \end{aligned} \tag{2.4}$$

Bagheri et al. also exploited similarities between RBF and Kriging surrogates to come up with an uncertainty quantification method for RBFs [12]. The formula for this uncertainty quantification method is given in Eq. (2.5).

$$\widehat{U}_{RBF} = \varphi(\|\mathbf{x}' - \mathbf{x}'\|) - \boldsymbol{\Phi}'^\top \boldsymbol{\Phi}^{-1} \boldsymbol{\Phi}' \tag{2.5}$$

where $\varphi(\|\mathbf{x}' - \mathbf{x}'\|) = \varphi(0)$ is a scalar value.

The uncertainty ($\widehat{U}_{RBF}$) of solutions far away from earlier evaluated solutions is higher compared to solutions close to earlier evaluated solutions. This uncertainty quantification method can therefore be used in infill criteria that require uncertainty quantification method to avoid getting stuck in a local optimal solution. However, as can be derived from Eq. (2.5), the uncertainty quantification method is only dependent on the input space and not on the scale of the objective and/or weights of the RBF models. It is therefore needed to use a consistent scale for both the input and the output space.

Scaling Techniques

For surrogate approximations, various scaling and transformation functions can be used to improve the surrogate fit. Four scaling and transformation techniques used in this work are described below.

Scale: The input space/decision variables are scaled into the range $[-1, 1]$ with $\mathbf{x} = 2 \cdot (\mathbf{x} - \mathbf{x_{lb}})/(\mathbf{x_{ub}} - \mathbf{x_{lb}}) - 1$. By scaling large values in the input space, computationally singular (ill-conditioned) coefficient matrices in Eq. (2.2) can

be prevented. In case the large values in the input space are kept, the linear equation solver will terminate with an error, or it will result in a large root mean square error [13]. Additionally, when fitting the RBFs, a change in one of the variables, is relatively the same change in all the other variables, making each variable in the basis equally important and equally sensitive.

STANDARDIZE: The relationship between the input space and the objective function values is modeled with the surrogates. To keep a consistent scale between the input and output scale, the output function values are standardized by using $y' = (y - \bar{y})/\sigma$. Here σ is the standard deviation of y, and $\bar{y}$ the mean of y. By using this standardization method, the uncertainty quantification from Eq. (2.5) can be used.

SCALE CONSTRAINT: The constraint evaluation function should return a continuous value, namely the amount by which the constraint is violated. Since it is possible to have multiple constraints, and each constraint is equally important, every constraint output is scaled with $c' = c/(\max(c) - \min(c))$, where $\max(c)$ is the maximum constraint violation encountered so far, and $\min(c)$ is the smallest constraint value seen so far. After scaling, the difference between $\min(c)$ and $\max(c)$ becomes 1 for all constraints, making every constraint equally important while 0 remains the feasibility boundary.

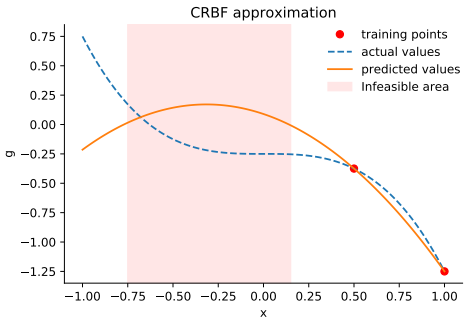
PLOG: In cases where there are very steep slopes, a logarithmic transformation of the objective and/or constraint scores can be beneficial for the predictive accuracy [125]. Therefore, the scores are transformed with the PLOG transformation function. The extension to a matrix argument $\mathbf{Y}$ is defined component-wise, i.e., each matrix element y_{ij} is subject to PLOG.

$$\text{PLOG}(y) = \begin{cases} +\ln(1+y), & \text{if} \quad y \geq 0 \\ -\ln(1-y), & \text{if} \quad y < 0 \end{cases} \tag{2.6}$$

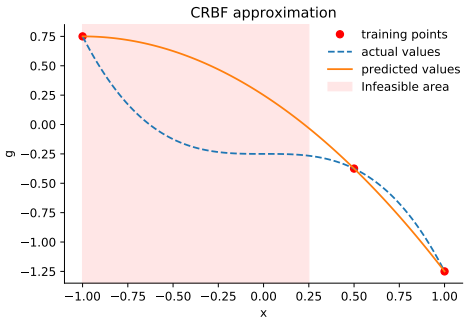
Radial Basis Functions Illustrative Examples

A visual representation (adapted from [142]) of how a Cubic RBF surrogate model is used to model a 1-dimension constraint function and how they become more accurate when more training points are added is presented in Figure 2.2. In the figures, the dashed blue line is the constraint function that the RBFs have to approximate, the red dots are the training points that have been evaluated and used to train the cubic RBF

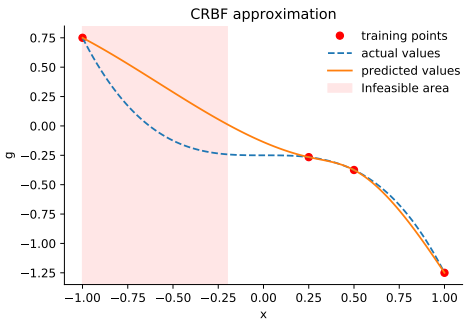
model, the solid orange line is the RBF prediction, the red shaded area is the predicted infeasible area where the RBF prediction is larger than 0 ($g(x) > 0$). Evident from the figures is that the surrogate's accuracy improves with the addition of more training points.



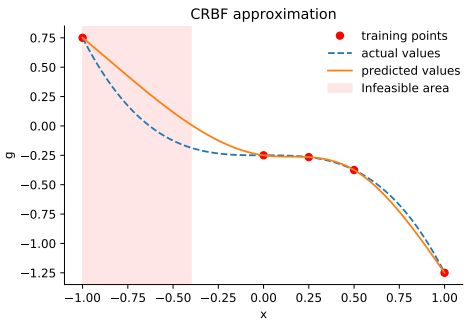
(a) RBF fit with 2 training points.



(b) RBF fit with 3 training points.



(c) RBF fit with 4 training points.



(d) RBF fit with 5 training points.

Figure 2.2: The dashed blue line is the constraint function that the Cubic RBFs have to mimic, the red dots are the training points that have been evaluated and used to train the cubic RBF model, the solid orange line is the RBF prediction given x, the red shaded area is the predicted infeasible area where the surrogate prediction is larger than 0 $g(x) > 0$.

Drawbacks of using Surrogates

There are a few scenarios for which surrogates are not ideal to use. If for example the constraint or objective is computationally cheaper than evaluating a solution on a surrogate, it is better to directly use the original constraint and or objective function. Another reason to not use surrogate-assisted algorithms is in the scenario where the

objective and or constraint function is highly multi-modal. If the global structure of the objective function is weak, and there are many local structures consisting of hills, finding an accurate fit of the to-be-modeled function with a surrogate is very difficult. This is difficult because many training points would be required to get an accurate fit of all the hills and alleys. It is for this reason that highly multi-modal problem landscapes are difficult to model with surrogates. Decision makers should therefore take the multi-modality, and expensiveness of the functions into account when selecting surrogate-assisted optimization algorithms.

2.3.3 Local Search Methodologies For Surrogate Exploration

Finding good solutions on surrogates is done by optimizing an acquisition function (also sometimes refereed to as infill criterion). The acquisition functions in this work are optimized with the so-called Constraint Optimization BY Linear Approximations algorithm (COBYLA) [120].

COBYLA linearly approximates the constraints and the acquisition function in a small trust region. In this trust region, COBYLA maximizes the acquisition function subject to the constraints by maximizing the following function:

$$\Psi(\mathbf{x}) = \hat{F}(\mathbf{x}) + \mu \left(-\max(c_i(\mathbf{x}) : i = 1, \ldots, m)\right)_+, \quad \mathbf{x} \in \mathbb{R}^d \tag{2.7}$$

Here, $\mathbf{x}$ is a solution in the input space, $\hat{F}$ is in our case the linear approximation of the acquisition function, c_i is the i-th linear approximation from the m constraint functions, the subscript $+$ means that the expression in the brackets becomes 0 if none of the constraints are violated, and μ is a self-adaptive penalty parameter that makes sure that the approximation of a new solution $\Psi(\mathbf{x}^*)$ with a smaller constraint violation and better acquisition function score is preferred over the starting solution approximation $\Psi(\mathbf{x}^0)$. After the best solution in the trust region on the linear approximations is found, it is evaluated on the constraint surrogate and acquisition function. When the linear approximation of COBYLA in the trust region underestimates the acquisition function, the trust region increases in size, while if it overestimates, the trust region becomes smaller. This way, when nearing the solutions that have the optimal acquisition score, the trust region becomes smaller and smaller until it falls below an $\varepsilon > 0$ value and COBYLA terminates.

Since COBYLA is a local optimizer, it can get stuck in a local optimum [169]. To overcome this problem, multiple instances of COBYLA are run in parallel. Searching for optimal solutions in multiple locations simultaneously is also a well-known strategy

in the Island model for parallel optimization [70], in parallel simulated annealing [91], and in ant colony optimization [51]. After all COBYLA instances have converged, all solutions are found. The solution with the best acquisition score is selected for evaluation on the expensive objective and constraint functions.

2.4 Benchmark Test Functions

For assessing the performance of various constraint multi-objective optimization algorithms a set of benchmark test functions are collected. An overview of the test functions is given in Table 2.1. In this table, the reference point (the worst possible value for each objective), the approximated Nadir point (the worst possible value for each objective among all solutions on the Pareto frontier), the number of objectives k, the number of dimensions d, the number of constraints m, and the feasibility ratio (P%) after one million random samples are given.

Table 2.1: Test function name, reference point used during optimization, Nadir point approximation based on all experiments in this work, number of objectives k, number of decision variables d, number of constraints m, percentage of feasible solutions $P(\%)$ after one million random samples.

Function	Reference point	Nadir point	k	d	m	$P(\%)$
BNH	(140, 50)	(136.00, 50.00)	2	2	2	96.92
CEXP	(1, 9)	(1.00, 9.00)	2	2	2	57.14
SRN	(301, 72)	(222.99, 2.62)	2	2	2	16.18
TNK	(2, 2)	(1.04, 1.04)	2	2	2	5.05
CTP1	(1, 2)	(1.00, 1.00)	2	2	2	92.67
C3DTLZ4	(3, 3)	(2.00, 2.00)	2	6	2	22.22
OSY	(0, 386)	(-41.81, 76.00)	2	6	6	2.78
TBTD	(0.1, 50000)	(0.1, 10000)	2	3	2	19.46
NBP	(11150, 12500)	(12500, 114.09)	2	2	5	41.34
DBD	(5,50)	(2.79, 16.86)	2	4	5	28.55
SRD	(7000, 1700)	(5879.98, 1696.46)	2	7	11	96.92
WB	(350, 0.1)	(35.31, 0.0145)	2	4	5	35.28
BICOP1	(9, 9)	(1.00, 1.00)	2	10	1	100
BICOP2	(70, 70)	(1.10, 1.11)	2	10	2	10.55
MW1	(1,7)	(1.00, 1.00)	2	8	1	0.007
MW2	(1,7)	(1.00, 1.00)	2	6	1	0.55
MW3	(1,7)	(1.00, 1.00)	2	6	2	1.32
MW11	(30,30)	(2.06, 2.04)	2	6	4	1.38
TRIPCOP	(34, -4, 90)	(7.67, -11.77, 25.91)	3	2	3	15.85
SPD	(16, 19000, -260000)	(11.16, 12435.27, -259148.04)	3	6	9	3.27
CSI	(42, 4.5, 13)	(42.77, 4.00, 12.52)	3	7	10	18.17
WP	(83000, 1350, 2.85, 15989825, 25000)	(74573, 1350, 2.85, 7874925, 25000)	5	3	7	92.06

The following functions are artificially created test functions: BNH [41], CEXP [46], SRN [50], TNK [50], CTP1 [46], C3DTLZ4 [141], OSY [41, 50], NBP [62], BICOP1 [44], BICOP2 [44], TRICOP [44], MW1 [97], MW2 [97], MW3 [97], MW11 [97].

The following functions are coming from real-world problems: Two-Bar Truss Design (TBTD) [66], Disk Brake Design (DBD) [66], Ship Parametric Design (SPD) [117],

Car-Side Impact (CSI) [83], speed Reducer Design (SRD) [105], Welded Beam (WB) [66], Water resource management Problem (WP) [83].

The test functions are selected because they are diverse, well known, and some mimic industrial problems. The Pareto frontiers of the functions vary between 2 and 5 objectives and the shapes can be classified as concave, convex, connected, disconnected, or even mixes of these characteristics [6]. The constraints of the selected test problems are also diverse since for some problems they are very strict, while for other problems (almost) the entire search space is feasible. Next to the feasibility of the problems, on some Pareto frontiers, the constraints are active, while on other problems they are not, or partially active. Next to the artificially created test functions, a set of real-world-inspired problems is selected to assess how well the optimization algorithms operate in situations resembling industrial optimization scenarios.

2.5 Algorithm Performance Metrics

The performance of algorithms can be determined with performance metrics. These metrics help to determine which algorithm perform well on benchmark problems and visualizing the evolution of the performance metric gives insight in how fast the algorithms converge. Since the focus of this work is multi-objective optimization algorithms the most used multi-objective metrics are presented and visualized.

2.5.1 Multi-Objective Performance Metrics

The two commonly used multi-objective performance metrics used in this work are the HyperVolume (HV), and the Inverted Generational Distance+ (IGD+) metric.

Hypervolume

The *hypervolume* (also known as the Lebesgue measure) translates the multi-objective problem into a unary performance score that represents the volume of the region in the objective space that is dominated by a given set of solutions [18, 175]. It is the most widely used performance metric in multi-objective optimization [128] and measures and captures the overall convergence and diversity of the set of solutions forming the Pareto front. The HV is calculated by determining the volume of the region in the objective space between the solutions on the obtained Pareto front and a pre-defined reference point (also sometimes referred to as the anti-optimal point [158]). For a single solution on two objective problem, this is easy to compute as it is the surface

between the solution and the reference point. If more solutions are on the Pareto front, the overlapping region is only counted once, this is done by calculating the union of the overlapping regions. The formal definition of the hypervolume is given in Definition 2.10.

Definition 2.10 (Hypervolume Indicator)**.**

$$HV(\mathbf{Y}, \mathbf{f}^{ref}) = \Lambda_k(\cup_{\mathbf{y_i} \in \mathbf{Y}}[\mathbf{y}, \mathbf{f}^{ref}]) \tag{2.8}$$

here Λ_k denotes the Lebesgue measure on $\mathbb{R}^k$, with k being the number of objective functions, $\mathbf{y}_i$ the i-th Pareto optimal solution, $\mathbf{Y}$ all Pareto optimal solutions, and $\mathbf{f}^{ref}$ the reference point in k dimensions.

The HV is a useful measure for comparing the performance of different optimization algorithms, as well as for comparing different solution sets with each other. Solution sets with higher HV are considered better compared to solutions with lower HV. The HV of three solutions is visualized in Figure 2.3, where the triangles represent the evaluated non-dominated solutions, and the reference point is indicated by a star. The surface of the shaded area is the HV of this particular Pareto front.

Inverted Generational Distance +

Another performance metric often used in multi-objective optimization is the *inverted generational distance+* metric (IGD+) [82]. It is used as a measure to check how close the known Pareto frontier is to the obtained dominated area. The IGD+ metric evaluates diversity and convergence as follows:

$$IGD^+(A, S) = \frac{1}{\mid S \mid} \left(\sum_{i=1}^{|S|} (d_i^+)^2 \right)^{\frac{1}{2}} \tag{2.9}$$

Here S is the known Pareto front, A is the dominated area by a Pareto front Y obtained by an algorithm, and d_i^+ is the smallest Euclidean distance from a solution on the known Pareto front s_i to the dominated area of A. This way, if the obtained dominated area of the solutions found by the algorithm is far away from the known Pareto front, the IGD+ value increases. A smaller IGD+ value is therefore preferred over a larger IGD+ value. The IGD+ metric and the distances of 8 known solutions are visualized in Figure 2.4.

The IGD+ metric can only be used on test instances where the Pareto front is

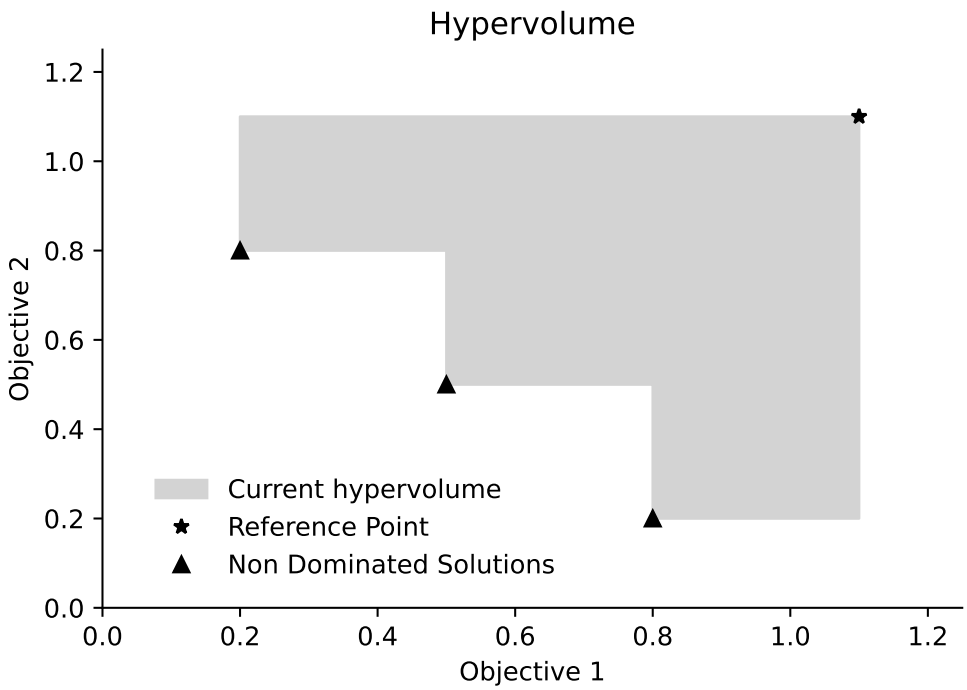


Figure 2.3: Visual representation of hypervolume. Total hypervolume between the 3 solutions and the reference point is $0.3 \times 0.8 + 0.3 \times 0.5 + 0.3 \times 0.3 = 0.48$.

known. In this work, the Pareto front used for the IGD+ metric is approximated by combining all obtained feasible Pareto efficient points of all experiments, then normalizing the objective scores, and finally selecting a well-spread set of solutions. The true Pareto front of computationally expensive engineering problems are often impossible to determine, therefore this revised metric is only used on test problems.

2.5.2 Optimization Result Visualizations

One of the key goals of Bayesian optimization is finding the global optimal (set of) solutions in as few evaluations as possible. Visualizing the set of optimal solutions in multi-objective optimization is usually done on a 2-dimensional or 3-dimensional Pareto frontier. When more than 3 objectives are to be visualized, or the objectives should be visualized together with the constraints and decision variables, then a parallel coordinate plot can be used. To visually compare the obtained Pareto frontiers of more than one algorithm run Empirical Attainment Function (EAF) difference plots

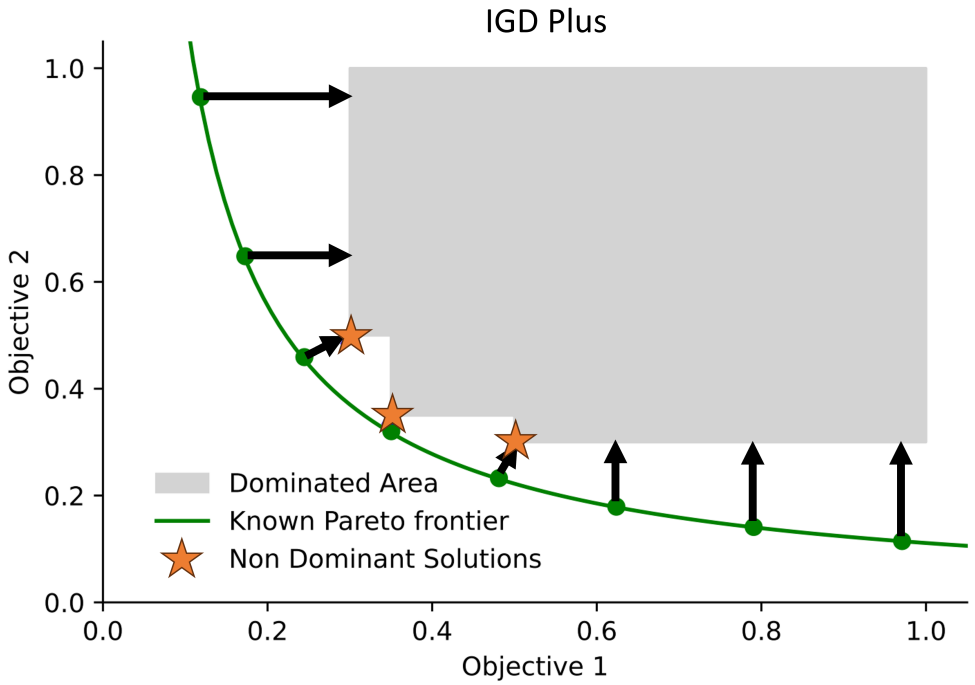


Figure 2.4: IGD+ score visualized on a two objective problem. IGD+ is the average length of the arrows from the well-spread known solutions to the closest point of the dominated area.

can be used for two-dimensional Pareto frontiers. The downside of visualizing only the final result is that information about how fast the results are found is missing. Convergence plots fortunately can give answer to this issue. On convergence plots, a performance score is plotted after every function evaluation from a run of an optimization algorithm on a specific test function. However, when comparing the convergence of algorithms on a set of different function evaluations Empirical Cumulative Distribution Function (ECDF) plots can be used. In the following subsections, all the visualization techniques are explained in more detail.

Pareto Frontier Plot

The Pareto frontier plot is a visual representation of the objective scores from Pareto-optimal solutions. The objective score of each Pareto efficient solution is plotted on the Pareto frontier. The Pareto frontier this way shows the trade-off between the objectives. Improvements of a solution in one objective on the Pareto frontier can only be made by sacrificing any of the other objectives. An example Pareto frontier

on the Two-bar Truss Design (TBTD) problem [66] obtained with the SAMO-COBRA algorithm (the SAMO-COBRA algorithm will be introduced in Chapter 5) is given in Figure 2.5.

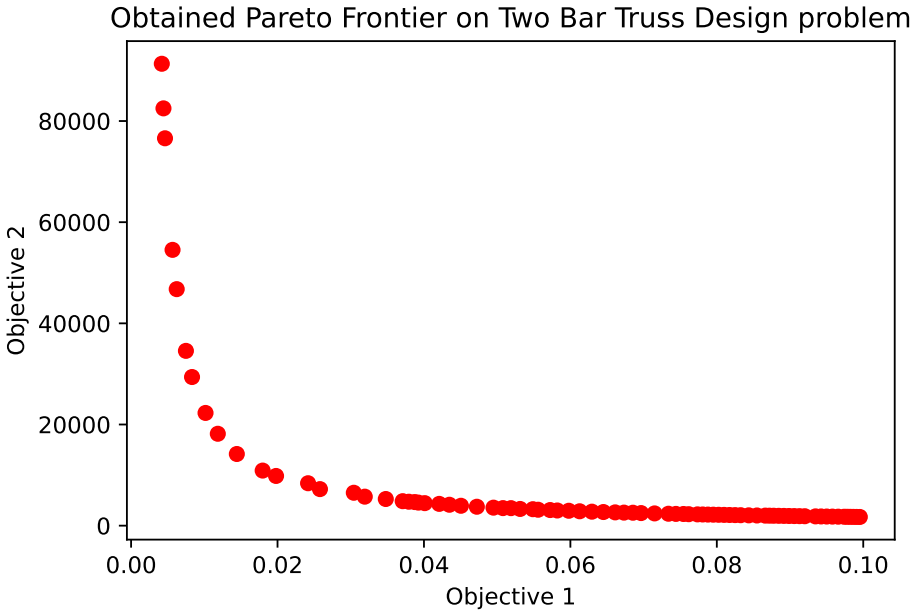


Figure 2.5: Pareto Frontier of TBTD Problem obtained with SAMO-COBRA algorithm after 120 function evaluations.

By plotting the Pareto frontiers of different algorithms, the objective space can be inspected. Inspection of the objective space shows in which regions which algorithm performs better. An illustrative example of the SAMO-COBRA algorithm and the SA-NSGA-II algorithm (The SA-NSGA-II algorithm will be introduced in Chapter 5) of the TBTD problem is presented in Figure 2.6.

From this figure it can be concluded that SAMO-COBRA has obtained slightly better and many more good solutions for objective 2, while the SA-NSGA-II algorithm has achieved better results in objective 1 since it found very small values for this objective.

Parallel Coordinate Plots

Two dimensions can be plotted on a regular Pareto frontier with two axes. In higher dimensions, parallel coordinate plots can be used [78]. In a parallel coordinate plot,

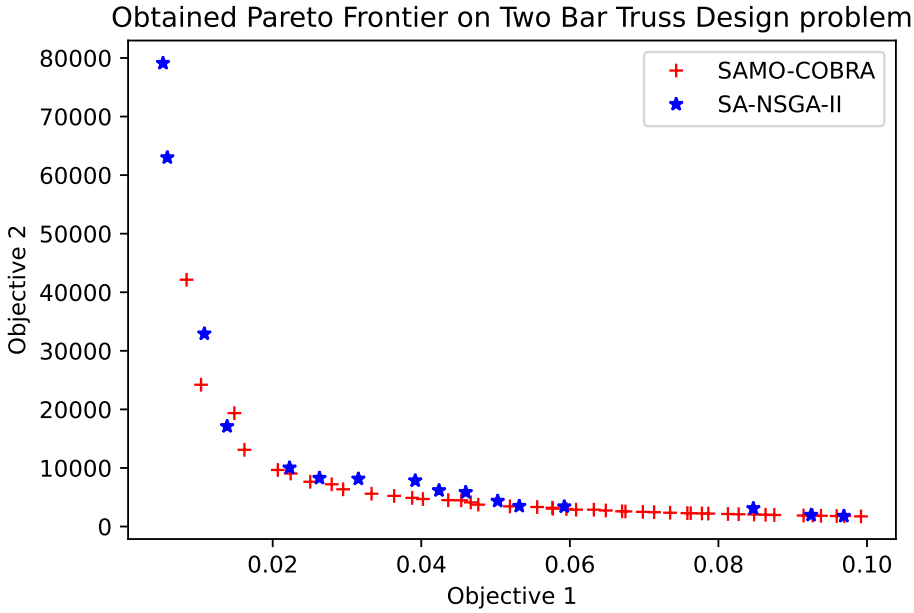


Figure 2.6: Pareto Frontiers on TBTD problem obtained with the SAMO-COBRA and SA-NSGA-II algorithm after 120 function evaluations.

each vertical axis represents a parameter, objective, and/or constraint while the horizontal axis has no meaning except for that it is the minimum and maximum value that has been found for the objective. Each solution in the parallel coordinate plot is represented by a line that connects the vertical axes. An example of a parallel coordinate plot of the Water resource management Problem (WP) [83] is presented in Figure 2.7.

The first thing that can be concluded from this parallel coordinate plot is that the solutions on each axis are very well spread since there are no large gaps between the solutions. The second thing that can be concluded is that there is a very clear inverse correlation between objective 3 and objective 4.

Empirical Attainment Difference Function

Comparing the obtained Pareto frontiers of different algorithms of one run per algorithm can still be done with a regular Pareto frontier plot as in Figure 2.6. However, when the algorithms have been used to optimize the benchmark problem more than once and the results are each time a bit different due to the stochastic nature of the

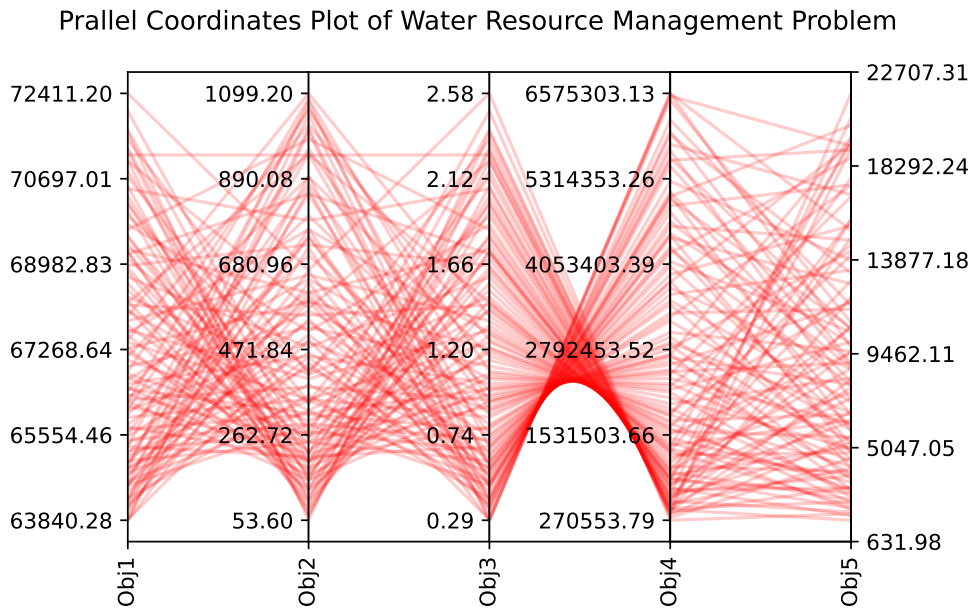


Figure 2.7: Parallel coordinate plot of objective scores from the 5 objective water resource management problem.

algorithms then comparing them becomes more challenging. For this reason, Empirical Attainment Difference Functions (EAF) have been developed [96]. In the EAF difference plots the dark areas mark where the two algorithms have obtained different results. The more frequent a certain area is dominated by an algorithm the darker the grayscale is. An example of an EAF difference plot on the TBTD problem is given in Figure 2.8. The EAF difference plot again confirms what was shown earlier in Figure 2.6, the SAMO-COBRA algorithm manages to find the minimum values of objective 2 on the Pareto frontier, while SA-NSGA-II found smaller values for objective 1.

Convergence Plots

On convergence plots, the x-axis typically shows the number of function evaluations required to achieve a performance score that is on the y-axis. By analyzing the convergence plots it can be identified after how many evaluations the algorithm has found a good (set of) optimal solution(s) and has converged. If multiple algorithms (or algorithm configurations) are plotted together, the convergence can be compared. An

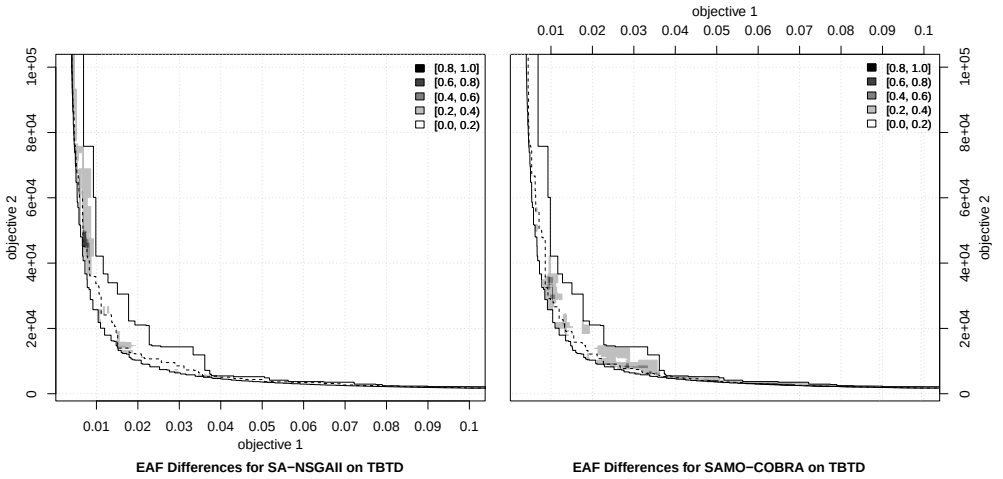


Figure 2.8: Empirical Attainment Difference Function plot on TBTD problem for comparing 10 independent SA-NSGA-II and 10 SAMO-COBRA runs.

illustrative example for a convergence plot on the TBTD problem is given in Figure 2.9.

In the convergence plot where the hypervolume is to be maximized, it is shown that the SAMO-COBRA algorithm with batch size 1 achieves the highest hypervolume after 120 function evaluations. Besides the final score the convergence plot also shows the performance after fewer function evaluations. Inspection of this tells us that the SAMO-COBRA with batch size 1 obtains the best results and converges faster compared to the other configurations.

Empirical Cumulative Distribution Functions

Empirical Cumulative Distribution Functions (ECDF) [76, 166] are used to visualize the convergence of the different algorithms on a set of test functions simultaneously in one plot. An ECDF plot is based on a set of target values that are linearly distributed (sometimes also other distributions are chosen) between zero and the maximum achievable performance score per test function. The proportion of target values attained by the algorithm is the score reported on the ECDF curve. To be able to visualize the performance of an algorithm on multiple different benchmark functions, the corresponding ECDF target score proportions are aggregated. This then results in a curve that grows as more target values are reached. A formal description, adapted from [166], is given in the following definition:

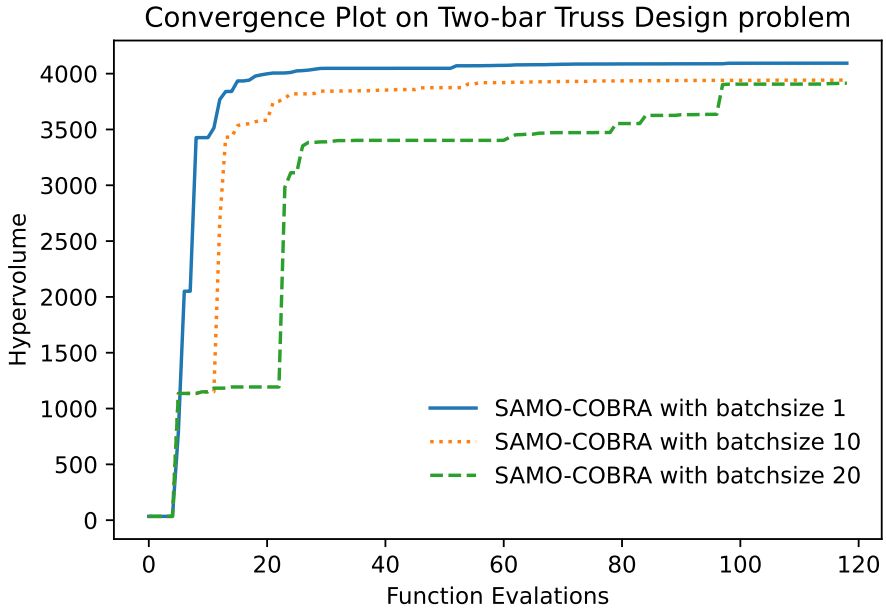


Figure 2.9: Illustrative example of a convergence plot of SAMO-COBRA algorithm with different batch sizes on TBTD problem.

Definition 2.11. An ECDF curve requires to select a set $\{v_1, \ldots, v_n\}$ of target values. The ECDF shows for each budget t the fraction $| \{(i,j) \mid 1 \leq i \leq r, 1 \leq j \leq n\} | / r \cdot n$ of the (run r, target value) pairs (i, v_j) that satisfy that $V(A, f, t, i) \geq v_j$. Here $V(A, f, t, i)$ denotes the function value of the best among the first t evaluated solution candidates in run i. So the ECDF is expressed as $\hat{F}(t) = \frac{1}{nr} \sum_{j=1}^{n} \sum_{i=1}^{r} \mathbf{1}_{V(A,f,t,i) \geq v_j}$, where $\mathbf{1}_C$ denotes the indicator variable, which is one where the condition C is satisfied.

Two ECDF curves with hypervolume as the performance metric, 10 independent runs per benchmark problem from Section 2.4, are given in Figure 2.10. The ECDF curve of the SAMO-COBRA algorithm is completely above the results from the SA-NSGA-II algorithm. This shows that for each budget the SAMO-COBRA algorithm achieves a larger portion of the target values compared to the SA-NSGA-II algorithm. This means that on average the SAMO-COBRA algorithm finds a better hypervolume and also on average converges faster.

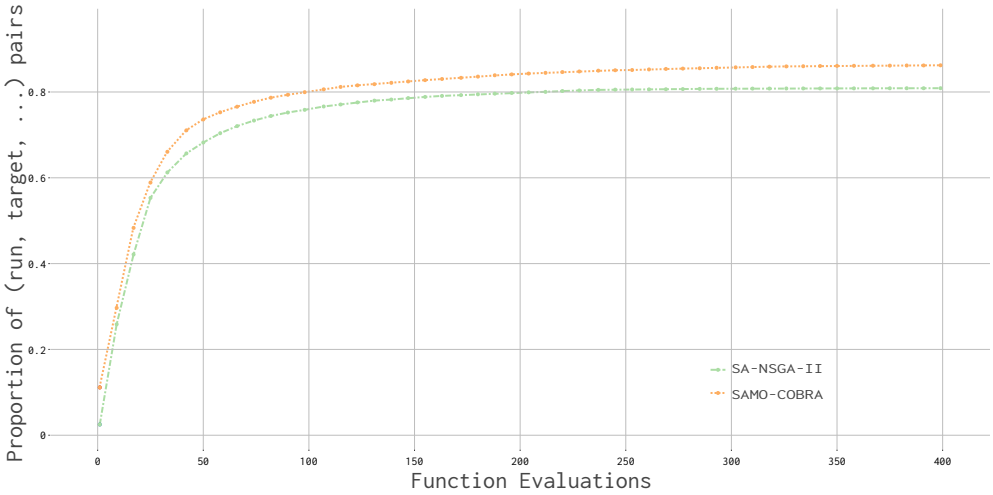


Figure 2.10: ECDF Curve that aggregates the results for the SA-NSGA-II and SAMO-COBRA algorithm that have optimized all benchmark problems from section 2.4 10 times.

Chapter 3

Ship Design Problem Characteristics

Ships are design intensive and belong to the most complex engineering objects worldwide [35, 106]. In this chapter, the aim is to detect patterns in this complex engineering problems by answering research question 1: *What are typical ship design optimization problem characteristics?* This chapter therefore describes ship design problem characteristics, their evaluation methods, guidelines for holistic ship design optimization, and an illustrative example.

3.1 Introduction

Traditionally, initial ship designs are generated by modifying existing designs by experienced architects [113]. The decisions made early in the design process are made using limited time and budgets and are difficult and costly to reverse in later design stages when these decisions turn out to be sub-optimal [68, 115]. In this traditional way, ships are designed and optimized in an iterative design process illustrated by the design spiral [58]. In the design spiral as presented in Figure 3.1 several sequential decisions are made in such a manner that the objectives are optimized and constraints imposed by physics and regulating authorities are met. However, this approach is time-consuming and the design process likely leads to a so-called locally optimal solution. The locally optimal solution is found due to the fact that it is impossible for human experts to consider all the dependencies between the decision variables, the

constraints, and the objectives [112].

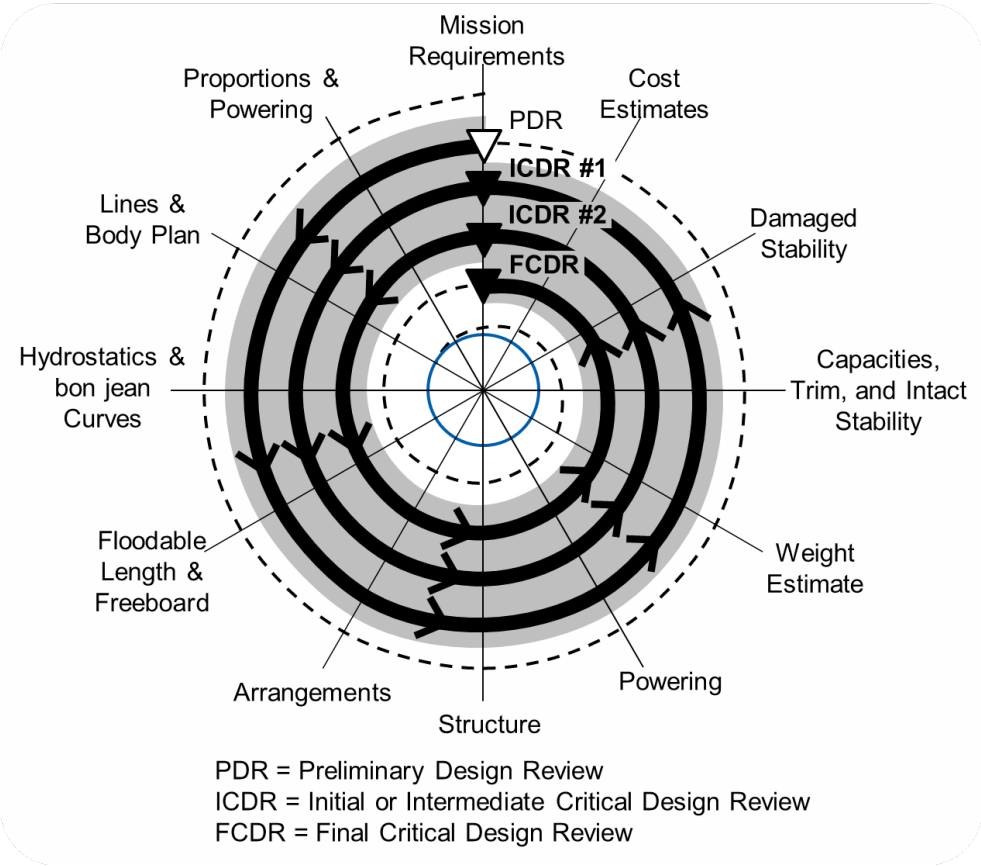


Figure 3.1: Classic Design Spiral [58] where iteratively details are added and where iteratively the design is adjusted by checking all ship design aspects requirements and optimizing the objectives one by one for various ship design stages.

In practice, this means that a ship is optimized not in one iteration but by designing and building multiple ones. This is because in the first design phase, the main dimensions of a ship are to be decided on. This brings many complex engineering questions which will have a very high impact on the final result. However, it is these decisions that need to be made in the shortest amount of engineering time. The more unique the ship is, the higher the challenge will be to answer the questions with confidence. Answering these complex engineering questions is usually done based on experience and heuristics. As the design progresses, these early choices are hard to reverse without major delay. This pushes the designers towards risk mitigation instead

of innovation [149]. Innovation is therefore often seen over several designs, each being a small improvement over the other.

Decisions in the ship design process can be made using two different design philosophies, the empirical and the simulated design method. The empirical design method is based on reference data of similar built vessels. The simulated design method uses estimations, calculations, and simulations to optimize the economical and physical characteristics of the to-be-designed vessel.

3.1.1 Empirical Design Method

In the Empirical Design method, the main dimensions are based on similar-built vessels. A similar vessel is marginally improved or the data from a set of similar vessels is used to make a regression model, after which empirical design formulas can be deducted. Examples of empirical design equations to estimate light ship weight have been created by Watson [45], Schneekluth and Bertram [129], D'Almeida [55], Andrews [7], Molland [108] and Papanikolaou [114]. With these equations, a naval architect can easily estimate ship design parameters and their corresponding KPIs. The equations are usually calibrated for different ship types. It is however important to handle the relations carefully. It is the naval architects' job to update the relationships whenever possible [108]. In reality updating is often not frequently done and the equations are only available for the most popular ship types. A second note to keep in mind is that extrapolation of the regression models remains problematic [114].

3.1.2 Simulated Design Method

If no or very little data of similar ships is available, or if the ship type is uncommon so there are no existing empirical formulas, it is challenging to use the empirical design method. The naval architect in this case is forced to design a ship from scratch using estimations, calculations, and simulations. An efficient way to do this is by utilizing a parametric 3-D model connected to simulation software. General design knowledge and design experience are used to set up this 3-D parametric model. Examples of the parametric modeling approach can be found in e.g. [100, 121, 149]. Typically, after a parametric model is set up, the designs are optimized for their economical and physical characteristics with optimization algorithms [149].

A second, more automated, parametric design method has recently been proposed by Charisi et al [37]. In this work, it is shown that knowledge-based engineering is a good option when designing a ship when not enough similar ship data is available.

With knowledge-based engineering, general multidisciplinary knowledge is translated into individual product models/building blocks that represent a small part of a vessel. These product models are then used, scaled, and combined into an entire ship design in an object-oriented way.

3.1.3 Radical Design Choices

In order to make more radical design changes with the empirical or simulated design method in a short time and to manage the involved risk, a holistic ship design method is needed to assist the naval architect in making informed decisions. Ship designs can be optimized much more efficiently in a holistic manner where all ship design aspects are considered simultaneously. In recent years, parametric models have been used in combination with optimization algorithms to explore the design space early in the design process, enabling engineers to gain better insight into the design problem, reducing the probability for design changes later, and increasing the chances for optimal cost and operational performance [86, 149].

3.2 Holistic Ship Design Optimization

When optimizing a ship all disciplines should work together to come to an optimal solution. This solution however is often hard to find since there are many choices to be made. As mentioned earlier, in practice, they are made based on experience and heuristics while they should be made based on estimated performance data. Fortunately, holistic optimization is slowly being adopted in the design industry.

Holistic optimization has been applied in different fields of engineering. For example, it has been studied before in ship design optimization by Papanikolau et al. [113]. It has also been used to optimize aircraft designs [5] and passenger cars [132]. Similar techniques have also been used to optimize plant-wide production processes [36]. All these optimization problems have in common that they are solved by taking a holistic, integrated perspective while solving the problem with an optimization algorithm.

The holistic ship design approach from this work consists of several elements, namely:

1. Different software packages in which the vessel's Key Performance Indicators (KPIs) are evaluated.

2. An optimization problem definition with a parameterized design, and objective and constraint functions.

3. An optimization framework where the optimization algorithm, the problem definition, and the different software packages are coupled.

Formulating a ship design problem is however not always straightforward, therefore several guidelines are defined.

3.2.1 Ship Design and Evaluation Software

There are various maritime software packages in which ship design aspects can be defined and KPIs can be evaluated. The C-Job Design Circle [149] in Figure 3.2 illustrates this.

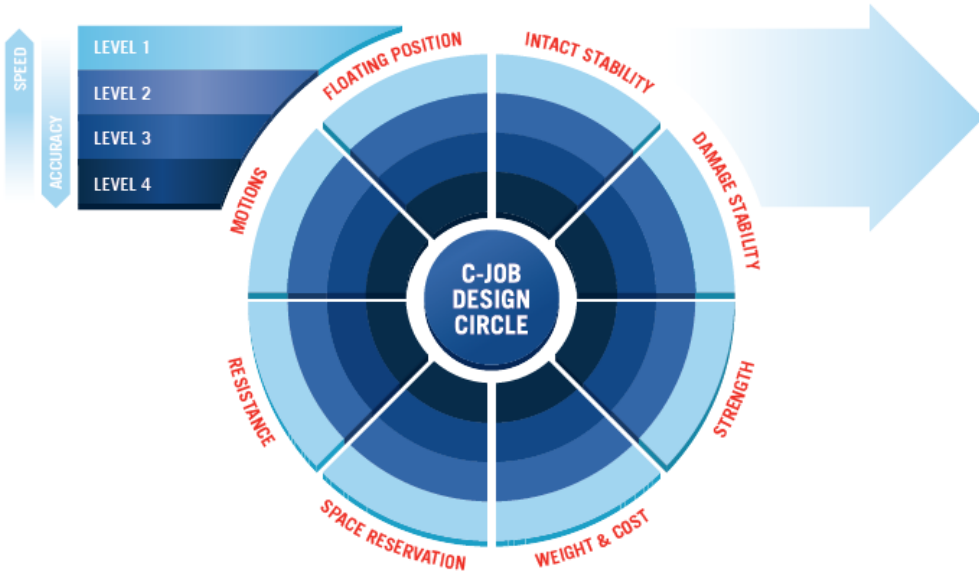


Figure 3.2: The Accelerated Concept Design Circle illustrates the method where all KPIs are considered in every iteration. The level of accuracy of the design simulators can be picked and mixed by the user. High-fidelity simulations that deliver highly accurate estimations require more computational effort compared to low-fidelity approximations which are therefore way faster.

In the design circle, the following design aspects are highlighted: floating position, intact stability, damage stability, strength, weight & cost, space reservation, resistance, and motions. As can be seen in the design circle (Figure 3.2), there are multiple levels of accuracy per design aspect. A high level (level 4) has a higher accuracy and requires more computational effort compared to a low level (level 1) that is less accurate and

is computationally cheaper. As an example, level 1 to level 4 are given for resistance at service speed (also sometimes referred to as power requirement):

Level 1 Is based on empirical design methods, is very coarse, and typically consists of a regression line that is used to estimate the resistance. This regression line is fitted on a dataset from a set of existing vessels that have a similar desired operational speed and main dimensions as the vessel to be designed. This regression line is then used to estimate the required power of the new to-be-designed ship. Estimation with this method requires almost no computational effort but is also very coarse since minor changes in the design (e.g. a change in bulb size) do not reflect in any change in estimated resistance. This method can however be a good guideline for helping to choose the main dimensions of the vessel. This method for required power estimation in combination with verifying if the selected set of vessels is really operating at the design speed can be accurate in cases where the to-be-designed vessel is very similar to the existing vessels.

Level 2 The empirical Holtrop & Mennen method [79] is used to estimate the power requirement. The Holtrop & Mennen method is a well-known resistance estimation method. It considers factors like hull shape, dimensions, and speed to calculate different resistance components, including residuary, wave-making, viscous, and appendage resistance. Making estimations with this method however requires a bit more effort compared to level 1 because the Holltrop & Mennen method requires more information about the hull shape (Block Coefficient, Longitudinal center of buoyancy, Prismatic Coefficient, bulb size, U versus V shape frames), dimensions and speed. The downside of this method is that it expects the hull to be well-faired and hydrodynamically optimized, therefore the results are for an optimized and well-faired hull. However, often in the initial design stages where the Holtrop & Mennen estimation method is used, the hull is not yet optimized and faired. The Holtrop & Mennen estimated resistance therefore does not reflect the resistance of the current hull but what the resistance is after the hull is optimized and faired. Furthermore, the method is not suited for vessels that are operating at very high speeds, or have an odd length-over-breadth ratio and are therefore very different than the vessels in the database used to develop the Holtrop & Mennen method.

Level 3 A potential flow solver like the one from Tahara et al [140] or Rapid from Raven [122] can be used for hull resistance and drag estimation. A typical potential flow solver is a computational tool used to analyse how water flows around

a ship's hull. It simplifies the Navier-strokes equations by omitting viscosity (it does not take vortices, friction, or flow separation into account) and assuming incompressible and irrational flow. The results from a potential flow calculation consist of the wave-making resistance, the dynamic trim and sinkage, the wave pattern, pressure distribution, and the flow lines on the hull. Therefore, potential flow solvers could be used to optimize the bow of the ship but are less feasible for optimizing the aft ship where the thickness of the boundary layer is significant and flow separation and viscous forces are important. Potential flow solvers can often excellently be used in initial hull shape design stages. A simple potential flow calculation typically requires a few minutes.

Level 4: A Reynolds Averaged Navier-Stokes Equation (RANSE) analysis, often called Computational Fluid Dynamics (CFD) analysis, is used to analyze the fluid flow behavior around the hull. This is the most accurate level of the design circle and requires the most computational effort. Typically CFD analysis is done with commercial software like STAR-CCM+ [135] or open source software like OpenFOAM [168]. The results from a RANSE analysis are typically pressure drag, friction drag, dynamic trim, and dynamic sinkage, the pressure on the hull, the shear stress on the hull, wave pattern, and flow lines on the hull. The same results (with similar accuracy) can be obtained by executing model tests in a basin. It is typically used to optimize the hull in one of the final stages of the design where the final details are to be sorted out about the fairing, and the ideal appendage location and size. A simple resistance calculation in calm water typically requires 20 minutes on a high-performance computer with 144 cores. Very detailed full-scale ship simulations of a complete model in realistic operating conditions and turbulent flow would require much more computational resources.

Besides the resistance and power requirement for ship design, the other design aspects such as stability, strength, weight, motions, floating positions, and space reservations can also be defined and computed at different levels of accuracy. The speed and consequently the level of accuracy at which the design variants are evaluated can be picked and mixed by the user. This means that if resistance is considered very important a RANSE calculation can be selected, while strength for example can be calculated with a simple longitudinal bending moments check, and if for example the space reservation is guaranteed to be feasible by the parametric model, then evaluation for this KPI can be completely skipped. The level of accuracy and duration of the

evaluation influences how many evaluations can be done in the often short available time in the early design stages. A simple volume check of a cube shape tank can be executed thousands of times in a very short amount of time while a RANSE analysis is more expensive in terms of computational cost and can therefore be computed much less often.

The diverse nature of ship design tasks leads to a wide variety of specialized software packages. Noteworthy commercial software packages used in this work are:

Rhinoceros 3D or Rhino in short, is primarily used in ship design to create 3D models of vessels. This allows designers to shape hulls, parameterize hulls, fair hulls, and define tanks and rooms. The programming, import, and export functionality of Rhino makes it convenient to integrate Rhino with other analysis tools for structural and hydrodynamic assessments.

NAPA software is used to model, parameterize, and evaluate any type of vessel. The NAPA software is mainly used to define the space reservation with room and tank arrangements and evaluate the damage and intact stability criteria of vessel designs. Other aspects that can be calculated with NAPA are seakeeping, floating position, and structural aspects for weight calculations. The programming capabilities in NAPA make the software modular, easy to adjust, and give users the possibility to establish a connection to other software packages.

DELFTship maritime software, is a visual hull modeling and stability analysis program that allows. If modeled correctly, all the data required for damage and intact stability can be calculated together with the bending moments and shear forces. Just like in other design software packages, it is possible to program custom scripts in DELFTship so that it can be coupled to other packages.

STAR-CCM+ is powerful computation fluid dynamics (CFD) software package developed by Siemens. It is mainly used for hull hydrodynamics, to assess the propulsion requirement for the vessels. These calculations are crucial when optimizing hull shapes for fuel consumption reduction.

All these software packages require commercial licenses and if multiple simulations are to be run in parallel, multiple licenses are required during the entire evaluation time. The commercial license requirement together with available computing resources typically drastically limits the number of allowed evaluations during the optimization process. Therefore, it is important to pay attention to how many evaluations are possible in the available time and how the number of evaluations can be decreased.

3.2.2 Optimization Problem Definition Guidelines

There are some guidelines to follow when setting up a ship design optimization problem (or any optimization problem for that matter) in a holistic manner. The guidelines deal with the parameterization of the solution, and how to formulate the constraint and objective functions.

Parameterization Guidelines

In the parameterization stage of ship design, the naval architect can use imagination and creativity to create an innovative and new design that should be optimized. Important to remember when parameterizing and later optimizing a ship design model is that the best vessel that can be found by the optimization algorithm is dependent on the parameterization of the vessel. So if the parameterization is bad, the results will be bad. To give two illustrative examples:

1. If the goal is to make an as light as possible vessel, but the parameterization doesn't allow for vessels that are smaller than the design that is parameterized, it is very unlikely that a lighter vessel can be found with this parameterization.

2. If one wants a design with as small as possible hull resistance at design speed, parameterizing the interior will probably only help a tiny bit in reducing the resistance because it has a marginal influence on the floating position. It would be much better in this case to parameterize the hull.

Therefore, the ship should be parameterized in such a manner that the performance of the vessel also depends on the decision parameters that are defined in the parameterized model. Defining decision variables that only marginally influence the constraint and/or the objective values will only have marginal influence so they can better be neglected or be regarded as a constant throughout the whole process.

Secondly, it is important that after all the decision variables have been set, the geometry is formed accordingly. After this, the geometry should not be changed anymore (for example to meet a constraint) until a new combination of decision variables is chosen. This way, the input truly corresponds with one unique geometry. After the geometry has been evaluated, it will be these decision parameters that define the geometry and the corresponding constraint and objective values. This approach can sometimes result in infeasible solutions since some combinations of decision parameters will result in a geometry that violates one or more of the constraints. This is not a major problem since optimization algorithms directly learn the relationship between

the decision parameters of the solutions and the constraint and objective scores and therefore will learn to propose better combinations in the next iterations. Because the relationship between the parameters and the constraint and objective functions is learned by the optimization algorithms, the number of parameters can also have a significant impact on how many function evaluations are required to find (a set of) optimal solution(s).

Constraint Function Guidelines

A constraint function is there to check if the solutions comply with the hard boundary conditions set by the stakeholders. A guideline to take into consideration when defining constraint functions is to make sure that the output of the constraint function is real-valued and preferably continuous. Bad practice is a simple Boolean constraint value corresponding to the constraint being violated or met. A real-valued constraint function is important since optimization algorithms can learn by how much the constraints are violated or met. This information can be learned by the optimization algorithm and can be crucial since, more often than not, the optimal design configuration is located on the border of the feasible area. The constraints are defined as function inequalities: $g_i(\mathbf{x}) \leq 0$ where $\mathbf{x}$ are the decision parameters and $g_i()$ is one of the evaluation functions used to evaluate one of the m constraints. Note that here 0 is the feasibility boundary, exactly how it should be formulated according to our problem formulation in Equation 2.1. However, in reality, a typical constraint function $g'(\mathbf{x})$ consists of three parts: a continuous function g' which takes input arguments $\mathbf{x}$ that represent the decision variables that can modify the solutions, and a constraint boundary c that represents a maximum or minimum value for the constraint. These three parts taken together form for example the inequality constraint $g'(\mathbf{x}) - c \leq 0$. This inequality constraint can then be rewritten to its general form: $g(\mathbf{x}) \leq 0$. When constraints have a minimum value they can be rewritten without loss of generality to a function with a maximum value by multiplying by -1. Note that the function $g'(\mathbf{x})$ does not need to be a function that can easily be written by mathematical formulas. The function $g'(\mathbf{x})$ can also be a software program that calculates any of the KPIs.

In some design cases, also contain hard equality constraints. Sometimes, they can simply be avoided by adjusting the parameterization of the design, while in other cases an equality constraint is inevitable. In these cases there is an equality constraint $h(\mathbf{x}) = 0$, it should be replaced with two inequality constraints. This way, there is no need to resort to special equality constraint handling methods. Good practice is however to add a small margin (ϵ) around the constraint boundary so that in case there

are tiny numerical instabilities this does not influence the feasibility of the solutions.

$$h(\mathbf{x}) = \begin{cases} g(\mathbf{x}) - 0.5\epsilon \leq 0 \\ g(\mathbf{x}) + 0.5\epsilon \geq 0 \end{cases} \tag{3.1}$$

Objective Function Guidelines

The objectives are measurable KPIs that the stakeholders look to improve as much as possible, it can be anything as long as it is numerically measurable and dependent on the decision parameters. When defining the objective functions, one should define a real valued, and again preferably continuous, function that only represents one objective at a time. This is preferred above a weighted sum of all objectives. Similarly, one should not add penalties to the objective score if the design violates any of the constraints. This way, the objective function remains smooth and the true optimal solution can be found more easily by optimization algorithms. Objectives are typically conflicting and non-commensurable. Therefore, often one objective can not be improved without sacrificing another objective. As a consequence, there is not one perfect solution but a set of non-dominated solutions that form the Pareto optimal set. The definition of the Pareto optimal set is used as described in 2.9.

3.2.3 Optimization Framework

The different software packages, the formulation of the objectives, constraints, and parameterization come together in the holistic Accelerated Concept Design (ACD) framework [149]. Because every aspect of the ACD framework is fully automated, the designs can easily be varied and optimized towards, for example, the total cost of ownership. By using any of the surrogate-assisted multi-objective optimization algorithms from Chapter 5, the whole design space can be considered which makes it more likely that a globally optimal solution will be found.

3.3 Illustrative Ship Design Optimization Problem

In this section, a example ship design optimization case is presented for illustration purposes. In practice, every ship design optimization problem is different but in theory, it is nothing more than defining a ship design optimization problem and solve it. Note that the design and parameterization stage is where the naval architect can use their imagination and creativity to come up with new innovative designs.

To illustrate this process of defining and optimizing a vessel, a Trailing Suction Hopper Dredger (TSHD) designed by C-Job Naval Architects is described and optimized. This TSHD will be optimized to reduce total cost of ownership. Total cost of ownership consists of Captial Expenses (CapEx) and Operational Expenses (OpEx). The original design with the annotated decision parameters is shown in Figure 3.3.

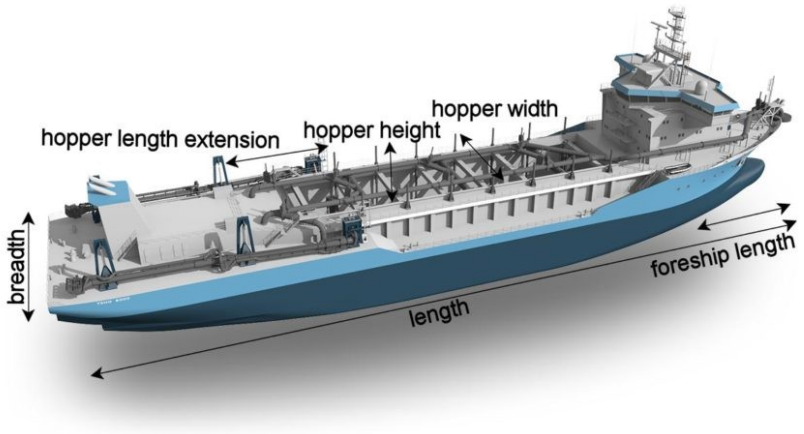


Figure 3.3: Trailing Suction Hopper Dredger with design parameters annotated.

3.3.1 Parametric Geometry

The first step is to generate a parametric geometry of a vessel. The geometry depends on decision variables which are numerical quantities for which values can be varied in the optimization process [41]. The decision variables are Free-Form Deformation (FFD) [43] parameters to modify the hull and parameters that are used to rearrange the bulkheads in the vessel.

Repositioning Bulkheads

The rooms inside the hull can easily be parameterized. This is very useful since it will be the rooms and their loading that influences the floating position, intact stability, damage stability, draft, heel, trim ect. Parameterizing rooms can be done by moving locations of bulkhead along the x, y and z axis. As an example, the bulkheads that define a hopper of a TSHD are moved along the x-axis in Figure 3.4. This way, the hopper can be made smaller, larger, or the location of the hopper can completely be

moved.

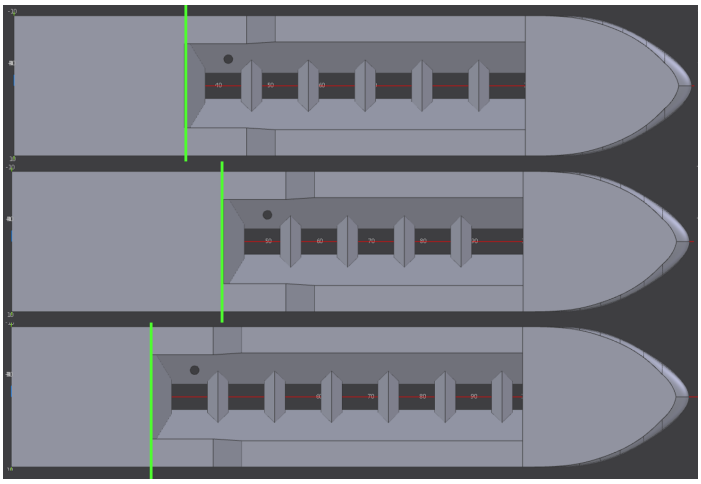


Figure 3.4: Bulkheads moved forward and backward to define different hopper lengths. The bulkhead positions are marked in bright green.

Good practice is to make bulkheads dependent on each other. To illustrate this the following example is given: if bulkhead 1 is located at frame 0, the room between bulkhead 1 and bulkhead 2 is 7 frames long, then bulkhead 2 is located at frame 7. The room between bulkhead 2 and bulkhead 3 is 10 frames long. Then bulkhead 3 is located at frame $7 + 10 = 17$. In case the length of room 1 is increased by x frames, then move bulkhead 2 to $7 + x$ and move bulkhead 3 to $7 + x + 10$. In case the length of room 1 is decreased by x frames, then move bulkhead 2 to $7 - x$ and move bulkhead 3 to $7 - x + 10$. This dependency is displayed in Figure 3.5.

Free-Form Deformation

When using automated optimization tools for altering the hull form of the vessel, a challenge may arise with keeping the fairness (and other important characteristics) of the hull surface. Using the Free-Form Deformation technique, the hull form change undergo rather radical changes in a robust way. The optimization by FFD is done by applying a lattice (box) around the surface or part of the surface which is to undergo transformation. The lattice is then deformed by translating it's vertices. The control points of the circumscribed surface will consequently undergo transformation leading to a new surface geometry. An example of a FFD applied on the fore- and aft-ship of a hull is illustrated in Figure 3.6.

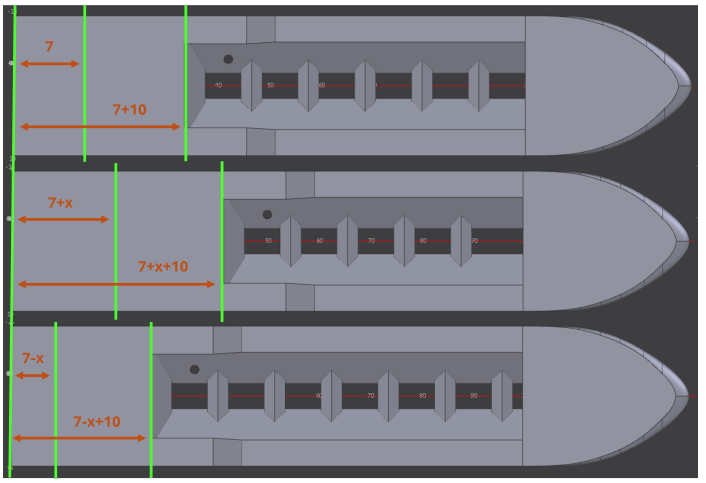


Figure 3.5: Bulkheads moved forward to make rooms larger or smaller. The bulkhead locations are marked in bright green and the red arrow lengths indicate the distance between the bulkheads.

Trailing Suction Hopper Dredger Parameters

The TSHD design that is parameterized for this illustrative example is presented in Figure 3.3. The model has the following six decision variables: $\Delta_{breadth} \in [-1.6, 3.4]$, $\Delta_{length} \in [-2.8, 9.8]$, foreship length $\in [16, 22]$, hopper length extension $\in [5, 9]$, hopper breadth $\in [5, 9]$, hopper height $\in [12, 16]$. The ship length and ship breadth are annotated with a Δ because these variables are used to apply a FFD on the initial hull. The other parameters are all bulkhead locations which can be moved and/or extended along the x, y, and z direction.

3.3.2 Ship Design Constraints

For the TSHD some quite specific constraints have to be set in order to create sensible design variations that meet the stakeholders' demands and which do not violate any of the regulations. These constraints mainly focus on the hopper volume, stability criteria, space reservation, fuel capacity, water ballast requirement, and the dredging area where the vessel will be dredging and dumping. The details of these constraints are defined below:

Hopper volume: The hopper where the dredged soil is dumped in, should at least be 8000 cubic meters to meet the client's demand. The inequality constraint for the hopper is therefore: $8000m^3 - VOL(hopper) \leq 0$.

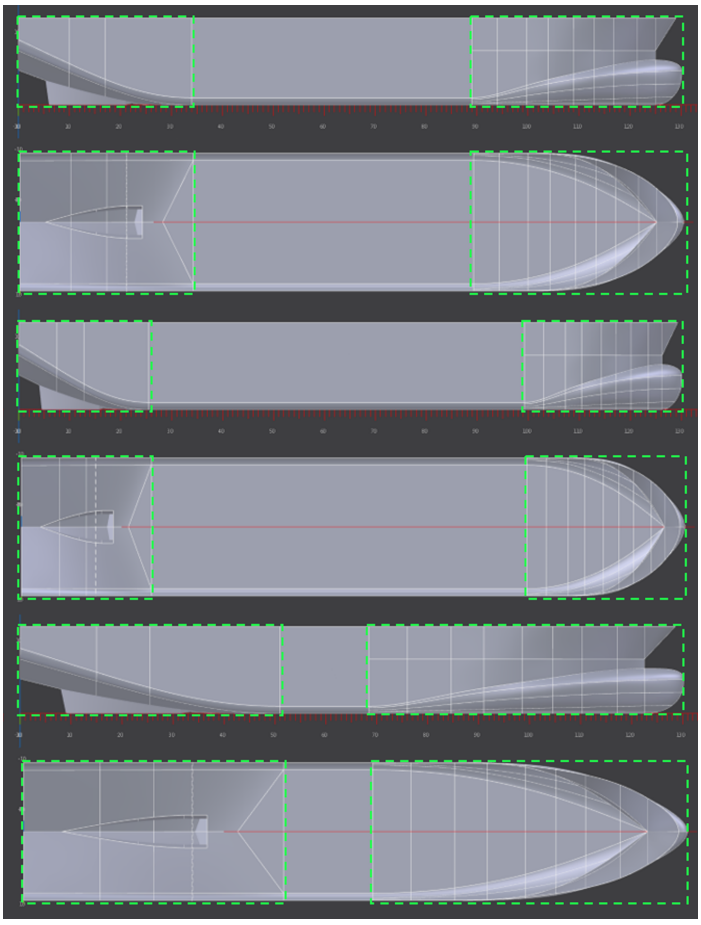


Figure 3.6: Free Form Deformation applied to fore- and aft-hull.

Fuel capacity: The design variation has 9200 kW installed power, the fuel type is Marine GasOil (MGO), and the design variation should at least be able to sail for 21 days. Therefore a relatively conservative fuel capacity constraint is defined: $820m^3 - VOL(fueltanks) \leq 0$.

Propulsion Engine: The engine room should at least be large enough to accommodate the required space and volume of the propulsion engine. Therefore, the propulsion engine and the engine room are modelled and intersected. The volume of the intersection of the two rooms should be at least be equal to engine volume, if this is not the case, the engine does not fit in the engine room. $VOL(Engine) - VOL(EngineRoom \cap Engine) \leq 0$.

Dredge Pump: The procedure for the dredge pump is the same as for the Propulsion Engine. Therefore the constraint for the dredge pump is the following: $VOL(Pump) - VOL(PumpRoom \cap Pump) \leq 0$.

Accommodation Space: The accommodation on the design variation should host 29 crew members. For 29 persons an estimation of the required space is made, this estimation resulted in a required 1100 cubic meters accommodation space. Therefore the inequality constraint for the accommodation is the following: $1100m^3 - VOL(Acc) \leq 0$.

Draft: The design variation should be able to sail in shallow waters, therefore the draft T should not exceed 7 meters when fully loaded. The corresponding inequality constraint is therefore: $T - 7m \leq 0$ such that when the design variation is fully loaded it is still capable of sailing in shallow waters.

Fore peak bulkhead: The fore peak bulkhead, also referred to as collision bulkhead, should be positioned according to the International Convention for the Safety Of Life At Sea (SOLAS) [110]. The constraint from SOLAS is as follows: $\min(0.05L, 10m) \leq d \leq \max(0.08L, 0.05L + 3)$, where L is ship length and d is the collision bulkhead position. Rewritten and separated into two inequality constraints, this rule then gives the following inequalities: $\min(0.05L, 10m) - d \geq 0$, and $\max(0.08L, 0.05L + 3) - d \leq 0$.

Stability Criteria: The stability requirements are evaluated by expressing the applicable regulatory requirements as a meta-centric height value GM_c. Given any possible loading condition of the ship, the obtained meta-centric height GM_o of the design variation should at least be larger or equal to the prescribed metracentric height value to pass all the stability requirements. The stability function inequality therefore is the following: $GM_o - GM_c \leq 0$.

Trim: Given different loading conditions with the hopper empty and the hopper filled, without using water ballast, the trim varies. Because the TSHD at hand is a ballast-less design, it is necessary to keep the difference in trim within practical limitations. The trim has therefore been limited to a maximum difference of 2.1 meter. The inequality constraint for trim is therefore described as follows: $(\max(trim_{lc}) - \min(trim_{lc})) - 2.1m \leq 0$.

Heel: The same procedure as for trim is followed for heel. The difference between the maximum and minimum heel is not allowed to be larger than 0.2°. The function

inequality is therefore the following: $(\max(heel) - \min(heel)) - 0.2^\circ \leq 0$

Strength is also a critical constraint, but because the steel weight is also optimized, it is made sure that only design variations are generated that comply with the strength regulations, therefore a strength constraint is not added. All the constraints defined in this section are evaluated with custom macros written in the NAPA basic software. Evaluating one design for all constraints requires a couple of minutes.

To show the severity of the constraints, an experiment was conducted to check how much of the design space was feasible. To get an indication 200 random design variations are generated and evaluated. 24% of the 200 design variations were feasible.

3.3.3 Ship Design Objectives

The objective in this illustrative example is minimizing the total cost of ownership of the dredger. The total cost of ownership of this particular dredger is calculated by the client, the ship owners, and the operators of the dredger. The main cost drivers of the dredger are therefore calculated to enable stakeholders to evaluate a given Pareto optimal solution and come to a decision on which one of the Pareto optimal solutions should be selected for consecutive design stages.

In this particular case, the total cost of ownership can be separated into building costs, operational gains, and operational expenses. Building costs are largely driven by the material cost and consequently can be directly linked to the weight of the vessel. Operational gains can be defined as ship speed and moved cargo, which are both fixed for this optimization case, and operational cost is largely driven by fuel consumption, maintenance, and crew costs. Minimizing the total cost of ownership can thus be achieved by minimizing the steel weight while also minimizing the resistance at service speed. These objectives are a classical example of conflicting ones, a long and slender design variation will have a smaller resistance factor and a higher estimated steel weight compared to a wider shorter variation of the same vessel.

The resistance of the different design variations is estimated with a potential flow solver [140]. Because of the nature of a potential flow solver, the mesh describing the hull shape has been idealized. The potential flow code cannot provide absolute resistance values but is suitable for comparing the resistance of different design variations.

The steel weight is more complex since the lightship weight, maximum bending moments, and loading conditions should be in balance with each other. Therefore, for every design variation a minimum steel weight needs to be estimated that meets the maximum bending moment requirement while keeping Equation 3.2 and Equation 3.3

in balance line in Equation 3.4:

$$\Delta_{actual} = L \cdot B \cdot T \cdot Cb \cdot \rho \tag{3.2}$$

$$\Delta_{required} = DWT + LSW \tag{3.3}$$

$$\Delta_{required} = \Delta_{actual} \tag{3.4}$$

where Δ is the actual and required displacement, L is Length, B is Breadth, T is Draft, Cb is Block-coefficient, and ρ is the water density, DWT is the Dead Weight, and LSW is Light Ship Weight.

3.3.4 Optimization of Ship Design Problem

In the illustrative example, a potential solver is used to estimate the resistance, and NAPA macros are used to compute the constraints and the steelweight objective. This results in an optimization problem that requires 5 to 10 minutes per design evaluation. In other cases where RANSE analysis are done for resistance calculations these 5 to 10 minutes however can quickly grow to hours. If the goal remains to find good solutions on the Pareto frontier in a limited amount of wall clock time, special attention should be given to the selection process of the optimization algorithm. How to limit the number of required function evaluations, how to evaluate solutions in parallel, and how to deal with expensive and inexpensive functions is described in more detail in Chapter 5. The obtained results on this illustrative ship design problem are reported in Chapter 6 together with other Real World Application results.

3.4 Conclusions and Future Work

In this chapter ship design methodologies, parameterization, and evaluation methodologies are described that together form a typical ship design optimization problem. After identifying good parameterization practices, different evalulation methodologies, and guidelines on how to set up the optimization problems the holistic Accelerated Concept Design framework is set up. The parameterization of the design in the holistic framework is the most important part of the ship design optimization process. This is because the final designs are only as good as the parameterization allows. If there

is enough flexibility in the parameterization that allows for improvements in the objectives and that effectively deals with the constraints, a new and better design can be generated. Secondly, the level of accuracy of the evaluations of the constraints and objectives is important. The accuracy (and evaluation software) significantly influences the required computational effort and determines together with the available commercial licenses available the total evaluation budget. The evaluation budget is critical since enough evaluations are required for optimization algorithms to converge. A large required evaluation budget is often problematic since in the conceptual design stage, the lead time is often limited.

For future work, more research is needed to find out what the best parameterization setup is for which design case. This will however require the joint effort of optimization experts, naval architects, and structural and hydrodynamic engineers.

Chapter 4

Empirical Design Optimization Approach

As described in the problem characteristics of Chapter 3, the quickest and most coarse way to estimate the main particulars of a new design is by using and learning from data of reference vessels. This already is part of the answer to research question 3: *How can data be used to find feasible Pareto efficient ship design solutions?* In the current chapter, a new method is introduced that uses reference data, machine learning algorithms, and an optimization algorithm to find suggestions for the main particulars and KPIs of new ship designs. With this new method, designers can make more informed decisions in the preliminary design phase where very limited information is available and decisions need to be made in a short amount of time. However, it is in the preliminary design phase where the most influential decisions are made regarding the global dimensions, the machinery, and therefore the performance and costs. In this chapter, it is shown that a machine learning algorithm trained with data from reference vessels is more accurate when estimating key performance indicators compared to existing empirical design formulas. Finally, the combination of the trained models with optimization algorithms proves to be a powerful tool for finding Pareto-optimal design solutions from which the naval architect can learn. Although the application domain of this chapter is ship design, the approach can also be transferred to other application domains.

4.1 Introduction

In the preliminary design stage, more knowledge should be used when making decisions. With the method described in this chapter, naval architects are better supported by data to make design decisions instead of relying only on instincts, knowledge, and experience. The decision support is in the form of machine learning models which can be used to validate ideas, assumptions, and design variations. This helps the naval architect avoid innovation risks and to find better design variations. On top of this, without much additional effort, the naval architect can use the trained machine-learning models in combination with an optimization algorithm. This optimization algorithm can then be deployed for searching advantageous and competitive design variations. The only requirement for the reference optimizer that is proposed in this chapter to work adequately is enough relevant ship data and a good design problem setup.

4.2 Data Description for Reference Studies

The solution proposed in this chapter utilizes the power of empirical design methods in combination with parametric optimization. However, for empirical design method to work properly, data is needed. Fortunately, a lot of data services have become available for the maritime industry. The most prominent ones are:

World Fleet Register is a ship data and intelligence platform from Clarksons Research with data about ship earnings, vessel parameters, and new-build data [40].

Sea-web collects static ship data of existing and even scrapped ships and tracks vessels worldwide [136].

BRL shipping consultants has a subscribers area where reports are available about the active fleet and about newly built vessels [26].

Marine Traffic is a platform that allows even without logging in to obtain the location of vessels plus general static ship data [98].

AISHub is an AIS data sharing platform where you can get access to global AIS radar stations when you join with your own AIS antenna [2].

All this static and operational data has been collected and aggregated into more than 100 particular data fields per vessel and a large database with historical locations of ships. Examples of collected data fields are: Length, breadth, draft, block

coefficient, light ship weight, dead-weight, maximum continuous rating of the engines, maximum speed, but also more ship specific data fields for specific ship types such as: Bollard pull, passenger capacity of urban transportation vessels, number of car lanes, crane capacity for offshore vessels, hopper volume of dredgers, and ice class qualification.

This data can be used in a reference study in the preliminary ship design process since design trends can be visualized, design trends can be learned, and gaps and competitive advantages in the market can be found.

4.2.1 Visualizations

After the relevant parameters for a vessel type have been selected, the parameters can be summarized and plotted. When three parameters are relevant it is still possible to visualize them in two dimensions. As an example, the dead-weight and moulded breadth together with the Twenty-foot equivalent Unit (TEU) capacity is given for a set of container vessels in Figure 4.1.

However, it is often the case that more than three parameters are relevant in the preliminary ship design stage, which makes it challenging to visualize. To still be able to investigate a selection of ships or design variations with more than three parameters, parallel coordinate plots can be used from Section 2.5.2. In Figure 4.2, a parallel coordinate plot is made for several hundred container vessels with a length between perpendiculars between 175 and 200 meters.

Interpretation of Visualizations

As can be inspected from Figure 4.1, the moulded breadth has a maximum of 32.4 meters, a well-known maximum width for ships to still be able to pass through the Panama Canal. This maximum moulded breadth can also be seen in Figure 4.2. Moreover, it is now also possible to simultaneously see all other relevant parameters of the container vessels. For example, the limited draught for the majority of vessels in this selection is smaller than or equal to 12 meters, also an important Panama Canal dimension. Besides this, one can simultaneously see the conflicting relationship between block coefficient (Cb), and Maximum Continuous Rating (MCR) and their influence on service speed. The vessels with a high block coefficient, and small maximum continuous rating, also have a slow service speed and vise-versa. When designing new vessels, these plots can be very helpful for the designers.

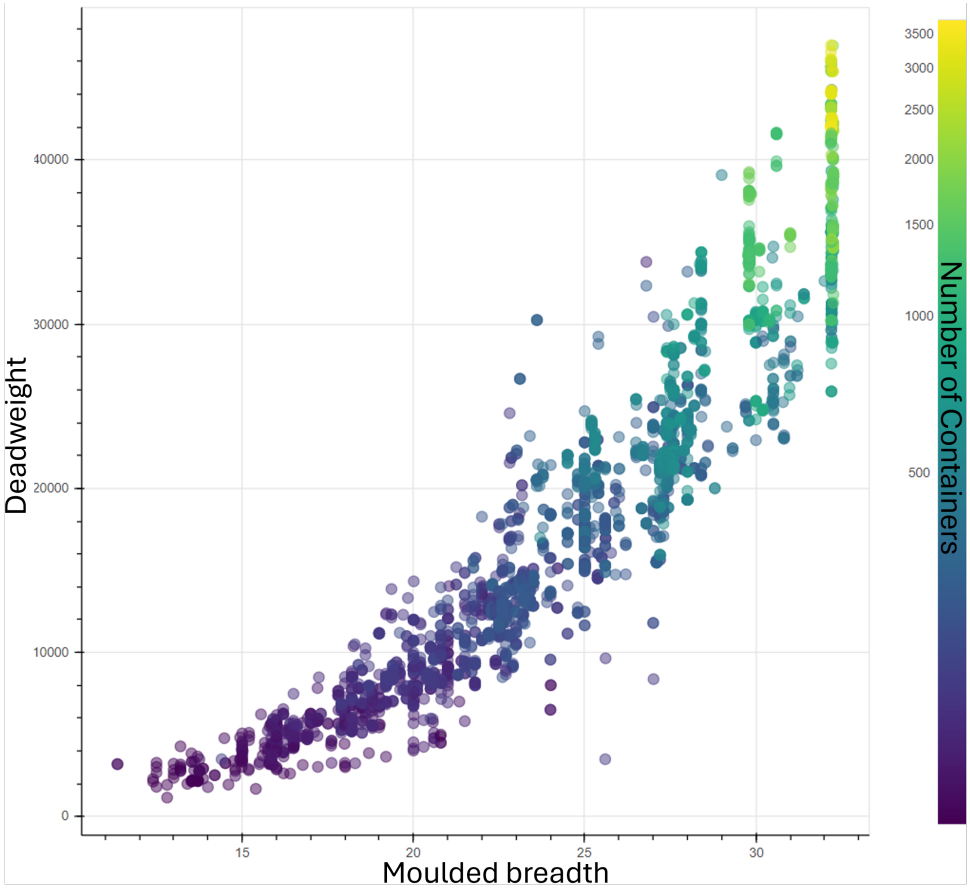


Figure 4.1: Container vessels color-coded by container capacity.

4.2.2 Data Pre-processing

Preliminary data analysis showed duplicate vessels and vessels which are very similar. To make sure that specific vessels are not over-represented, but still enough data is available, data pre-processing must be done. The pre-processing consist of three steps and is done so that machine learning algorithms can be trained with *cleaner* data.

1. All except for one of the vessels with exact duplicates must be deleted. Ships are considered to be duplicates if their *gross tonnage, length between perpendiculars (Lbp), breadth overall (Boa), draught (T)*, and *MCR* are equal.

2. All but one vessel out of a series of sister vessels are deleted. If the earlier

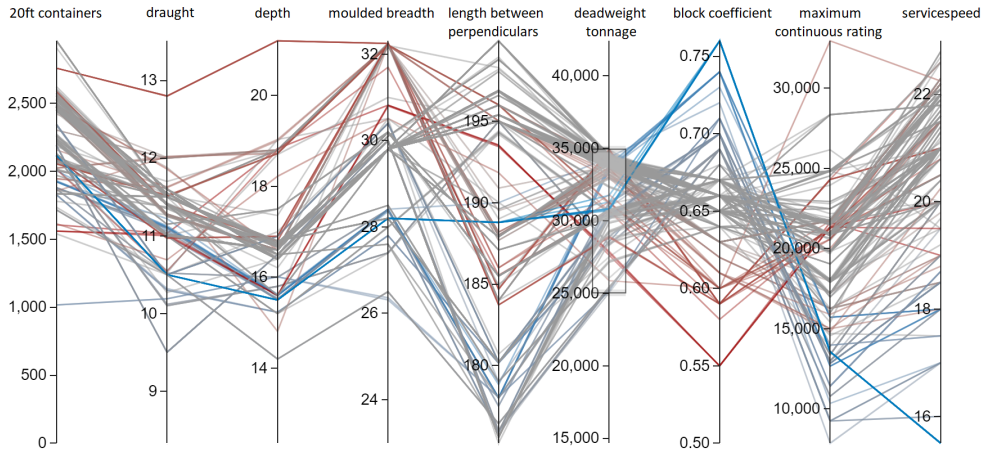


Figure 4.2: Parallel coordinate plot of container vessels color-coded by block coefficient, and a deadweight tonnage selection between 25000 and 35000 tonnes.

mentioned variables are all within 1 percent of each other, the vessels are marked as too similar.

3. A second degree polynomial and interacting features are created. The two degree polynomials and interacting features of the example $[a, b]$ would be: $[a, b, ab, a^2, b^2]$.

Reasons for deletion of duplicates and very similar vessels are to prevent the potential over-fitting of machine learning models. If a series of sister vessels would be present, the machine learning model would automatically put more weight on the sister vessels compared to one unique vessel. A second argument to delete sister vessels is, once a machine learning model has learned from a vessel, a second sister vessel does not add much knowledge but will only add computation and training time.

The second degree polynomial and interacting features are created to generate more potentially interesting features from the design parameters that are known. This way, the machine learning models have more features to learn from which potentially leads to more accurate results.

4.3 New Empirical Design Methodology

This section describes how the new empirical design method is used in combination with an optimization algorithm in the so called reference optimizer. As mentioned in the related work section, it is often the case that the empirical design equations are not available for a specific ship type or that the available equations are outdated. This is unfortunate since designing ships with wrong or outdated design equations will most likely not lead to optimal decisions. The empirical design equations are therefore replaced with machine learning models. These machine learning models make sure that it is no longer need to solely depend on predefined equations or the experience and knowledge of naval architects.

Machine learning models are used to learn the relationships, similarities, and trends between hundreds of data points. However, for machine learning models to work properly, the relationship between the dependent and independent variables need to be learned. The dependent and independent variables are chosen by the naval architect. The machine learning models learn the relation between the independent and dependent variables in the training phase. After the training phase, the trained machine learning models are coupled to an optimization algorithm that can exploit the trends learned and search for optimal design configurations that outperform the existing designs.

4.3.1 Setup Design Challenge

For the machine learning algorithm to work well a design challenge should be set up by the user. The design challenge consists of three parts, the design variables, the constraints, and finally the objectives as described earlier in Section 3.2.2.

Design Variables are set up by choosing the design parameters that have a significant influence on the final design and which are allowed to vary. The allowed variations in the variables are controlled with a user-defined lower and upper limit. However, the limit can not be smaller or larger than the smallest and largest ship in the collection. Examples of a set of design variables are: *Length between perpendiculars (Lbp), draft (T), Breadth overall (Boa), block coefficient(Cb)*, and *service speed (V)*.

Constraints are also set by the user. The design constraints are typically hard limitations or strong wishes for the to-be-designed ship. Examples of constraints

are *deadweight (DWT)* capacity of 30,000 tons, a *cargo capacity* of 2000 TEU, a *length overall* smaller than 180 meters, or a *draught* of not more than 12 meters.

Objectives of a ship design are usually the key performance indicators that deal with operational expenses and investments. Ideally, they are as low as possible, however, they most often do not go hand in hand and are most of the time conflicting. Examples of three objectives are: minimizing the *light ship weight (LSW)*, maximizing the *deadweight (DWT)* capacity, and minimizing the *Maximum Continuous Rating (MCR)* of the main engines.

Once the design variables, constraints, and objectives have been set by the user, the relationship between the variables and the constraints and objectives can be learned.

4.3.2 Random Forest Regression

A random forest regression model [25] can be used to learn the relationship between the features and one target variable. A random forest regressor is chosen because it is robust against outliers and overfitting and because it can deal with discrete parameters which comes in handy as the data used for training comes from existing ships and might not always be 100% reliable. In the new empirical design methodology the features are the design variables plus the polynomial features and the target variable is one of the constraints or one of the objectives. Therefore, for each constraint, and for each objective a new unique random forest regression model is trained.

The random forest regression model learns the relation between the features and the target by fitting a multitude of decision trees. One decision tree is fitted to learn the relation between a set of random selected features with the corresponding target values. The data with the random selected features is sequentially greedily split into two sub-samples based on one of the features until the number of samples in the nodes reaches a threshold value. Resulting in an upside-down tree with nodes, branches for splits, and leaves with similar target scores.

Once e.g. 100 decision trees have been trained with the 100 randomly selected feature sets, the random forest is done training. The trees in the forest can be traversed which makes a prediction of the target variable for an unseen combination of feature values possible for each tree. These 100 outcomes of the 100 decision trees are then averaged into a final prediction. The process of making a prediction is visualized in Figure 4.3. Because a multitude of trees are fitted, the random forest regression model is robust against outliers in the training data. However, due to the fact that the final

score depends on the average of all the trained trees, the random forest regression model is not capable of extrapolation.

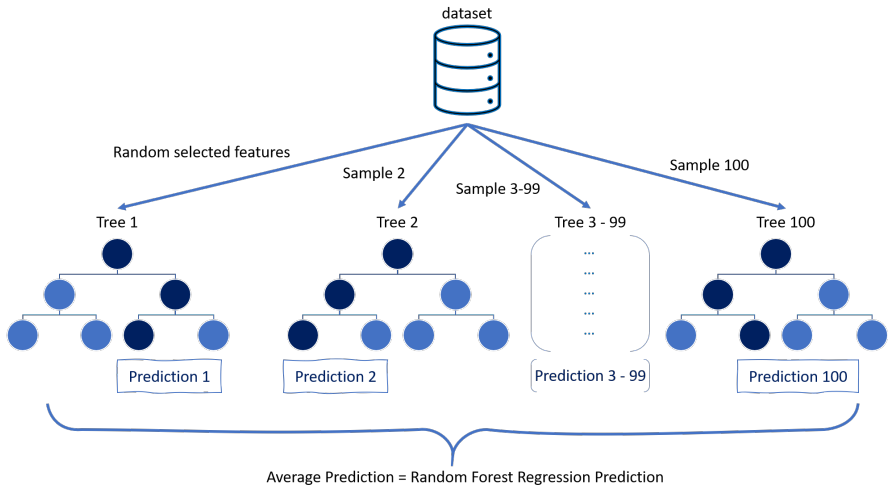


Figure 4.3: Random Forest Regression Model Illustration.

4.3.3 Isolation Forest

In the reference optimizer not only the user-defined constraints limit the search space. The search space is also limited by an anomaly detection algorithm. The anomaly detection algorithm used is named Isolation Forest [94]. Isolation forest is an unsupervised machine learning algorithm that tests how easy it is to isolate certain data points. It does so by recursively splitting the data by randomly selecting a variable and a random split value between the lower and upper limits. If a sample is easy to isolate by randomly splitting the data set, it is marked as an anomaly. A sample that is hard to isolate versus a sample that is easy to isolate is visualized in Figure 4.4.

In practice, this means that in case a design variation is unique and lies outside of the trend, or if the database contains a ship with length by accident reported in feet instead of meters, it is marked as an anomaly. When searching for a new design variation, design variations that are marked as anomalies by the isolation forest will no longer be considered. This is the case because they do not follow the pattern and therefore their prediction is probably incorrect. On top of this, the isolation forest will make sure that the design variations will not exceed the limits, so that the random forest regression model is not forced to extrapolate.

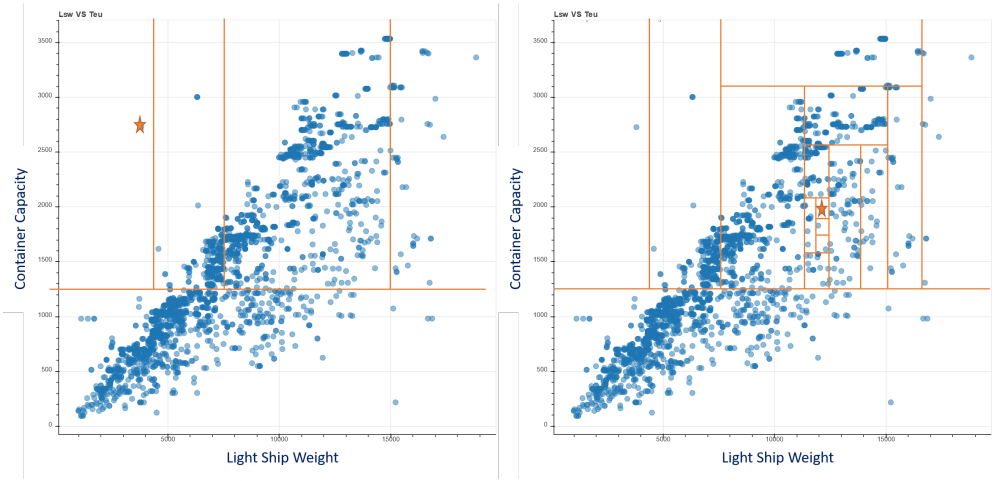


Figure 4.4: Illustration of Isolation forest with easy to isolate sample on the left and a hard to isolate sample on the right.

4.3.4 Design Problem Optimization

The reference optimizer searches for Pareto optimal designs that do not violate any of the constraints. This can be done with any multi-objective optimization algorithm that can deal with constraints but in the reference optimizer it is done with the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [49]. NSGA-II optimizes the design challenge by modifying the design variables. NSGA-II is allowed to vary the design variables between the user-defined lower and the upper limit. The design variations that come out of NSGA-II are evaluated on the random forest regression models to predict the constraint and objective scores. The objective and constraint scores are then combined with the design variable values and tested to see if the combination can be easily isolated by the Isolation Forest. Once the isolation score, the objective score, and the constraint scores are evaluated they are given back to the NSGA-II algorithm. The NSGA-II algorithm includes the evaluated design variations in the population of previously evaluated solutions and then new solutions are generated with the non-dominated genetic sorting strategy. After NSGA-II has converged, the optimal designs are reported so that they can be inspected with a parallel coordinate plot and the objectives can be visualized on a Pareto frontier.

4.4 Empirical Design Experiments

To validate the models and the algorithms, different experiments are conducted. The first experiment is set up to test the predictive capabilities of the random forest regression models. In the second experiment, a set of ships is intentionally modified to see if the isolation forest is capable of identifying the newly created anomalies. Finally, in the last experiment, everything is connected and novel container ship variations are generated on a Pareto frontier.

For all the experiments 2538 container ships are used. 1219 of these vessels have the duplicate characteristics and are filtered out during the preprocessing phase as described in Section 4.2.2.

4.4.1 Random Forest Regression Experiment

The random forest regression models are intended to predict the performance and capital investment cost of the ships of the future. In this experiment, such a situation is mimicked. Three different KPIs are learned by the random forest regression models with data from 1019 ships built before 2005, and then the random forest regression models are tested with data from 96 ships built after 2010. By comparing the predicted values with the actual values it can be determined if the trained random forest regression model is good to use in practice.

The KPIs that are predicted in this experiment are LSW, MCR, and DWT. The KPIs are estimated with the random forest regressor and with empirical design equations for the specific KPIs. The design variables used to predict LSW are [*Lbp, Boa, T, Cb, MCR*]. The design variables used for MCR are [*Lbp, Boa, T, Cb, V*]. The design variables used to predict DWT are [*Lbp, Boa, T, Cb*].

Random Forest Regression Results

The accuracy of the random forest regression model is determined with the R^2 measure [104]. This measure compares the real KPI values with the predicted values and see how much variation of the dependent variable can be explained by the model. With R^2 scores of 0.93, 0.90, and 0.95 for LSW, MCR, and DWT, it can be confirmed that the random forest regressor is capable of capturing a lot of variance.

Empirical Design Equation Results

Light Ship Weight for container vessels can be predicted with the Empirical Design Equation of D'almeida [55]. The prediction of D'almeida for LSW is dependent on the *steel weight (SW)*, *outfitting & equipment weight (OEW)*, and *machinery weight (MW)*:

$$\begin{aligned} LSW &= SW + OEW + MW \\ SW &= 0.0293 \cdot Lbp^{1.76} \cdot Boa^{0.712} \cdot T^{0.374} \\ MW &= 2.35 \cdot (MCR/0.745699872)^{0.60} \end{aligned} \tag{4.1}$$

The D'almeida equations use the same independent variables as in the random forest regressor model to calculate the LSW. However, the estimate of this empirical formulation only obtains an R^2 score of 0.84.

Maximum Continuous Rating can be estimated with the empirical formula named the *Admirality constant* [129]. The admirality constant C can be calculated by using the maximum continuous rating (MCR) and displacement values (Δ) from reference vessels and then plugging it in the following formula:

$$MCR = \frac{\Delta^{2/3} \cdot V^3}{C} \tag{4.2}$$

The mean admirality constant (C) of the reference vessels is then stored so that it can be used in later approximations for MCR given different discplacements. In the experiment, the *Admirality constant* itself (C) is approximated with the reference vessels from before 2005. The mean C from the vessels before 2005 is used to make predictions for the container vessels after 2010. The R^2 score for this formula is 0.87, again a worse R^2 score compared to the random forest regressor.

Dead Weigth Tonnage is dependent on the LSW of the vessel. The empirical formula for DWT is:

$$DWT = \Delta - LSW \tag{4.3}$$

but since the empirical equation for light ship weight has a worse R^2 score compared to the random forest regressor, it is no surprise that also for DWT, the R^2 score of 0.89 is lower compared to the R^2 score of the random forest regressor.

4.4.2 Isolation Forest Experiment

In the isolation forest experiments, the isolation forest is trained with the data from the container vessels as described earlier. After this, two data fields per vessel are modified to create impossible design parameter/KPI combinations. All vessels are then evaluated by the trained isolation forest to see if they are marked as an anomaly or not. The modified design parameters/KPI values and the percentage of anomalies detected are presented in Table 4.1.

Modified Columns	Anomaly Percentage
no modification	15%
$Lbp/1.1$, $T \times 1.1$	36%
$Lbp/1.25$, $T \times 1.25$	56%
$Lbp/1.5$, $T \times 1.5$	88%
$Lbp/1.75$, $T \times 1.75$	99%
$Lbp/2$, $T \times 2$	100%
$MCR/2$, $V \times 2$	95%
$LSW/2$, $Cb \times 2$	97%
$Lbp/2$, $DWT \times 2$	97%
$Cb/2$, $Lbp \times 2$	100%
$T/2$, $Cb \times 2$	100%
$Cb/2$, $V \times 2$	100%

Table 4.1: Modified columns and classified anomaly percentage after this modification.

The experiments indicate that as the vessels undergo more significant modifications, the isolation forest identifies an increasing percentage of vessels as anomalies. The experiments also show that there is a small percentage of vessels that have been radically changed but have not been marked as an anomaly. This indicates that the anomaly detection algorithm does not detect all anomalies and that the naval architect should pay attention when analyzing the results and do a few integrity checks on the results.

4.4.3 NSGA-II Experiment

For this experiment, it is assumed that the random forest regression model and the isolation forest perform as intended so that the NSGA-II algorithm can be tested. If the NSGA-II algorithm can find feasible and realistic Pareto-optimal solutions it can be confirmed that the reference optimizer works as intended.

The reference optimizer is tested again on a container ship case. In this experiment NSGA-II was allowed to vary the main particulars of the vessel (LBP, Boa, T, Cb,

V). The LSW and MCR are minimized, while the DWT capacity should be larger than or equal to 28000 tonnes. The results are visualized on the Pareto frontier in Figure 4.5 and Figure 4.6.

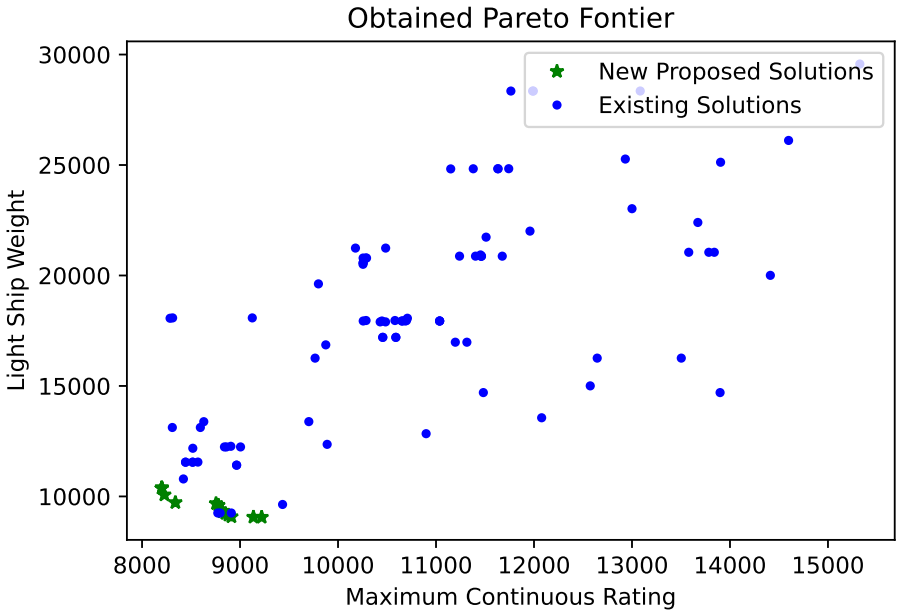


Figure 4.5: Obtained Pareto efficient solutions for test case. Blue solutions indicate existing vessels that do not violate any of the constraints while green solutions are the proposed solutions by NSGA-II.

NSGA-II in this experiment found 14 Pareto efficient solutions along the Pareto front interposed by Pareto efficient existing vessels. As previously described the algorithm only uses data and does not know any physics, it is the task of the naval architect to double-check the feasibility of the proposed solutions. In this experiment, the physical integrity of the proposed solutions are checked with the DWT Equation 4.3.

The weight balance of the vessel i.e. the sum of the DWT and LSW need to be in line with the corresponding displacement that can be calculated with: $Lbp{\cdot}Boa{\cdot}T{\cdot}Cb{\cdot}\rho$. Here $\rho = 1.025$ which is the water density of salt water. The found maximum deviation for existing vessels is 7% with an average of 0.2%. This indicates that data from the existing vessels is not always 100% accurate. The found maximum deviation for the proposed vessels by the reference finder is 2.6% with an average of 1.8%.

To give more details of the obtained Pareto efficient solutions, a parallel coordinate

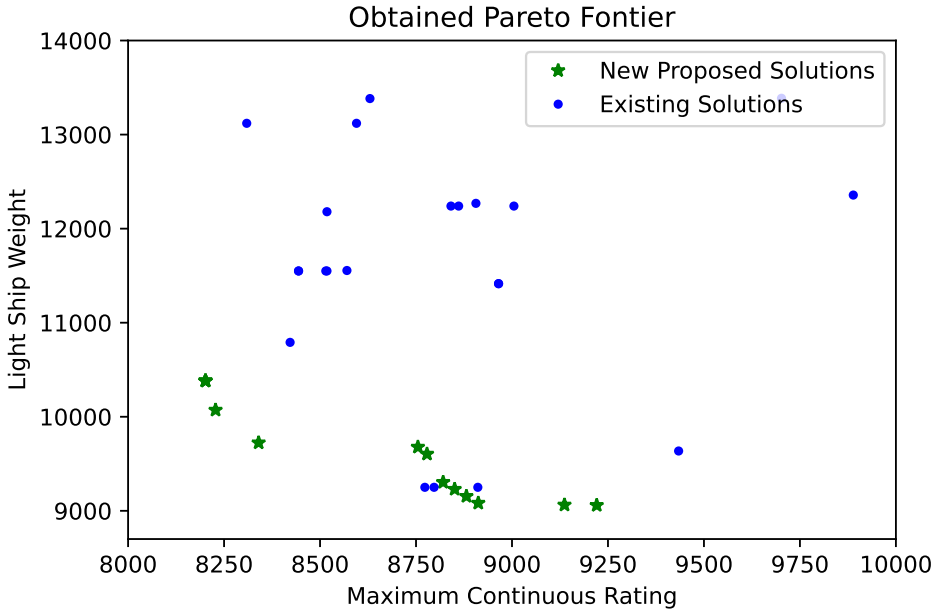


Figure 4.6: Zoomed in on Pareto frontier for test case. Blue solutions indicate existing vessels that do not violate any of the constraints while green solutions are the proposed solutions by NSGA-II. Inspection of this figure reveals that the majority of the existing solutions are dominated by the newly proposed solutions.

plot is made and presented in Figure 4.7. In the parallel coordinate plot, the main dimensions and the resulting performance indicators can be inspected and compared with the existing vessels.

4.5 Discussion

The reference optimizer as introduced and described in this chapter has two drawbacks that hold for any optimization process. The first drawback of the reference optimizer is that it needs a sufficient amount of good data for the random forest regressors to make accurate predictions. Good data without mistakes is important since otherwise, the random forest regressors will learn a wrong trend and the predictions will be off. Accurate and 100% reliable data is difficult to gather (especially for the less common vessel types) and sometimes only possible to obtain with expensive subscriptions.

A second drawback of the reference optimizer is that the design challenge should

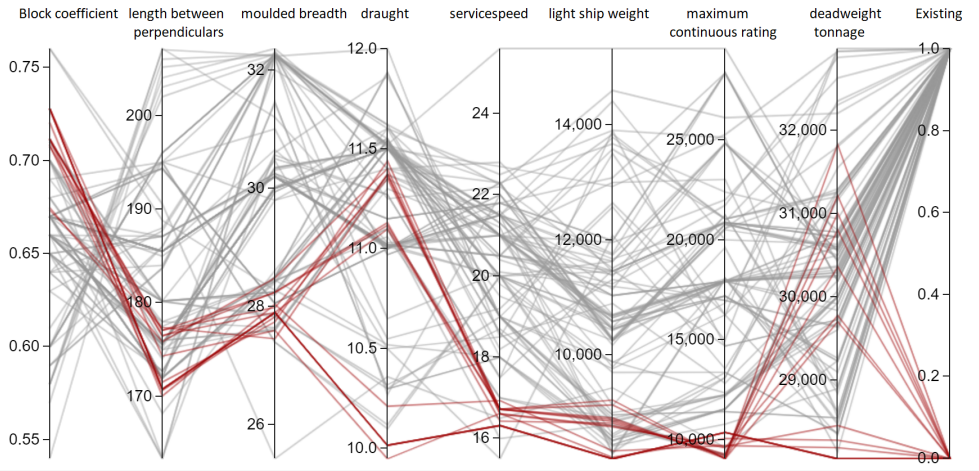


Figure 4.7: Parallel coordinate plot of the proposed solutions in red versus the existing vessels in grey.

be set up properly. For this, a naval architect needs to learn the basics of machine learning algorithms. During the training of naval architects at least the choice of what to choose as independent and dependent variables should be addressed in combination with different performance metrics.

4.6 Conclusion and Future Work

In this chapter, an alternative generic way is presented on how naval architects can use data to make preliminary design decisions by visualizing the data, learning from the data with machine learning algorithms, and finally finding optimal configurations with optimization algorithms.

The experiments in this chapter show that random forest regressors can give better estimations for light ship weight, dead weight, and maximum continuous rating compared to empirical design equations often used by naval architects. Besides a better estimation of key performance indicators, the random forest regressors are also capable of predicting key performance indicators for which no empirical design equations are readily available in the literature.

After training the random forest regressor and an anomaly detection algorithm, the models are coupled to a multi-objective optimization algorithm. This setup is capable of automatically generating optimal design configurations for preliminary ship

design problems. As a practical use case, a container vessel design challenge has been executed. The setup proposed 14 new Pareto-efficient solutions. The preliminary designs consisted of the main particulars of the vessels plus the key performance indicators light ship weight, maximum continuous rating, and deadweight. After this, the preliminary designs have been validated with integrity checks to verify their feasibility.

For future work, it is intended to improve the performance of the machine learning models even further, train them with more accurate data, and integrate a more robust anomaly detection algorithm to detect obvious mistakes better.

Chapter 5

Multi Objective Simulation Based Optimization

In simulated design optimization, a naval architect designs and optimizes a design by using a 3D model (ideally according to the guidelines from Section 3.2) and couples this 3D model to a simulator that evaluates Key Performance Indicators (KPIs). The simulators that evaluate the KPIs can be computationally very demanding. To speed up the process, occasionally calculations can be run in parallel, while other simulations require expensive licenses for each simulation in combination with specific hardware and software. Simulations are in some cases used to evaluate constraints, but more frequently to compute the objective scores. Where simulations can be costly to evaluate, some objectives and constraints are also computationally inexpensive. Such simple calculations can be called many times and in parallel. In this chapter research question 3 is answered *How to find the Pareto frontier of computationally expensive problems?* This chapter describes new optimization algorithms that can be used to optimize design problems with continuous decision variables, multiple constraints, and multiple objectives.

This chapter deals with the challenging design characteristics as efficiently as possible by splitting them up into separate research topics that answer the subquestions:

1. The first topic that is dealt with answers the following two subquestions: *How to deal with expensive multi-objective problems? How to efficiently satisfy constraints in multi-objective optimization?* Answers to these questions will make clear how to find the Pareto frontier of constraint multi-objective optimization

problems in as few function evaluations as possible.

2. The second topic deals with the subquestion: *How to propose multiple solutions for evaluation in parallel?* Answering this question helps us to reduce the total wall clock time for the evaluation.

3. As mentioned in the introduction of this chapter, not all evaluation methods are computationally expensive, therefore the last research question to addressed is *How to deal with a mix of expensive and inexpensive functions?*

Finally, the research question *How do the proposed algorithms compare to state-of-the-art algorithms?* is addressed throughout the entire chapter for every topic separately so that it becomes clear how the proposed methodologies perform compared to other algorithms.

5.1 Constraint Multi-Objective Optimization

Handling constraints in optimization problems can be done in several ways: using penalty functions, by separating the constraints and objectives, treating constraints as additional objectives, or hybrid methods [13, 59]. In this work, only separation of constraints and objectives is considered because the main issue with penalty functions is that the ideal penalty factors cannot be known in advance, and tuning the parameters requires a lot of additional function evaluations. The issue with treating constraints as additional objectives is that it makes the objective space unnecessarily more complex with a too strong bias towards the constraints.

In this Section, the SAMO-COBRA algorithm is introduced that uses separation of constraints and objectives in combination with surrogates. SAMO-COBRA, which is an abbreviation for Self-Adaptive Multi-Objective Constraint Optimization by using Radial Basis Function Approximations, owes its name to the very efficient constraint handling algorithms: COBRA [123] and SACOBRA [13]. Besides constraint handling, SAMO-COBRA has shown to be efficient in finding Pareto-optimal solutions, thereby solving constraint multi-objective problems by using a limited number of function evaluations. SAMO-COBRA is compared to two new state-of-the-art algorithms to empirically show the efficiency.

5.1.1 Related Work

Existing work on surrogate-assisted optimization is typically limited to a subset of three relevant requirements: multi-objective, constraint, and speed. For example, methods exist for quickly solving constraint single-objective problems (e.g. SACOBRA [13]), for multi-objective optimization without efficient constraint handling techniques (e.g. SMS-EGO [118] and PAREGO [88]), or for constraint multi-objective optimization without using meta-models, leading to a large number of required function evaluations (e.g. NSGA-II [49], NSGA-III [83], SPEA2 [174], and SMS-EMOA [18]). The recently proposed CEGO [154] and ECMO [133] algorithms address all three requirements, however, their computational cost grows very fast as they use Kriging surrogates that have a higher computational complexity compared to Radial Basis Functions [12, 60, 160].

Only very occasionally a surrogate-based algorithm is published that deals with both constraints and multiple objectives in an effective manner without using a Kriging surrogate (e.g., Datta's and Regis' SMES-RBF [44] and Blank and Deb's SA-NSGA-II [19].)

The algorithms used in the experiments of this section are described in more detail.

NSGA-II

The Non-dominated Sorting Genetic Algorithm, version II [49] is a classic multi-objective optimization algorithm. NSGA-II starts with a random design of experiments that is evaluated on all objectives. After the initial sample, all the solutions are ranked with a non-dominated sorting algorithm that defines multiple Pareto frontiers on different dominance levels. The crowding distance (density of the solutions in the objective space) is then computed for all solutions per dominance level. The crowding distance and the Pareto frontier rank are used to determine which solutions are selected to create an offspring population. The solutions with a higher crowding distance score and better dominance score have a higher chance of getting selected. The offspring population is then created with a crossover and mutation operator to introduce new combinations of decision parameters. For the offspring population and the parent population, the crowding distance and non-dominated sorting algorithm again define which p solutions from the parent and offspring population combined survive to the next iteration. The algorithm terminates until the evaluation budget is exhausted.

NSGA-III

The adaptive NSGA-III algorithm [83] is a many-objective optimization algorithm based on NSGA-II [49] and the original NSGA-III algorithm [48]. the adaptive NSGA-III algorithm starts with a random initial sample. Then in every iteration it emphasizes certain individuals in the population who are both non-dominant and close to a set of reference points are well distributed and are generated in desirable locations for solutions. The algorithm can both be used for constraint and unconstraint problems since in every iteration the non-useful reference points are re-allocated around the useful feasible reference points [83]. For each solution in the population, the degree of constraint violation is measured which influences together with the closeness to the reference points if it is selected for recombination.

CEGO

The CEGO optimization algorithm [154] by default uses a Latin Hypercube Sample of size $3 \cdot d$. After the initial sample Kriging models (also sometimes referred to as Gaussian Process Regression models) are trained for the objectives, while RBFs are used for the constraints. The CEGO algorithm then combines the $\mathcal{S}$-Metric-Selection-based Efficient Global Optimization (SMS-EGO [118]) algorithm with the constraint handling techniques from the Self-Adjusting Constraint Optimization by Radial Basis Function Approximation (SACOBRA [13]) to propose feasible Pareto efficient solutions. After a user-defined number of function evaluations, the algorithm terminates the evaluated solutions and the corresponding objective and constraint solutions are returned.

SMES-RBF

SMES-RBF [44] is a surrogate-assisted evolutionary strategy that uses cubic Radial Basis Functions as a surrogate for the objectives and constraints to estimate the actual function values. In every iteration, a large number of offspring solutions are generated by using a mutation operator on the parent population. The number of offspring solutions that are generated from the parent population is chosen rather large however, not all offspring solutions are evaluated on the real objective and constraint functions. Instead, the RBFs that are updated every iteration are used to determine the offspring solutions feasibility and objective scores. Only the most promising solutions according to a non-dominated sorting procedure are evaluated on the real objective

and constraint function. This process continues until the evaluation budget limit has been reached.

SA-NSGA-II

A variant of NSGA-II [49], called Surrogate-Assisted NSGA-II (SA-NSGA-II)[1], integrates surrogate assistance into the optimization cycle for the optimization of unconstraint and constraint multi-objective optimization problems. The surrogates employed in the SA-NSGA-II algorithm are RBFs with a cubic kernel and a linear tail. The idea is based on executing the optimization algorithm for multiple generations only on surrogate models (one for each objective and constraint) before calling the expensive optimization function. This embedded surrogate-based optimization loop provides a set of candidate solutions, from which a subset is selected. Assuming p solutions shall be evaluated using the expensive simulation in each optimization cycle, the candidates are first separated into p clusters in m-dimensional objective space before determining the selected solution for each cluster by performing a roulette wheel selection based on their crowding distances. After evaluating these p solutions on the expensive function, all surrogate models are updated and the new optimization cycle is started if the solution evaluation budget is not exhausted yet. SA-NSGA-II can also optimize constraint optimization problems by using the parameter-less domination approach [47] used in NSGA-II's selection operators.

5.1.2 SAMO-COBRA

The new SAMO-COBRA algorithm is designed to deal with continuous decision variables, multiple objectives, multiple complex constraints, and expensive objective function evaluations in an efficient manner. The idea behind the algorithm is that in every iteration, for each objective and for each constraint independently, the best transformation and the best RBF kernel are sought. In each iteration, the best fit is used to search for a new unseen feasible Pareto efficient point that contributes the most to the hypervolume between a user-defined reference point and the Pareto frontier. The pseudocode of SAMO-COBRA can be found in Algorithm 1. The Python implementation can be found on the Github page [143]. More details about the algorithm are given in the subsections below.

[1] Availabe on pysamoo as SSA-NSGA-II [20].

Algorithm 1: SAMO-COBRA. **Input:** Objective functions $f(\mathbf{x})$, constraint function(s) $g(\mathbf{x})$, decision parameters' lower and upper bounds $[\mathbf{x_{lb}}, \mathbf{x_{ub}}] \subset \mathbb{R}^d$, reference point $\mathbf{ref} \in \mathbb{R}^k$, number of initial samples N, maximum evaluation budget N_{max}, $RBF_{kernels}$ $(\varphi) = \{cubic, gaussian, multiquadric, invquadric, invmultiquadric, thinplatespline\}$ **Output:** Evaluated feasible Pareto efficient solutions.

Function SAMO-COBRA(f, g, $[\mathbf{x_{lb}}, \mathbf{x_{ub}}]$, **ref**, N, N_{max}, $RBF_{kernels}$):
  $\mathbf{X} \leftarrow \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ ▷ Generate initial design, $\mathbf{X} \in \mathbb{R}^{d \times N}$
  $\mathbf{F} \leftarrow f(\mathbf{X})$ ▷ Obtain objective scores, $\mathbf{F} \in \mathbb{R}^{k \times N}$
  $\mathbf{G} \leftarrow g(\mathbf{X})$ ▷ Obtain constraint scores, $\mathbf{G} \in \mathbb{R}^{m \times N}$
  $RBF^* \leftarrow \{(Cubic, standardized) | \forall \mathbf{f} \in \{\mathbf{F} \cup \mathbf{G}\}\}$ ▷ initialize best RBF
  while $N < N_{max}$ **do**
    $\hat{\mathbf{X}} \leftarrow$ SCALE$(\mathbf{X}, [-1, 1]^d)$ ▷ Scale input space to $[-1, 1]^d$
    $\tilde{\mathbf{F}} \leftarrow$ PLOG$(\mathbf{F})$ ▷ See function plog in Eq. (2.6)
    $\tilde{\mathbf{G}} \leftarrow$ PLOG$(\mathbf{G})$ ▷ See function plog in Eq. (2.6)
    $\hat{\mathbf{F}} \leftarrow$ STANDARDIZE$(\mathbf{F})$ ▷ Standardize objective space
    $\hat{\mathbf{G}} \leftarrow$ SCALE CONSTRAINT$(\mathbf{G})$ ▷ 0 remains feasibility boundary
    for $\varphi \in RBF_{kernels}$ **do** ▷ For each kernel
      for $i \leftarrow 1$ **to** k **do** ▷ For each objective
        $\hat{S}_i^{\varphi} \leftarrow$ FITRBF$(\hat{\mathbf{X}}, \hat{\mathbf{F}}_{(i,\cdot)}, \varphi)$ ▷ Fit with std(F) values
        $\tilde{S}_i^{\varphi} \leftarrow$ FITRBF$(\hat{\mathbf{X}}, \tilde{\mathbf{F}}_{(i,\cdot)}, \varphi)$ ▷ Fit with PLOG(F) values
      end
      for $j \leftarrow 1$ **to** m **do** ▷ For each constraint
        $\hat{S}_{k+j}^{\varphi} \leftarrow$ FITRBF$(\hat{\mathbf{X}}, \hat{\mathbf{G}}_{(j,\cdot)}, \varphi)$ ▷ Fit with scaled(G) values
        $\tilde{S}_{k+j}^{\varphi} \leftarrow$ FITRBF$(\hat{\mathbf{X}}, \tilde{\mathbf{G}}_{(j,\cdot)}, \varphi)$ ▷ Fit with PLOG(G) values
      end
    end
    $S^* \leftarrow \left\{ S_i^{(RBF_i^*)} \mid \forall i = 1, \ldots, (k+m) \right\}$ ▷ Apply best RBF conf.
    $\mathbf{PF} \leftarrow$PARETO$(\mathbf{X}, \mathbf{F}, \mathbf{G})$ ▷ PF indicator $\mathbf{PF} \in \{0, 1\}^N$
    $\mathbf{x}^* \leftarrow$ MAX(HV, **PF**, **ref**, S^*) ▷ Get solution with largest HV
    $\mathbf{x}_{new} \leftarrow$ SCALE$(\mathbf{x}^*, [\mathbf{x_{lb}}, \mathbf{x_{ub}}])$ ▷ Scale to original scale
    $N \leftarrow N + 1$ ▷ Increase iteration counter to new matrix sizes
    $\mathbf{X} \leftarrow [\mathbf{X}\ \mathbf{x}_{new}]$ ▷ Add new solution, $\mathbf{X} \in \mathbb{R}^{d \times N}$
    $\mathbf{F} \leftarrow [\mathbf{F}\ f(\mathbf{x}_{new})]$ ▷ Add evaluated objectives, $\mathbf{F} \in \mathbb{R}^{k \times N}$
    $\mathbf{G} \leftarrow [\mathbf{G}\ g(\mathbf{x}_{new})]$ ▷ Add evaluated constraints, $\mathbf{G} \in \mathbb{R}^{m \times N}$
    $RBF^*, \mathbf{SE} \leftarrow$SELECTBESTRBF$(\mathbf{SE}, S, \mathbf{x}^*, \mathbf{F}, \mathbf{G}, \mathbf{PF}, N)$
  end
return $(\mathbf{F}_{(\cdot,\mathbf{PF})}, \mathbf{G}_{(\cdot,\mathbf{PF})}, \mathbf{X}_{(\cdot,\mathbf{PF})})$

Initial Design of Experiments

Bossek et al. showed empirically that, when dealing with sequential model-based optimization, in most cases it is best to use the Halton sampling strategy [73] with an initial sample that is as small as possible [23]. The smallest initial sample for RBF surrogate-assisted optimization algorithms is $d + 1$ since that many evaluations are required to train the first RBF. A few experiments where 2 alternative initial sampling strategies are compared to the $d+1$ initial Halton sample strategy proposed by Bossek et al. confirmed that a small initial sample size and Halton sampling also lead to the best results when applied to the BNH, CEXP, SRN, TNK, CTP1, and TRICOP constraint multi-objective problems from Section 2.4. In the small experiments, the SAMO-COBRA algorithm was run 10 times and the hypervolume performance metric was checked after $40 \cdot d$ function evaluations

Table 5.1: Hypervolume after $40 \cdot d$ function evaluations for SAMO-COBRA with different initial sampling sizes and strategies. **Bold** indicates significantly better or indifferent results according to a Wilcoxon rank-sum test with $p \leq 0.05$.

Function	Halton $\mathbf{d+1}$	Halton $\mathbf{3 \cdot d}$	LHS $\mathbf{d+1}$
BNH	**5256.4**	5255.7	**5256.3**
CEXP	3.7973	**3.7976**	**3.7979**
SRN	**62391**	**62375**	**62387**
TNK	**8.0505**	**8.0487**	8.0442
CTP1	**1.3030**	**1.3030**	1.3029
TRICOP1	**20611**	**20611**	20610

As can be seen from the results in Table 5.1, the Halton sampling strategy with $d + 1$ initial samples in most cases leads to better or similar results compared to the other two initial sampling strategies. Therefore, it is advised to create an initial Halton sample of size $d + 1$ before the sequential optimization procedure starts, when using SAMO-COBRA.

Every sample in the initial design is then evaluated (lines 2-4 of Algorithm 1) so that all samples have their corresponding constraint and objective scores.

Radial basis Function Surrogates

The SAMO-COBRA algorithm employs Radial Basis Functions with a polynomial tail as a surrogate. Details on how his surrogate can be fitted and how it can be used to predict values for unseen data points are described in Section 2.3.2. Because upfront it can not be known if a Plog transformation is beneficial, and which kernel is ideal,

all different kernels and transformations are applied. This results in $6 \times 2 = 12$ RBF options to choose from: $\Phi = \{Cubic, Gaussian, Multiquadric, InverseQuadratic, InverseMultiquadric, ThinPlateSpline\} \times \{\text{PLOG}, standardized\}$. Initially, the RBF configuration with a *Cubic* kernel with standardized objective scores is selected (line 5 from Algorithm 1. This surrogate configuration is then used in the search for a feasible Pareto-Efficient solution by maximizing the hypervolume contribution.

Maximize Hypervolume Contribution

After modeling the relationship between the input space and the response variables with the RBFs, the RBFs are used as cheap surrogates. By using Eq. (2.4) for each unseen input $\mathbf{x}'$, every corresponding constraint and objective prediction can be calculated. Given the RBF approximations for a solution $\mathbf{x}'$, the constraint predictions can be used to check if the solution is predicted to satisfy all the constraints. Besides the constraint predictions, the objective predictions can be used to see if the solution is a preferred solution or not. Whether one solution is preferred above another solution can be computed with an infill criteria, also known as acquisition function. There are two infill criteria considered in this work, the S-Metric Selection criterion ($\mathcal{S}$-metric), and the Predicted HyperVolume criterion ($\mathcal{P}_{hv}$). Computation of the two infill criteria is done as follows:

1. Compute all objective values for a given solution $\mathbf{x}'$ with Eq. (2.4). With the interpolated objective values, compute the additional predicted hypervolume ($\mathcal{P}_{hv}$) score this solution adds to the Pareto frontier. This is a purely exploitative infill criterion without any uncertainty quantification method.

2. Compute all objective scores for a given solution $\mathbf{x}'$ with Eq. (2.4) and subtract the uncertainty of each objective given $\mathbf{x}'$ and Eq. (2.5). With the interpolated objective score minus the uncertainty, the potential HV that this solution could add to the Pareto frontier is calculated. This infill criterion is similar to the Kriging $\mathcal{S}$-metric Selection ($\mathcal{S}$-metric) criterion from Emmerich et al. [18]. Because of the subtracted uncertainty, it will be more exploratory compared to the $\mathcal{P}_{hv}$ criterion.

How much a solution adds to the Pareto frontier is based on how much HV the solution adds between the already evaluated non-dominated solutions and a predefined reference point. A visual representation of the HV scores of two different solutions is displayed in Figure 5.1. By using any of the two infill criteria, the constraint multi-objective problem has been translated into a constraint single-objective problem.

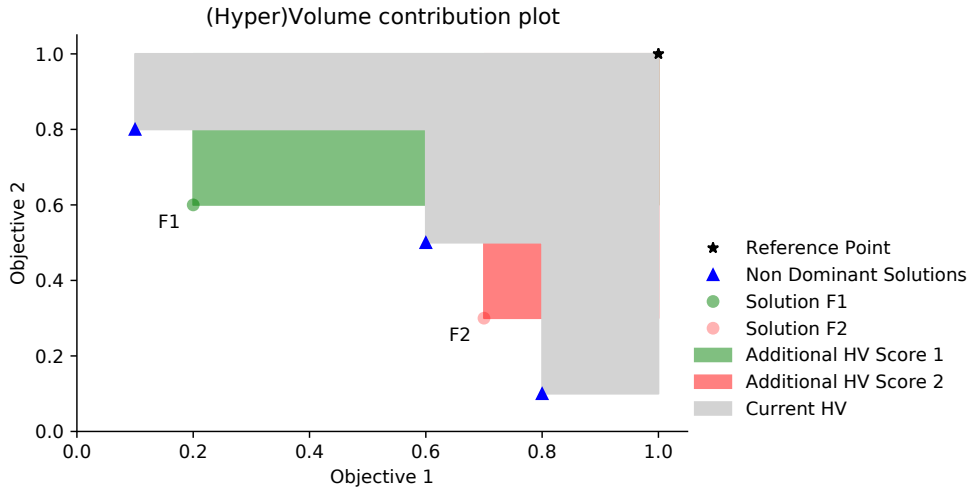


Figure 5.1: Visual representation of hypervolume contribution of two solutions. The hypervolume contribution of solution F1 is equal to $0.2 \cdot 0.4 = 0.08$, the hypervolume contribution of solution F2 is equal to $0.1 \cdot 0.1 = 0.01$. This makes solution F1 more desirable compared to solution F2.

The single point acquisition function optimization problem can be mathematically defined as follows:

$$\begin{aligned} \mathbf{x}^* \in \underset{\mathbf{x} \in \Omega \subset \mathbb{R}^d}{\operatorname{argmax}} \; & \mathcal{P}_{hv}(\mathbf{f}'(\mathbf{x})) \\ \text{subject to } & \mathbf{g}'(\mathbf{x}) \leq \mathbf{0} \end{aligned} \tag{5.1}$$

After an infill criterion is chosen by the user, the constraint single-objective problem can be optimized. The COBYLA (Constraint Optimization BY Linear Approximations) algorithm [120] is used to maximize the infill criterion (line 24 of Algorithm 1). COBYLA is allowed to vary $\mathbf{x}'$ between the lower and the upper bound of the design space $\mathbf{x}' \in [\mathbf{x_{lb}}, \mathbf{x_{ub}}]$. This way, COBYLA searches for a Pareto-optimal solution that does not violate any of the constraints and has the highest possible infill criterion score.

If no feasible solution can be found, the solution with the smallest constraint violation is selected for evaluation. Note that COBYLA does not use the real objective and constraint function evaluations during the search for the next best solution. Instead, COBYLA uses the cheap RBF surrogates as surrogates for the real objective and constraint functions. The chances of finding the best feasible Pareto-optimal solu-

tion can be increased by starting the surrogate search not from one solution but from multiple randomly generated solutions independently. Therefore COBYLA starts 16 times from a randomly generated solution. Each independent local search done by COBYLA gets an allocated search budget.

Only after the next best solution on the surrogates is found, it is evaluated on the real objective and constraint functions (lines 25-29 of Algorithm 1).

Surrogate Exploration and RBF adaptation

Because in the first iterations the RBFs do not model the constraints very well yet, an allowed error (ϵ) of 1% for each constraint is built in. If the solution evaluated on the real constraint function is feasible, the error margin of this constraint approximation is reduced by 10%. If a solution is infeasible, the RBFs surrogate approximation is clearly still wrong. Therefore, the error margin of the corresponding constraint is increased by 10%.

Besides the error margin, in every iteration, also the best RBF kernel and transformation strategy is chosen (line 30 of Algorithm 1). The pseudocode of this function can be found in Algorithm 2. Finding the best RBF kernel and transformation strategy is done by computing the difference between the RBF interpolated solution and the solution computed with the real constraint and objective functions. This difference is computed every iteration, resulting in a list of historical RBF approximation errors for each constraint and objective function, for each kernel, with and without the PLOG transformation.

Based on the RBF approximation errors, the best RBF kernel and transformation are chosen. Bagheri et al. show empirically, that if only the last approximation error is considered in the single objective case, the algorithm converges to the best solution faster [11]. This is the case because when closer to the optimum, the vicinity of the last solution is the most important. In the multi-objective case, the vicinities of all the feasible Pareto-optimal solutions are important. Experiments confirmed that the approximation errors of the feasible Pareto-optimal solutions and the last four solutions should be considered. The approximation errors of the last four solutions ensure that the algorithm does not get stuck on one RBF configuration and the error of the Pareto-efficient solutions ensures that all the vicinities of the optimal solutions are considered. The Mean Squared Error measure is used to quantify which RBF kernel and which transformation function in the previous iterations resulted in the smallest approximation error.

Algorithm 2: SelectBestRBF
Input: **SE** Historic squared RBF approximation error, per RBF kernel, with and without PLOG transformation, for each objective, and for each constraint. S surrogate models for each kernel, with and without PLOG transformation, for each objective, and for each constraint. $\mathbf{x}^*$ last evaluated solution. **F** objective scores, **G** constraint scores, **PF** Pareto frontier indicator vector. N number of function evaluations.
Output: best RBF kernel, and PLOG strategy for each objective and constraint separately, and historic squared approximation errors.

Function SelectBestRBF($\mathbf{SE}, S, \mathbf{x}^*, \mathbf{F}, \mathbf{G}, \mathbf{PF}, N$)**:**
  $\mathbf{ID} \leftarrow \mathbf{PF} \cup \{\mathbf{ID}_i \leftarrow 1 \mid \forall i = N-4, \ldots, N\}$ ▷ Mark last 4 and Pareto front in a vector to select relevant approximation errors
  $T \leftarrow \{T_i \leftarrow \infty \mid \forall i = 1, \ldots, (k+m)\}$ ▷ Temporary approx. errors
  for $\varphi \in RBF_{kernels}$ **do** ▷ For each kernel check approx. errors
    for $i \leftarrow 1$ **to** k **do** ▷ For each obj. with and without PLOG
      $\hat{\mathbf{SE}}^{\varphi}_{i,N} \leftarrow \left(\text{INTERPOLATE}(\hat{S}^{\varphi}_i, \mathbf{x}^*) - \mathbf{F}_{i,N}\right)^2$ ▷ Save RBF Error
      $\tilde{\mathbf{SE}}^{\varphi}_{i,N} \leftarrow \left(\text{INTERPOLATE}(\tilde{S}^{\varphi}_i, \mathbf{x}^*) - \mathbf{F}_{i,N}\right)^2$ ▷ Save RBF Error
    end
    for $j \leftarrow 1$ **to** m **do** ▷ For each constr. with and without PLOG
      $\hat{\mathbf{SE}}^{\varphi}_{k+j,N} \leftarrow \left(\text{INTERPOLATE}(\hat{S}^{\varphi}_{k+j}, \mathbf{x}^*) - \mathbf{G}_{j,N}\right)^2$ ▷ Save RBF Error
      $\tilde{\mathbf{SE}}^{\varphi}_{k+j,N} \leftarrow \left(\text{INTERPOLATE}(\tilde{S}^{\varphi}_{k+j}, \mathbf{x}^*) - \mathbf{G}_{j,N}\right)^2$ ▷ Save RBF Error
    end
    for $i \leftarrow 1$ **to** $k+m$ **do** ▷ For each surrogate find best strategy
      if $(\sum_{n=1}^{N} \mathbf{ID}_n \cdot \hat{\mathbf{SE}}^{\varphi}_{i,n}) < T_i$ **then** ▷ If error sum < temp
        $T_i \leftarrow \sum_{n=1}^{N} \mathbf{ID}_n \cdot \hat{\mathbf{SE}}^{\varphi}_{i,n}$ ▷ Save approx. errors in temp
        $RBF^*_i \leftarrow (kernel = \varphi, \text{PLOG}{=}False)$ ▷ Save best strategy
      if $(\sum_{n=1}^{N} \mathbf{ID}_n \cdot \tilde{\mathbf{SE}}^{\varphi}_{i,n}) < T_i$ **then** ▷ If error sum < temp
        $T_i \leftarrow \mathbf{ID}_n \cdot \tilde{\mathbf{SE}}^{\varphi}_{i,n}$ ▷ Save approx. errors in temp
        $RBF^*_i \leftarrow (kernel = \varphi, \text{PLOG}{=}True)$ ▷ Save best strategy
    end
  end
return $(RBF_*, \mathbf{SE})$

5.1.3 Multi-Objective Optimization Experiments

Two experiments are set up to compare SAMO-COBRA with other state of the art algorithms. In these experiments, two variants of the SAMO-COBRA algorithm are tested, one without the uncertainty quantification method ($\mathcal{P}_{hv}$), and one with the uncertainty quantification method ($\mathcal{S}$-metric). The performance of the two variants are compared to the performance of the following algorithms: CEGO [154], SA-NSGA-II [19], NSGA-II [49], NSGA-III [83], and SMES-RBF [44]. The performance of the algorithms except for SMES-RBF are assessed on 18 benchmark functions. SMES-RBF could not be tested since the implementation of SMES-RBF has not been made available and as such it could only be compared to the results reported in the SMES-

RBF publication.

All test functions from Table 2.1 are used except for the MW test problems. This is because these problems have a very low feasibility ratio and therefore are not ideal for testing the performance of surrogate-assisted optimization algorithms. Each algorithm is tested 10 times on every test function to get a trustworthy result. The results for NSGA-II and NSGA-III had a high variance. Therefore, 100 runs are executed for those algorithms. In the first experiment, the algorithms are given a fixed budget to find a feasible Pareto frontier. In the second experiment the algorithms are evaluated to see how many function evaluations they require to achieve a predefined threshold performance.

Hyperparameter Settings

In the experiments for each algorithm either the original implementation is used or an implementation which was readily available in Python. For all algorithms, the recommended hyperparameters from the original implementations are used. Since there are no clear recommendations for the hyperparameters of NSGA-II and NSGA-III, a grid search is conducted. In the grid search the optimal population size and number of generations are determined for NSGA-II. For NSGA-III a grid search is done to find the best parameter value for the number of divisions that influence the spacing of the reference points of NSGA-III. For the sake of brevity, only the results with the best scores from this grid search are reported.

The implementations of the different algorithms are listed here: the original implementation of CEGO can be found on the dedicated Github page[2]. The original implementation of IC-SA-NSGA-II and SA-NSGA-II can be found on the personal page of Julian Blank[3]. For NSGA-II and NSGA-III the implementation of Platypus is used[4]. The implementation of the SMES-RBF algorithm is not provided. Therefore, only the reported results from the SMES-RBF paper [44] can be compared.

More details concerning the implementation of SAMO-COBRA, the experiments, and the statistical comparison can be found on a dedicated Github page [143].

Fixed Budget Experiment

In the first experiment, each algorithm was given a limited fixed number of function evaluation after which the HV performance metric is computed. Each algorithm is

[2] `https://github.com/RoydeZomer/CEGO`
[3] `https://julianblank.com/static/misc/pycheapconstr.zip`
[4] `https://platypus.readthedocs.io/`

allowed to do $40 \cdot d$ function evaluations, here d represents the number of decision variables of the optimization test function. As a performance metric, the HV metric is selected to quantify the results. The HV is computed between the obtained feasible Pareto-optimal solutions and the reference point reported in Table 2.1. Higher HV scores mean that more HV is covered and therefore a better approximation of the Pareto frontier is found.

Convergence Experiment

In the second experiment, each algorithm is tested to see when it reaches a threshold value of the HV metric. The threshold is set to 95% of the maximum achievable HV per test function between the reference points in Table 2.1 and the Pareto frontier. Since the Pareto frontier is not known for every function, NSGA-II is used to find the maximal HV between a reference point and the Pareto frontier by running it with a population size of $100 \cdot d$ and allowing the algorithm to run for 1000 generations.

For each algorithm, after each iteration or generation, the HV is computed. As soon as the threshold value is achieved, the number of function evaluations are used as the performance metric. A small number of required function evaluations is desirable so the algorithm with the smallest number of evaluations is classified as the winner in this experiment.

To be able to compare the results of SMES-RBF with the results of SAMO-COBRA, a different experiment is conducted. In this experiment the number of function evaluations are compared between SAMO-COBRA and SMES-RBF to achieve the HV as reported in the SMES-RBF paper [44].

5.1.4 Results

The complete set of results from the experiments can be found on Github [143]. The results of the fixed budget and the convergence experiment are reported in table format in the following Sections.

Fixed Budget Experiment Results

The results of the first experiment, in which the HV is computed after $40 \cdot d$ function evaluations, is reported in Table 5.2. A Wilcoxon rank-sum test with Bonferroni correction is used to determine if there is a significant difference between the algorithm with the best results and the algorithm with the lesser results.

Table 5.2: Mean hypervolume after $40 \cdot d$ function evaluations for each algorithm on each test function. $\mathcal{P}_{hv}$ and $\mathcal{S}$-metric represent the SAMO-COBRA variants. The highest mean hypervolumes per test function are presented in **bold**. The Wilcoxon rank-sum test (with Bonferroni correction) significance is represented in cyan. Background colours represent the significant best and incomparable results: $p \leq 0.001$, while one shade lighter represents incomparability with $p \leq 0.01$, finally Red shows that the algorithm required more than 24 hours.

Function	PHV	SMS	CEGO	NSGA-II	NSGA-III	SA-NSGA-II
BNH	**5072.10**	5067.05	5037.20	4910.44	4673.78	4862.96
CEXP	**3.7968**	3.7968	3.7658	3.1545	2.9585	3.5790
SRN	**25016**	25004	24974	20767	19749	23261
TNK	0.2887	**0.2930**	0.2837	0.1181	0.1209	0.2485
CTP1	**0.3026**	0.3023	0.2972	0.2250	0.2193	0.2739
C3DTLZ4	1.3162	1.4698	1.3644	1.5069	1.5024	**1.6560**
OSY	**12628**	12515	12318	2260	2231	12313
TBTD	**486.7**	485.5	484.5	350.2	361.3	416.3
NBP	**798532**	798204	792130	737269	705200	763128
DBD	**34.635**	34.174	34.112	30.107	30.297	33.654
SRD	**3068272**	3028279	3011838	1839509	1761892	3064597
WB	0.3850	0.3799	**0.3984**	0.3247	0.3303	0.3718
BICOP1	**0.6641**	0.0	*terminated*	0.0003	0.0470	0.6489
BICOP2	**0.2283**	0.1752	*terminated*	0.1442	0.1466	0.1265
TRICOP	**49.654**	49.602	49.599	39.6356	38.1846	42.6394
SPD	$\mathbf{5.849 \cdot 10^9}$	$5.407 \cdot 10^9$	$4.960 \cdot 10^9$	$3.144 \cdot 10^9$	$3.106 \cdot 10^9$	$5.060 \cdot 10^9$
CSI	**8.3148**	7.2818	*terminated*	4.4687	4.3737	7.0922
WP	$\mathbf{3.4315 \cdot 10^{18}}$	$3.3544 \cdot 10^{18}$	$3.2455 \cdot 10^{18}$	$2.1620 \cdot 10^{18}$	$2.2026 \cdot 10^{18}$	$1.6930 \cdot 10^{18}$

SAMO-COBRA with the predicted hypervolume infill criterion ($\mathcal{P}_{hv}$) achieves in 15 out of the 18 test functions the highest mean hypervolume. The SAMO-COBRA algorithm with the S-Metric Selection ($\mathcal{S}$-metric) infill criterion achieves the highest mean hypervolume on the TNK test problem and in 7 other cases achieves a mean hypervolume that is statistically incomparable to the SAMO-COBRA algorithm with the $\mathcal{P}_{hv}$ infill criterion. The CEGO algorithm achieves the best mean hypervolume on the WB test function but this is incomparable with the $\mathcal{P}_{hv}$-SAMO-COBRA, $\mathcal{S}$-metric-SAMO-COBRA and SA-NSGA-II algorithms. On three other problems, the CEGO algorithm also achieves incomparable results. The CEGO algorithm however was terminated while optimizing 3 functions since the experiments took longer than 24 hours to find a Pareto frontier. This mainly happened on test problems with a high number of parameters. The SA-NSGA-II algorithm achieves the best hypervolume on the C3DTLZ4 test function. On the WB test function, SA-NSGA-II found an incomparable mean hypervolume.

Convergence Experiment Results

In Table 5.3, the number of function evaluations are reported that are required to achieve the 95% threshold value of the maximum HV. For some test functions, this

was quite easy to achieve since it only required to evaluate the initial sample. On other test functions the algorithms required many more evaluations to achieve the threshold.

NSGA-II and NSGA-III are terminated after 5000 function evaluations on the *C3DLTZ4*, *OSY*, *SPD*, and *SRD* test function. CEGO was not able to obtain the threshold value for the *SPD* and *CSI* function within 24 hours.

Table 5.3: The table shows the number of function evaluations needed to achieve the threshold hypervolume for each test function. The results of the algorithm with the smallest number of function evaluations are reported in bold accompanied with a ↑. $\mathcal{P}_{hv}$ and $\mathcal{S}$-metric represents the SAMO-COBRA variants. Experiments that required more than 5000 function evaluations are terminated and displayed as +5000. Experments that required more than 24 hours are terminated and represented with a (-).

Function	Threshold	PHV	SMS	CEGO	SA-NSGA-II	NSGA-II	NSGA-III
BNH	5005.5	**11** ↑	16	12	36	56	114
CEXP	3.6181	**13** ↑	16	23	71	392	404
SRN	59441	**15** ↑	**15** ↑	17	66	200	227
TNK	7.6568	11	**9**↑	**9** ↑	66	432	586
CTP1	1.2398	**10** ↑	14	14	36	140	170
C3DTLZ4	6.4430	**179** ↑	181	226	275	+5000	+5000
OSY	95592	**15** ↑	31	16	105	+5000	+5000
TBTD	3925	**31** ↑	58	49	357	324	369
NBP	1.024E8	**5** ↑	9	6	36	102	206
DBD	217.31	**13** ↑	19	16	48	112	142
SPD	3.6887E10	**43** ↑	125	-	205	+5000	+5000
CSI	25.717	**59** ↑	484	-	376	+5000	+5000
SRD	3997308	**17** ↑	55	28	81	952	1357
WB	32.9034	**7** ↑	10	10	43	24	24
BICOP1	76.6328	**22** ↑	25	35	119	1700	1975
BICOP2	4606.57	17	18	18	109	**10** ↑	12
TRIPCOP	19578.0	**7** ↑	8	**7** ↑	21	42	8
WP	1.5147E19	**48** ↑	66	111	292	3120	3876

As can be seen in Table 5.3, SAMO-COBRA with the $\mathcal{P}_{hv}$ infill criterion again outperforms the other algorithms for the majority of the test functions. This is interesting because this infill criterion is designed to be exploitative, despite that the infill criterion is exploitative the algorithm can still find 95% of the Pareto frontier.

SMES-RBF Convergence Experiment Results. As mentioned before, the implementation of SMES-RBF is not publicly available. Therefore, the reported results of SMES-RBF are compared with the results of SAMO-COBRA. In Table 5.4 the

number of function evaluations are reported that are required to obtain the same HV as reported in the SMES-RBF paper.

Table 5.4: Number of function evaluations after which SAMO-COBRA with the $\mathcal{P}_{hv}$ infill criterion achieved the same hypervolume for the test functions as SMES-RBF.

Function	SMES-RBF	PHV-SAMO-COBRA
BNH	200	50
BNH	500	122
SRN	200	23
SRN	500	27
TNK	200	24
TNK	500	194
OSY	500	14
OSY	1000	14
OSY	2000	14
TRICOP	200	12
TRICOP	500	12
BICOP1	500	56
BICOP2	500	31
BICOP2	1000	31
BICOP2	2000	38
BICOP2	5000	82

As shown in Table 5.4, the number of function evaluations for SAMO-COBRA is much smaller. For the BICOP1 test function, the Nadir point reported in the original paper [44] is [3.458533.44905]. The objective scores of BICOP1 can only be positive; therefore, the absolute maximum achievable hypervolume should be smaller than $3.45853 \cdot 3.44905 \approx 11.93$. Interestingly, the hypervolume results from SMES-RBF algorithm after 1000, 2000, and 5000 function evaluations, as reported in the original paper [44], are higher than 12 (which is impossible). A comparison between the SMES-RBF and SAMO-COBRA algorithm could therefore not be made for the BICOP1 problem with more than 500 function evaluations.

Convergence Plots. To further inspect the performance of the algorithms over time, convergence plots are made for the BNH and TRICOP test functions. The convergence plots show the HV score computed after every iteration. In the convergence experiments, the same estimation of Nadir points as in the original SMES-RBF paper [44] are used as the reference points. The convergence of the HV on the BNH test function can be found in Figure 5.2. The convergence of the HV on the TRICOP test function can be found in Figure 5.3.

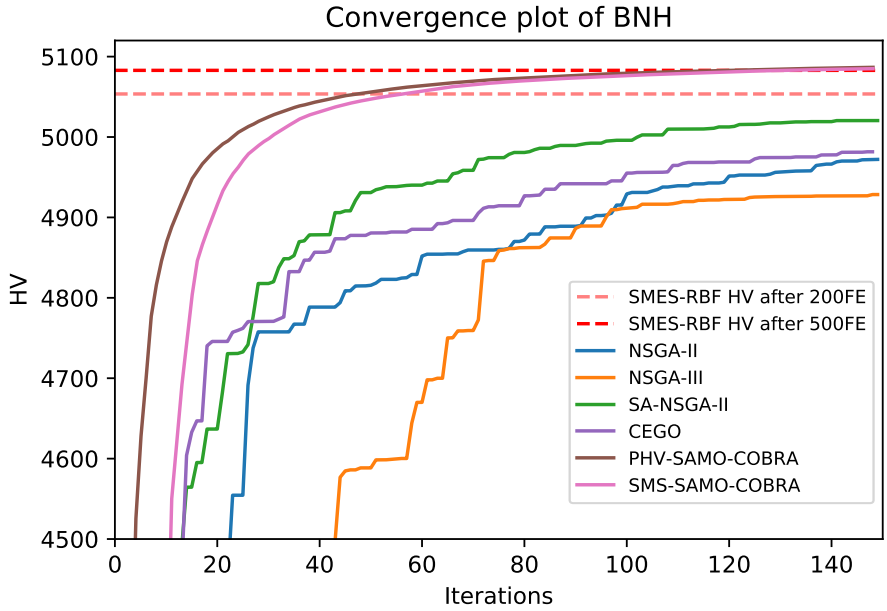


Figure 5.2: Convergence plot on BNH test problem for the NSGA-II, NSGA-III, SA-NSGA-II, CEGO, $\mathcal{P}_{hv}$-SAMO-COBRA, $\mathcal{S}$-metric-SAMO-COBRA algorithms. . The dashed lines represents the final obtained Hypervolume of SMES-RBF after 200 and 500 function evaluations.

5.1.5 Discussion $\mathcal{P}_{hv}$ vs. $\mathcal{S}$-metric Infill Criterion

An interesting conclusion from all the experiments is that the exploiting strategy of the $\mathcal{P}_{hv}$ infill criterion leads in most cases to the highest HV and to the least number of required function evaluations to obtain the 95% threshold. It is no surprise that this exploiting strategy works well in a constraint multi-objective setting, since a similar effect was already shown by Rehbach et al. [126]. Rehbach et al. show that in the single objective case, it is only useful to include an expected improvement infill criterion if the dimensionality of the problem is low, if it is multimodal, and if the algorithm can get stuck in a local optimum. The results in Table 5.2 and Table 5.3 allow us to give the following advice based on empirical results: When searching for a set of Pareto-optimal solutions, an uncertainty quantification method should not be used. This is due to the fact that, when searching for a trade-off between objectives, the algorithm is forced to explore more of the objective space in the different objective directions. The exploration of objectives stimulates diversity, which makes the algorithm less likely to

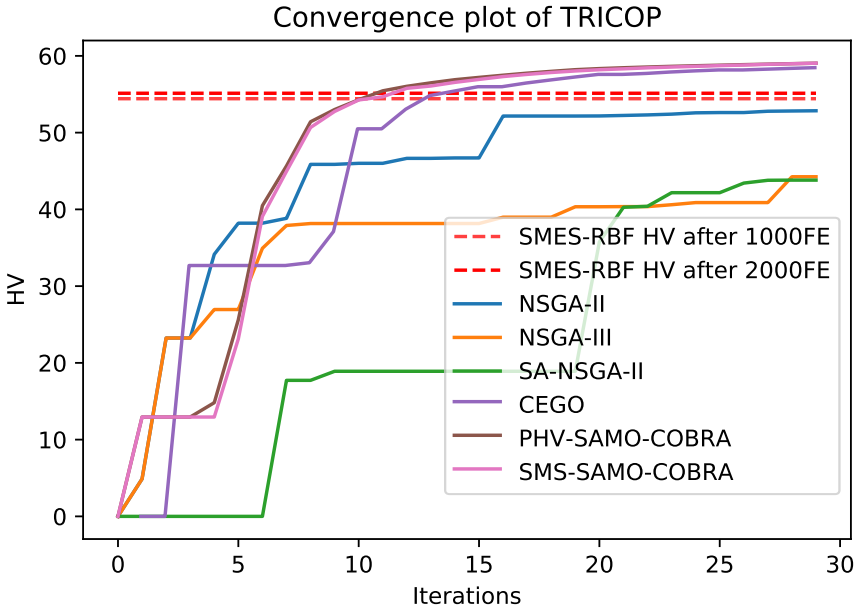


Figure 5.3: Convergence plot on TRICOP test problem for the NSGA-II, NSGA-III, SA-NSGA-II, CEGO, $\mathcal{P}_{hv}$-SAMO-COBRA, $\mathcal{S}$-metric-SAMO-COBRA algorithms. The dashed lines represent the final obtained Hypervolume of SMES-RBF after 1000 and 2000 function evaluations.

get stuck in a local optimum, thereby making the uncertainty quantification method redundant.

5.1.6 Conclusion and Future Work on Multi-Objective Optimization

In this paper, two variants of the SAMO-COBRA algorithm are introduced, based on using two different infill criteria: S-Metric-Selection ($\mathcal{S}$-metric) and Predicted Hypervolume ($\mathcal{P}_{hv}$), of which the latter is more exploitative than the former. The performance of the two SAMO-COBRA variants is compared to five other state-of-the-art algorithms: SA-NSGA-II, NSGA-II, NSGA-III and SMES-RBF. On 17 out of the 18 test functions, SAMO-COBRA with the $\mathcal{P}_{hv}$ infill criterion showed similar or better results. On the C3DTLZ4 test function, SA-NSGA-II obtained significantly better results. This can be explained that this function benefits much more from exploration than exploitation.

The SAMO-COBRA algorithm with the $\mathcal{P}_{hv}$ infill criterion showed to be very efficient at solving constraint multi-objective optimization problems in terms of required function evaluations. We speculate that this exploiting infill criterion works best in most cases because of the characteristics of multi-objective problems. While dealing with multi-objective problems, the algorithm is already forced to explore more of the objective space, making the uncertainty quantification method redundant.

The final conclusion from this research is that further research efforts should be put into creating infill criterion that can propose multiple solutions simultaneously. This way, in each iteration, evaluations can be run in parallel, and wall clock time can be reduced even further. That is why in the next section the multi-point infill criterion is introduced.

5.2 Parallel Multi-Objective Optimization

Algorithm classes that can deal with computationally expensive constraint multi-objective problems in parallel include multi-objective variants of evolutionary algorithms [56] and of Bayesian optimization [107]. In general, the former offers naturally built-in parallelism while typically requiring more function evaluations and the latter is more efficient in terms of function evaluations while typically not allowing for parallelism.

As described in earlier related work in Section 5.1.1, researchers have extended evolutionary algorithms by using surrogate models trained on the evaluated search points to allow for a fast prediction of objective and constraint function values for new candidate solutions (infill points), making them more efficient while keeping the benefits of parallelism [109]. A state-of-the-art algorithm from this class is for example the earlier described algorithm *Surrogate-assisted Non-dominated Sorting Genetic Algorithm* (SA-NSGA-II) [19].

Traditionally Bayesian Optimization algorithms on the other hand use an infill criterion to find a good solution on surrogate models. These infill criteria traditionally only propose one solution per iteration. Where the use of surrogates can potentially reduce the number of required evaluations to find optimum solutions, the infill criterion that proposes one solution per iteration drastically increases the time since all promising solutions have to be evaluated in series instead of in parallel.

Solving expensive optimization problems faster can, according to [92], be done in three different ways:

1. problem approximation and substitution,
2. algorithm design enhancement,
3. parallel and distributed computation.

In this section, it is demonstrated how a variety of these techniques are combined in one algorithm.

As mentioned before, Bayesian optimization algorithms approximate the fitness and constraint functions by using surrogates. In each iteration, the algorithm finds a new promising solution by optimizing an acquisition function using the response surface of the surrogates. Over time, different acquisition functions have been published for different surrogate models and for different purposes e.g; for single objective optimization [85], with an emphasis on exploration/exploitation [126], for constraint optimization [13], for parallel optimization [72], multi-objective optimization [118], and for constraint multi-objective optimization [154]. However, not much attention has been spent on an acquisition function that can both handle multiple constraints, multiple objectives, and propose multiple solutions for evaluation in parallel in an efficient manner.

For many real-world problems, candidate solutions can be evaluated in parallel using large computer clusters and multiple simulations. To make use of these resources, the optimization algorithm needs to be able to propose multiple candidate solutions in each iteration. Evaluating multiple solutions in parallel can reduce the total wall clock time significantly. Following the example of Li et al. [72], the total evaluation time, also referred to as the total cost of solving a computationally challenging optimization problem can be formulated as follows: $Totalcost = O(C) \cdot O(N)$. Here $O(C)$ is the average cost of the expensive evaluation, and $O(N)$ the average number of iterations of the optimization algorithm until a satisfactory solution is found. If two expensive evaluations can be run in parallel the cost can already be cut in half in terms of wall clock time ($p = 2$), i.e., $Totalcost = \frac{O(C) \cdot O(N)}{p}$. Obviously, when p solutions are proposed per iteration, the total cost can also be reduced by a factor p. The downside of proposing multiple solutions simultaneously is that the new batch of p solutions is selected based on the surrogates trained on $p - 1$ less samples as opposed to the sequential optimization procedure (where $p = 1$). This means that using parallel evaluations potentially results in additional required function evaluations compared to a sequential optimization run with a single solution per iteration.

To propose multiple solutions per iteration, in this section a new acquisition function is proposed that incorporates problem approximation and substitution, algorithm

design enhancement, and parallel and distributed computing techniques. This acquisition function is introduced and used in the SAMO-COBRA to demonstrate the effectiveness of proposing multiple solutions simultaneously. This makes the parallel SAMO-COBRA algorithm capable of doing multi-objective optimization while dealing with constraints and doing batch or one-shot optimization. The results of the experiments with this new infill criteria indicate that the missed information per iteration can also be beneficial since it also provides a means of exploration.

5.2.1 Related Work

Traditionally, evolutionary algorithms, genetic algorithms, and particle swarm optimization are population-based [9]. The evaluations of these populations can naturally be parallelized. However, evolutionary algorithms have the downside that they require a lot of function evaluations because they move in small steps before they converge to the global optimum.

On the other hand, Bayesian optimization does not require a lot of function evaluations and is used in case the objective and/or the constraint functions are expensive to evaluate. These surrogate-assisted optimization algorithms however typically do not use acquisition functions that can propose multiple solutions simultaneously. Allowing the surrogate-assisted algorithms to only propose one solution per iteration, which leads to longer running times and ineffective use of available resources.

Parallel Single Objective Optimization

According to a survey on parallel single objective optimization [72], the three most obvious techniques for parallelization are; multi-start local searches (if derivatives of the objective function are available), multiple parallel optimization runs (optionally in different sub-regions), and as described above with a population of designs. Other parallelization techniques often tend to combine different acquisition functions with different hyper-parameters to balance exploration and exploitation. Wang et al. [165] for example proposed a single objective multi-point acquisition function for Bayesian optimization. This acquisition function is based on the moment-generating function where the expected improvement is raised to the power t. For different values of t, the moment-generating function will therefore result in different proposed solutions with different trade-offs between exploration and exploitation.

Other techniques used to select p different solutions simultaneously are, for example:

- One way is to optimize a single point acquisition function, then assume that the surrogate prediction is accurate by adding the prediction to the evaluated solutions, and then optimize the acquisition $p - 1$ more times until p solutions are found [65]. This strategy is better known as the Kriging believer.
- It is also possible to use different surrogates (or weighted combinations of surrogates) fitted on the same data and optimize the infill criteria on these different surrogate models [75].

Parallel Multi-Objective Optimization

Besides the algorithm described in this paper, several surrogate-assisted multi-objective algorithms are already proposed where multiple points are proposed per iteration, e.g. MIP-EGO [137], MMBO [164], MOPLS-N [3]. The downside is that they all lack a constraint handling mechanism and fail to propose solutions on the constraint boundaries.

Mixed-Integer Parallel Efficient Global Optimization (MIP-EGO) [137] for example is designed to automatically optimize the configuration of artificial neural networks. MIP-EGO uses multiple random forests as surrogates and different infill criteria are optimized to propose different solutions simultaneously.

Wada and Hino proposed MMBO [164], a Bayesian multi-objective multi-point optimization algorithm together with a gradient approximation of the acquisition function. This algorithm proposes multiple points simultaneously in every iteration based on multi-point expected hypervolume improvement. This algorithm uses the expected hypervolume improvement as infill criteria and therefore uses the uncertainty quantification of the solutions to balance exploration and exploitation.

Akhtar and Schoemaker proposed MOPLS-N [3], a Multi Objective Population-based Parallel Local Surrogate-Assisted Search. MOPLS-N uses Radial Basis Functions (RBF) as surrogates, uses parallel local candidate search from the parent population centers, and uses boxed hypervolume improvement to judge one candidate solution in a box around one center.

Constraint Parallel Multi-objective Optimization

Additionally, as also mentioned in the survey [72], there is still a lack of well-performing adaptive sampling algorithms for constraint optimization. Constraint optimization is traditionally done by making use of penalty functions [72]. Tuning these penalty functions demands a lot of function evaluations [13]. To save function evaluations

during the optimization process, just like for the objective functions, surrogates can be used to model the constraint functions.

A few multi-objective optimization algorithms are found with both a constraint handling mechanism and capable of proposing multiple solutions per iteration:

1. GOMOEI is a Generalized Asynchronous Multi-objective Expected Improvement infill criteria (GAMOEI) proposed by Wauters et al. [167]. GAMOEI allows multiple points to be selected for evaluation asynchronously while balancing exploration and exploitation in an adaptive manner. The expected improvement infill criteria depends on the regular multi-objective expected improvement raised to a higher power. Constraints are dealt with by multiplying the probability of feasibility with the expected improvement. In their experiments, this however resulted in undesirable points far away from the Pareto frontier with little to no points on the constraint boundaries.

2. cK-RVEA is a many-objective reference vector-guided evolutionary algorithm that uses Kriging models as surrogates for the objectives and deals with the constraints by only using the feasible solutions for surrogate training [39]. Because this algorithm has as a basis an Evolutionary Algorithm, it has naturally built-in parallelism.

3. EGMOCO is a constraint multi-objective optimization algorithm that uses Kriging as a surrogate and exploits four different acquisition functions to propose multiple feasible Pareto-optimal solutions per iteration [173]. These four different acquisition functions all result in different proposed solutions so at maximum, four different solutions can be proposed and evaluated per iteration.

4. SBMO is a multi-objective algorithm that uses Kriging models as a surrogate for both the constraints and objectives. Because of the scalarization of the objectives, it can propose as many solutions per iteration as scalarizations are possible [74]. However, when the number of decision parameters increases the Kriging surrogates quickly become impractical to use.

One Shot Optimization

One-shot optimization [22, 24] or global surrogate modeling can be characterized by surrogate-assisted optimization algorithms where a surrogate is fitted only once with training data of an initial sample. After the surrogate is fitted, an optimal solution (or set of optimal solutions) is found on the surrogate, the obtained solutions are evaluated

and the algorithm terminates. This means that in contrast to other surrogate-assisted optimization algorithms, there is no evaluation budget for adaptive sampling. One-shot optimization is very popular and a classical approach in the maritime [130], automotive [131], aerospace [111] and other engineering domains. As already stated in the introduction of this Section, a lot of potentially available information for the last evaluation is missing when using this approach as all new solutions should be found based on the first initial samples. A benefit of one-shot optimization is that when a lot of computational resources are available they can easily be exploited.

5.2.2 Multi-Point Acquisition function

The related work gave inspiration for a new multi-point acquisition function that is introduced in this section. This new multi-point acquisition function is a reformulation of the single-point acquisition function from SAMO-COBRA as formally described in Equation 5.1. This single point acquisition function can also be used to propose multiple solutions simultaneously. For this to work, first the optimization problem should be reformulated so that multiple solutions can easily be optimized and judged on solution quality simultaneously. The reformulation of the solution vector is done by simply concatenating different solutions in one big solution vector. Suppose one solution contains d decision variables, then p solutions together can be formulated as a vector of $d \cdot p$ real values $\mathbb{R}^{p \cdot d}$. In this formulation, the first d values represent the first solution, whereas the last d values in this vector represent the p^{th} solution.

Given the p solutions ($\mathbf{x}_i, i \in \{i, \ldots, p\}$), and the cheap RBF surrogate for each objective ($f_i()', i \in \{i, \ldots, k\}$), also p predictions can be made for each objective. Since there are p solutions and k objectives, the RBF predictions can be combined in a vector of $p \cdot k$ objective function values as follows:

$$\mathbf{F} = (f'_1(\mathbf{x}_1), \ldots, f'_k(\mathbf{x}_1), \ldots, f'_1(\mathbf{x}_p), \ldots, f'_k(\mathbf{x}_p))$$

Here the vector $\mathbf{F}$ has a size of $p \cdot k$. The p solutions with the corresponding $p \cdot k$ predictions for the k objectives can, after this step, be split into the matrix $\mathbf{F}$ with p solutions (as rows) with k objective values (as columns). These p solutions can then be mapped to the objective space so that their combined performance in terms of hypervolume contribution can be judged. The judging of how good the combined p solutions are again computed with the $\mathcal{P}_{hv}$ infill criteria resulting in a Multi-Point Acquisition Function ($\mathcal{MP}_{hv}$). The hypervolume contribution of a set of solutions can be computed with the individual hypervolume contribution of each

solution minus the overlap. Because this multi-point acquisition function evaluates p solutions simultaneously, it will automatically prefer a set of solutions with diverse objective scores above a set of similar solutions with similar objective scores. This is the case because a set with diverse solutions with little overlapping hypervolume will dominate more objective space compared to a set of solutions with very similar scores with a lot of overlapping hypervolume. After the objectives are predicted with the RBFs for all p solutions, the $p \cdot k$ is then translated with the multi-point acquisition function into a single real value which represents the hypervolume contribution of the p solutions. Predicting p solutions simultaneously does not increase the total number of RBF surrogates, only the RBF surrogates are now used p times when evaluating p new solutions in parallel.

A similar formulation and strategy is used for the constraints. Because multiple solutions are now to be dealt with, also all the p solutions should be judged on feasibility simultaneously. Each solution has m constraints, leading to $p \cdot m$ constraint values to consider. With the RBF surrogates ($\mathbf{g}$) representing the m constraints, each RBF surrogate (g_j) can be used p times to predict the constraint values for the p solutions. This results in one long constraint vector of length $p \cdot m$, the first m constraint values represent the m constraint values for the first solution, the last m constraint predictions represent the constraints for the p^{th} solution.

With this new formulation for p solutions simultaneously, the multi-point acquisition optimization problem can be mathematically represented in the following way:

$$(\mathbf{x}_1^*, \ldots, \mathbf{x}_p^*) \in \underset{\mathbf{x}_i \in \Omega \subset \mathbb{R}^d}{\operatorname{argmax}} \mathcal{MP}_{hv}(\mathbf{f}'(\mathbf{x}_1), \ldots, \mathbf{f}'(\mathbf{x}_p))$$
$$\text{subject to } \mathbf{g}'(\mathbf{x}_i) \leq \mathbf{0}$$

A visual representation of the multi-point acquisition function is given in Figure 5.4.

Integration of Multi-Point acquisition function in SAMO-COBRA

The newly formulated acquisition function can be directly integrated into the SAMO-COBRA algorithm. The SAMO-COBRA with the new acquisition function is very similar to the original acquisition function. The difference is that now expensive evaluations can be evaluated in parallel. To maximally exploit the parallelism, the initial size of the design of experiments is now set to $\max(p, d+1)$. After the initial sample is evaluated, the all RBF model variants are again fitted for every objective and constraint independently. In the first iteration again the default RBF configuration is

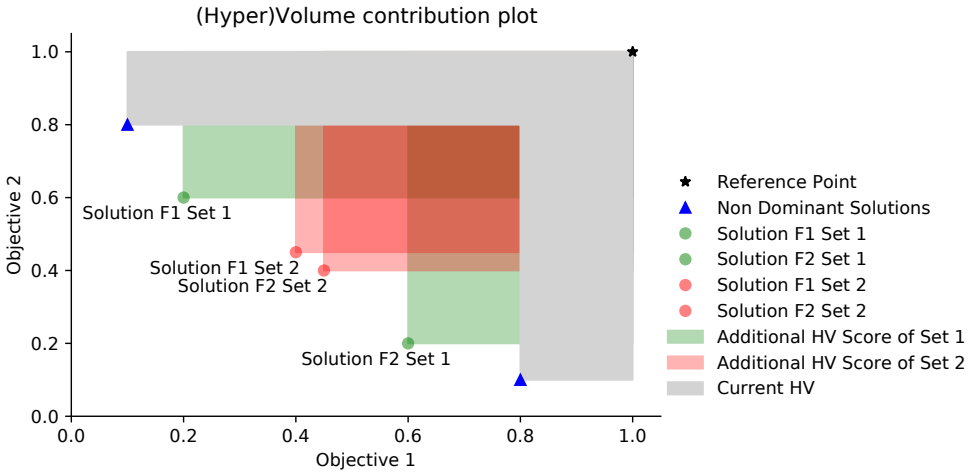


Figure 5.4: Visual representation of hypervolume contributions of two sets containing two solutions each. The hypervolume contribution of set 1 is equal to $2(0.6 \cdot 0.2) - (0.2 \cdot 0.2) = 0.2$. The hypervolume contribution of set 2 is equal to $2(0.35 \cdot 0.4) - (0.35 \cdot 0.35) = 0.1575$. So although the individual hypervolume contributions of the solutions of set 2 are higher compared to the individual hypervolume contributions of the solutions in set 1, the total hypervolume contribution of set 2 is smaller compared to the total hypervolume contribution of set 1. This makes set 1 more desirable compared to set 2.

chosen and the new acquisition function is optimized.

For the optimization of the multi-point acquisition function, any optimizer capable of optimizing one objective and dealing with multiple constraints can be chosen. For the integration in SAMO-COBRA, the COBYLA algorithm is again selected for this task.

By letting COBYLA start from a randomly generated vector of length $p \cdot d$ representing p solutions, COBYLA iteratively also optimizes these p solutions. Important to note is that COBYLA still does not use the real objective and constraint functions but the RBFs of the constraint and the RBFs of the objectives to optimize the acquisition function.

Experiments showed that the optimization problem characteristics like the number of decision variables d, the number of constraints m, the number of objectives k, and the number of solutions to be optimized in parallel p, all have an influence on whether COBYLA can converge to good solutions. The experiments show that more random starting points and larger evaluation budget for COBYLA lead to better results. However, more starting points and larger evaluation budgets for COBYLA also

lead to higher computational costs. Therefore, as a rule of thumb, it is recommended to let COBYLA start from $2(d+m+k)$ solutions when using the single point acquisition function, let COBYLA converge from $4(d+m+k)$ when using the multi-point acquisition function, and when doing one shot optimization, let COBYLA start from $8(d+m+k)$ solutions. A similar rule is created for the evaluation budget of COBYLA: The budget for using the single point acquisition function is $50(d+m+k)$, for using multi-point acquisition function $100(d+m+k)$, and for one shot optimization the evaluation budget for COBYLA is $200(d+m+k)$. After COBYLA has converged from the starting points, the solution set with the highest acquisition score is selected to be evaluated on the real objectives and constraint functions. If COBYLA can not find any feasible solutions, the solution set with the smallest cumulative constraint violation is selected and evaluated on the real objective and constraint functions.

Now instead of evaluating only one solution at a time, all p solutions are evaluated in parallel with the real objective and constraint functions. The results are added to the solution archive, and the RBF approximation error is again checked. Selection of the new best RBF modeling strategy is now not done by checking the approximation error from the solutions on the Pareto frontier and the last four evaluated solutions. Instead, the approximation error from the solutions on the current Pareto frontier and the last $2 \cdot p$ evaluated solutions are taken into consideration when selecting the best RBF modeling strategy. This way, if the parallel number of evaluations p is large, then the algorithm doesn't get stuck in a local optimal RBF configuration.

The process of surrogate fitting, acquisition function optimization, solution evaluations in parallel, and RBF strategy selection continues until the evaluation budget is exhausted. The SAMO-COBRA algorithm continues until the evaluation budget is exhausted. Note that this is not equal anymore to the number of iterations except for when $p = 1$ is chosen which is also still possible with the use of this acquisition function.

One Shot Optimization

The new Acquisition function can also be used for one-shot optimization. In the one-shot optimization configuration, the initial sample is of a size equal to half the evaluation budget. After the initial Halton sample is evaluated, the RBFs with the different configurations are fitted and the best RBF configurations are selected. Selection of the best input transformation and RBF kernel can in this case not be done based on historic approximation error. Nor can the best configuration be selected based on the RBFs trained with all the input data. Instead 10-fold cross-validation is used to select

the RBF kernel and transformation strategy. Selecting the optimal RBF configuration based on 10-fold cross-validation requires some computation time, however spending half of the evaluation budget based on wrongly estimated solutions is for obvious reasons much more computationally expensive. After the selection of the optimal RBF configurations for each constraint and objective separately, the multi-point acquisition function is optimized. The multi-point acquisition function is optimized such that in one run all solutions for the other half of the evaluation budget can be found. Finally, the predicted optimal solutions are evaluated with the real objectives and constraint functions and the algorithm terminates.

5.2.3 Multi-Point Acqusition Function Experiments

To test the performance of the multi-point acquisition function in the SAMO-COBRA algorithm and the one shot option several experiments are conducted. In the experiments, different batch (parallel candidate solution sizes p) sizes are tested for the multi-point acquisition function: 1 (original), 2, 3, 4, 5, 6, 10 and 20. Bigger batch sizes are not considered because then multi-point optimization strategy becomes too similar to one shot optimization. The test functions from Table 2.1 with the exception of the MW problems are selected for the experiments. Each test function is optimized in 11 independent runs with different seeds. Optimization of the test functions is done by using a reference point which is the worst possible objective score per function. The Nadir point [15] of the test functions is approximated by taking the extremes of the objective scores on the Pareto frontier from all combined experiment results. The hypervolume reported in the results of the experiments are calculated by computing the hypervolume between the Pareto frontier and the Nadir point. The algorithms variant, the experiments, and the raw results are also published on a dedicated Github page [144].

Hypervolume after Fixed Evaluation Budget

In the first experiment the hypervolume between the approximated Nadir point and the obtained Pareto frontier is calculated after a fixed evaluation budget. SAMO-COBRA with the different batch sizes has a total allowed evaluation budget of $40 \cdot d$. This evaluation budget leads to an initial Halton sample of $d+1$ samples and $39 \cdot d - 1$ iterations for the SAMO-COBRA algorithm with the single-point infill criteria. For batch sizes larger then 1, $\max(d+1, p)$ initial Halton samples and $\frac{40 \cdot d - \max(d+1,p)}{p}$ iterations are done.

Convergence Experiment

As explained before, with batch optimization, potentially a lot of wall clock time can be saved. The downside of proposing multiple solutions simultaneously is that the new batch of solutions is based on less information compared to when one solution would be added per iteration. In this experiment, it is tested how much information is lost per iteration, and on the other hand how much time can be saved. This is tested by taking 90% of the maximum achievable hypervolume as a threshold, then the algorithm convergence results can tell how much algorithm iterations and total number of function evaluations are required to achieve this hypervolume threshold for the different batch sizes.

One Shot Optimization Experiment

In the last experiment the algorithm and multi-point acquisition function is tested to see if it is capable of one shot optimization. The one shot optimization algorithm configuration is tested with 40 initial Halton samples and then in one iteration 40 new solutions are proposed with the multi-point acquisition function and then evaluated. The hypervolume between the Nadir Point and the obtained Pareto frontier is computed and compared to the hypervolume obtained with batch size 1.

5.2.4 Results

The results of the three experiments are presented in two Pareto frontiers, two convergence plots, and three tables. The overall results show that for test functions with a low feasibility rate, larger batch sizes lead to worse results after the same number of function evaluations. For other test functions with a higher feasibility rate, larger batch sizes can be very beneficial in terms of the required number of evaluations and therefore iterations.

Hypervolume Results

In Table 5.5 the mean hypervolume and standard deviation of the hypervolume between the Pareto frontier and Nadir point are given for the different test functions. As can be seen in the table, the hypervolume in most cases slightly decreases, and the standard deviation increases, when a larger batch size is chosen. In a few cases, the mean hypervolume is significantly better for larger batch sizes. It is expected that this

is the case because more exploration can be beneficial for functions that are hard to fit with RBF models with few initial data samples.

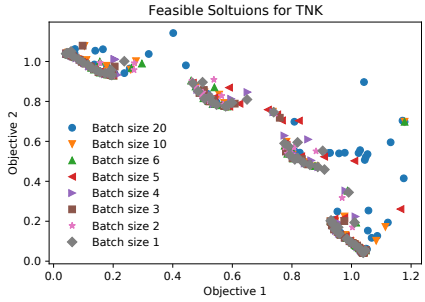
Table 5.5: Mean and standard deviation of hypervolume (hv) between Pareto frontier and Nadir point on set of test functions after $40 \cdot d$ function evaluations with different batch sizes (1,2,3,4,5,6,10,20) for the $\mathcal{MP}_{hv}$ infill criteria given 11 independent runs. HV Scores in **bold** indicate a higher mean compared to batch size 1. A * is added if the difference was significant according to the Wilcoxon rank-sum test with $p < 0.05$.

Function	Batch size 1		Batch size 2		Batch size 3		Batch size 4	
	hv	std	hv	std	hv	std	hv	std
BNH	4969.2	0.0133	4969.0	0.1	4967.5	1.5	4967.6	0.5
CEXP	3.7972	0.0005	3.7961	0.0015	3.7963	0.0016	3.7944	0.001
SRN	25019	5	25008	10	24977	16	24843	22
TNK	0.2988	0.0016	0.2966	0.0033	0.2965	0.0026	0.2949	0.0018
CTP1	0.2985	0.0001	0.2985	0.0001	0.2984	0.0001	0.2984	0.0001
C3DTLZ4	1.5446	0.0759	1.4288	0.17	1.3569	0.0018	1.2526	0.0473
OSY	12629	2	12352	97	12609	3	12526	71
TBTD	8052.6	48.5	7892.3	90.0	7690.2	153.9	7506.0	237.4
NBP	799579	190	800186	130	**799770***	258	798810	643
DBD	59.9960	0.0806	**60.0550***	0.0152	**60.0614***	0.0063	**60.0034**	0.0391
SPD	$5.511 \cdot 10^9$	$2 \cdot 10^6$	$\mathbf{5.513 \cdot 10^9}$	$3 \cdot 10^6$	$5.502 \cdot 10^9$	$3 \cdot 10^6$	$5.497 \cdot 10^9$	$8 \cdot 10^6$
CSI	7.5394	0.0038	7.5343	0.0049	**7.5438**	0.0064	7.5372	0.0077
SRD	2952123	95	2949030	574	2945522	755	2941958	935
WB	0.6375	0.0185	**0.6435**	0.0133	0.6373	0.0138	**0.6416**	0.0209
BICOP1	0.6640	0.0004	0.6609	0.0010	0.6442	0.0052	0.6226	0.0111
BICOP2	0.2549	0.0381	**0.2623**	0.0161	0.2289	0.0358	0.2294	0.0364
TRICOP	49.6407	0.0430	**49.6971***	0.0206	**49.7224***	0.0215	**49.6470**	0.0449
WP	$3.677 \cdot 10^{18}$	$5 \cdot 10^{15}$	$3.662 \cdot 10^{18}$	$3 \cdot 10^{15}$	$3.653 \cdot 10^{18}$	$9 \cdot 10^{15}$	$3.631 \cdot 10^{18}$	$1.4 \cdot 10^{16}$

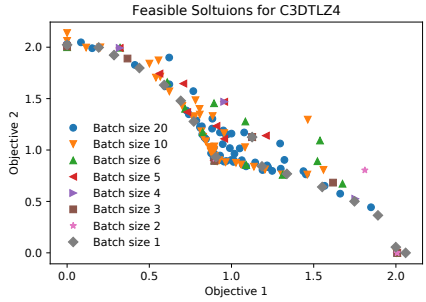
Function	Batch size 5		Batch size 6		Batch size 10		Batch size 20	
	hv	std	hv	std	hv	std	hv	std
BNH	4967.9	0.7	4967.6	1.2	4960.3	2.3	4949.8	5.7
CEXP	3.7964	0.0004	**3.7981***	0.0004	3.7925	0.0005	3.7794	0.0030
SRN	24729	53	24723	39	24583	97	24516	103
TNK	0.2953	0.0012	0.2985	0.0018	0.2957	0.0024	0.2676	0.0144
CTP1	0.2981	0.0001	0.2984	0.0002	0.2977	0.0003	0.2956	0.0008
C3DTLZ4	1.3375	0.0512	1.3933	0.0813	1.5091	0.0659	**1.5827**	0.0308
OSY	12396	139	12443	90	12073	105	11501	297
TBTD	7471.3	205.5	7419.4	300.2	7237.2	272.6	7213.8	231.5
NBP	798377	844	797753	1496	793709	2257	776697	1517
DBD	59.9676	0.0334	59.8961	0.0262	59.8108	0.0296	59.6967	0.0506
SPD	$5.497 \cdot 10^9$	$8 \cdot 10^6$	$5.474 \cdot 10^9$	$1.3 \cdot 10^7$	$5.369 \cdot 10^9$	$1.7 \cdot 10^7$	$5.259 \cdot 10^9$	$3.0 \cdot 10^7$
CSI	**7.5432**	0.0076	**7.5409**	0.0112	7.4329	0.018	6.8714	0.0704
SRD	2940695	1019	2939470	2003	2934512	941	2925597	3690
WB	**0.6475**	0.0128	0.6089	0.0263	0.6213	0.0169	0.5952	0.0125
BICOP1	0.6029	0.0160	0.5901	0.0139	0.4302	0.1118	0.3375	0.0809
BICOP2	0.2320	0.0356	0.2267	0.0160	0.2198	0.0435	0.2138	0.0250
TRICOP	**49.7100**	0.0402	**49.7270***	0.0259	49.5006	0.0825	49.3136	0.1001
WP	$3.583 \cdot 10^{18}$	$1.4 \cdot 10^{16}$	$3.556 \cdot 10^{18}$	$1.3 \cdot 10^{16}$	$3.492 \cdot 10^{18}$	$2.1 \cdot 10^{16}$	$3.471 \cdot 10^{18}$	$1.5 \cdot 10^{16}$

For two test functions, the obtained feasible solutions are plotted for the algorithm with different batch sizes. In Figure 5.5a all the obtained feasible solutions of the test problem TNK are presented. In this figure, it can be observed that the algorithm with batch size 1 rarely misses the Pareto frontier, while solutions of the larger batch sizes are often dominated by other solutions from these batch sizes. In Figure 5.5b all the obtained feasible solutions of the test problem C3DTLZ4 are presented. In this figure,

it can be observed that the solutions with the larger batch sizes show better coverage among the Pareto frontier versus the solutions from other batch sizes.



(a) Obtained feasible solutions in objective space for TNK test function. Different colors are used for different batch sizes in the acquisition function.



(b) Obtained feasible solutions in objective space for C3DTLZ4 test function. Different colors are used for different batch sizes the acquisition function.

Figure 5.5: Obtained Pareto Frontiers for TNK and C3DTLZ4 test function

Convergence Results

The results of the second experiment can be found in Table 5.6. This table clarifies that for the majority of the test functions, the threshold of 90% was reached before the allowed number of function evaluations. When comparing batch size 1 with the larger batch sizes for each test function. The best result with the least number of iterations on average required 75% less iterations, the trade-off is that the number of evaluations on average increases with 58% to find the 90% hypervolume threshold. So in the cases where time-consuming objective and constraint functions can be evaluated in parallel, the wall clock time can significantly be reduced.

In Figure 5.6a the convergence plot is given for the TNK test function. For this test function, the algorithm with different batch size combinations all converge to the approximated optimum except for batch size 20. In Figure 5.6b the convergence plot is given for the C3DTLZ4 function. Interestingly enough, in this convergence plot the extra exploration which is naturally included for larger batch sizes seems to be beneficial since the larger batch sizes 20, 10 and 6 perform better compared to batch sizes, 2, 3, 4, and 5.

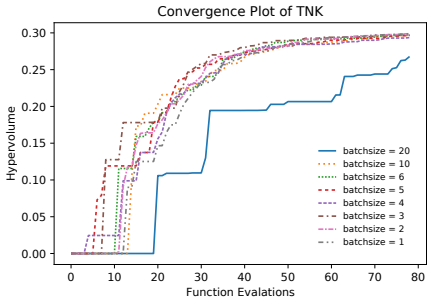
Table 5.6: Rounded mean number evaluations (Eval), mean number of algorithm iterations (Itr) given different batch sizes (1, 2, 3, 4, 5, 6, 10, 20) to achieve the hypervolume threshold. The threshold is 90% of the dominated area between the Nadir point and the Pareto frontier of all runs combined. A dash (-) indicates that the threshold is not achieved within $40 \cdot d$ function evaluations for any of the 11 runs, an arrow down (↓) indicates that not every run reached the threshold.

Function	Threshold	Batch size 1		Batch size 2		Batch size 3		Batch size 4	
		Eval	Itr	Eval	Itr	Eval	Itr	Eval	Itr
BNH	4496.2	9	9	10	5	10	4	8	2
CEXP	3.4380	9	9	11	6	11	4	12	3
SRN	22810	17	17	19	10	20	7	22	6
TNK	0.2775	44	44	47	24	48	16	48	12
CTP1	0.2717	13	13	15	8	15	5	15	4
C3DTLZ4	1.5788	197↓	197↓	219↓	110↓	-	-	-	-
OSY	11393	18	18	64	33	20	7	27	7
TBTD	7359.8	16	16	29	15	43	15	57↓	15↓
NBP	725935	16	16	15	8	14	5	19	5
DBD	54.133	14	14	15	8	15	6	14	4
SPD	$5.106 \cdot 10^9$	59	59	58	30	62	21	68	17
CSI	7.1691	120	120	124	62	119	40	118	30
SRD	2658080	14	14	14	7	16	6	17	5
WB	0.61745	94↓	94↓	88	45	106↓	36↓	65↓	17↓
BICOP1	0.59988	82	82	67	34	106	36	139	35
BICOP2	0.27667	353↓	353↓	338↓	169↓	-	-	356↓	89↓
TRICOP	45.3701	13	13	16	8	15	6	21	6
WP	$3.517 \cdot 10^{18}$	58	58	62	31	66	22	74	19

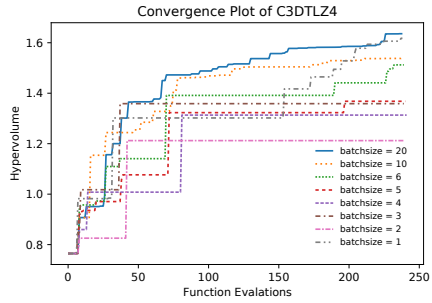
Function	Threshold	Batch size 5		Batch size 6		Batch size 10		Batch size 20	
		Eval	Itr	Eval	Itr	Eval	Itr	Eval	Itr
BNH	4496.2	8	2	9	2	13	2	21	2
CEXP	3.4380	13	3	11	2	17	2	31	2
SRN	22810	18	4	20	4	30	4	37	2
TNK	0.2775	46	10	44	8	49	5	68↓	4↓
CTP1	0.2717	19	4	16	3	19	2	34	2
C3DTLZ4	1.5788	-	-	-	232↓	24↓	187↓	10↓	
OSY	11393	37	8	25	5	39	4	125↓	7↓
TBTD	7359.8	61↓	13↓	36↓	6↓	76↓	8↓	77↓	4↓
NBP	725935	19	4	20	4	26	3	46	3
DBD	54.133	15	3	20	4	27	3	32	2
SPD	$5.106 \cdot 10^9$	62	13	77	13	103	11	140	7
CSI	7.1691	119	24	123	21	179	18	-	-
SRD	2658080	18	4	20	4	23	3	68	4
WB	0.61745	81	17	133↓	23↓	132↓	14↓	-	-
BICOP1	0.59988	206↓	42↓	322↓	54↓	-	-	-	-
BICOP2	0.27667	-	-	-	-	-	-	-	-
TRICOP	45.3701	16	4	17	3	19	2	33	2
WP	$3.517 \cdot 10^{18}$	92	19	103	18	-	-	-	-

One Shot Optimization Results

The Hypervolumes of the one-shot optimization algorithm experiments are presented in Table 5.7. Inspection of this table tells us that the test functions with a high



(a) Convergence plot for TNK function. Different colors are used for different batch sizes in the acquisition function.



(b) Convergence plot for C3DTLZ4 function. Different colors are used for different batch sizes in the acquisition function.

Figure 5.6: Obtained hypervolume convergence for TNK and C3DTLZ4 test function

feasibility rate tend to give much better results compared to test functions with a low feasibility rate. This indicates that the constraints are not well fitted after the initial sample and that more adaptive sampling steps lead to better constraint boundary approximation and therefore to better Pareto frontier approximations.

5.2.5 Discussion on Parallelization

Bayesian optimization is often used to optimize expensive black box optimization problems with long simulation times. Typically Bayesian optimization algorithms propose one solution per iteration. The downside of this strategy is the sub-optimal use of available computing power. To efficiently use the available computing power (or a number of licenses etc.) a multi-point acquisition function for parallel efficient multi-objective optimization algorithms is introduced. The multi-point acquisition function is based on the hypervolume contribution of multiple solutions simultaneously, leading to well-spread solutions along the Pareto frontier. By combining this acquisition function with a constraint-handling technique, multiple feasible solutions can be proposed and evaluated in parallel every iteration. The hypervolume and feasibility of the solutions can easily be estimated by using multiple cheap radial basis functions as surrogates with different configurations. The acquisition function can be used with different population sizes and even for one shot optimization. The strength and generalizability of the new acquisition function is demonstrated by optimizing a set of black box constraint multi-objective problem instances. The experiments show a huge time saving factor by using our novel multi-point acquisition function, while only marginally

Table 5.7: Mean and Standard deviation of hypervolume of the one-shot optimization algorithm configuration between the Nadir point and the obtained Pareto frontiers over 11 runs after 80 function evaluations with an initial Halton sample of 40. The results are compared to the result of the original infill criteria with batch size 1 by computing the hypervolume differences in a percentage.

Function	hv	std	Difference
BNH	4939.6	2	−0.60%
CEXP	3.6507	0.0240	−4.01%
SRN	23649	262	−5.79%
TNK	0.2044	0.0341	−46.18%
CTP1	0.2731	0.0091	−9.30%
C3DTLZ4	1.4308	0.0458	−7.95%
OSY	6144.9	1240.3	−105.52%
TBTD	6007.2	425.2	−34.05%
NBP	768803	4997	−4.00%
DBD	56.812	0.541	−5.60%
SPD	$2.9674 \cdot 10^9$	$3.058 \cdot 10^8$	−85.72%
CSI	5.9929	0.0472	−25.81%
SRD	2855825	61403	−3.37%
WB	0.5601	0.0126	−13.82%
BICOP1	0.4193	0.0482	−52.04%
BICOP2	0.0759	0.0296	−235.84%
TRICOP	47.750	0.798	−3.96%
WP	$3.198 \cdot 10^{18}$	$2.44 \cdot 10^{17}$	−14.98%

worsening the hypervolume after the same number of function evaluations. However, this claim only holds in cases where the evaluation of one of the objectives or constraints is computationally expensive and when they can be run in parallel. The results of the one shot optimization experiment however does not show very good results on problems that have a small feasibility ratio. Inspection of the results shows that the constraints are not well fitted after the initial sample and therefore, a lot of infeasible solutions are proposed in the one shot step. More adaptive sampling steps will lead to better constraint boundary approximation and therefore to more feasible solutions and therefore better Pareto frontier approximations.

5.2.6 Conclusion and Future Work on Parallel Optimization

A new acquisition function capable of multi-point multi-objective optimization is introduced and implemented together with a constraint handling mechanism. This new acquisition function is used to enhance the SAMO-COBRA algorithm, making the

algorithm able to propose multiple solutions per iteration. Experiments on a benchmark test set show that with larger batch sizes, in the ideal case on average 75% of the iterations can be saved, and therefore the waiting time can be reduced. This is especially interesting in cases where the evaluation of solutions is very time-consuming and when they can be evaluated in parallel. The new infill criteria offer the possibility to save wall-clock-time and give the user the power to better exploit the computational resources and the use of commercial licenses.

Future work will have to be put into dealing with multi-fidelity optimization problems, asynchronous function evaluations, and exploiting inexpensive functions to decrease wall clock time even further.

5.3 Expensive and Inexpensive Function Optimization

Real-world problems are often defined through multiple objectives and constraints, combined with the fact that objectives or constraints can be time-consuming ("expensive") to evaluate [14, 161, 172]. Expensive optimization problems are for example maritime design problems from Chapter 2 in which (commercial licenses of) finite element simulation or computational fluid dynamic tools are used for computing the performance characteristics of a design. These third-party software packages are computationally expensive to run, thereby increasing the overall duration of the optimization process. This leads to a very limited amount of allowed solution evaluations for the optimization algorithms.

Assuming that, at a maximum, a few hundred simulation runs are possible (i.e., solution evaluations of objective and constraint functions), the goal becomes to approximate the true Pareto front of feasible solutions as closely as possible with the given limited budget. To decrease the wall-clock-time, solution evaluations can be run in parallel as was shown in the experiments from Section 5.2.3. To decrease the wall-clock-time even more and to make fewer mistakes in the optimization process, the inexpensive constraint and objective functions (like volume objective, or main pariticulars check) can directly be used in the optimization algorithm instead of using a surrogate for them.

A state-of-the-art algorithm that can deal with similar problems is the recent *Inexpensive Constraint* extension of the SA-NSGA-II algorithm (IC-SA-NSGA-II [19]). The IC-SA-NSGA-II algorithm uses radial basis function surrogates *only for the ob-*

jectives and assumes that all constraints are inexpensive to evaluate.

The other algorithm that can be extended to exploit inexpensive functions is the SAMO-COBRA algorithm in combination with the multi-point acquisition function. Like SA-NSGA-II, the SAMO-COBRA algorithm uses radial basis function approximations for *all objectives and all constraints*. These two algorithms are designed with the purpose of modeling and optimizing surrogates of both the objective and constraint functions. However, there is a fundamental difference between the working of these two algorithms. While SA-NSGA-II and IC-SA-NSGA-II use a genetic algorithm's operators to create new candidate solutions, SAMO-COBRA uses a local search-based hypervolume maximization approach for creating new candidate solutions.

To facilitate a complete experimental comparison, a SAMO-COBRA variant that is inspired by IC-SA-NSGA-II's approach to differentiate between inexpensive constraints and expensive objectives is developed. This new variant however generalizes this approach and can exploit not only the inexpensive constraints but also the inexpensive objectives. The proposed *Inexpensive Objectives and Constraints*-SAMO-COBRA (IOC-SAMO-COBRA) allows the user to identify the expensive objectives and constraints, for which IOC-SAMO-COBRA will then use radial basis function surrogates, while it will use the inexpensive objectives and inexpensive constraints directly. A tabular overview of the four different algorithms and how they deal with expensive and inexpensive objectives and constraints is given in Table 5.8.

Algorithm	**Expensive constraints**	**Inexpensive constraints**	**Expensive objectives**	**Inexpensive objectives**
SA-NSGA-II	surrogate	surrogate	surrogate	surrogate
IC-SA-NSGA-II	direct	direct	surrogate	surrogate
SAMO-COBRA	surrogate	surrogate	surrogate	surrogate
IOC-SAMO-COBRA	surrogate	direct	surrogate	direct

Table 5.8: Overview of how the four algorithms deal with the (in)expensiveness of constraints and objectives. "Surrogate" means a surrogate replaces the objective/constraint, direct means that the objective/constraint is used without learning a surrogate for it.

5.3.1 Related Work

There is a growing interest in surrogate-assisted optimization [92, 84], surrogate-assisted constraint optimization [124], surrogate-assisted optimization in combination with parallelism [72], surrogate-assisted multi-objective optimization [38], and problems with heterogeneous evaluation times [4]. Different approaches have been developed for solving constraint multi-objective problems. However, very little research has

been done on surrogate-assisted algorithms that can deal with a mix of both expensive and inexpensive constraints and objective functions. There exists two algorithms that are very relevant and already partly address the problem:

1. **GP-CMOEA**, is like the IC-SA-NSGA-II algorithm a multi-objective optimization algorithm that uses both surrogates and exploits the inexpensiveness of the constraints to find feasible Pareto-optimal Solutions [170]. Due to the Gaussian Process regression surrogates, this method quickly becomes impractical when the number of parameters increases.

2. **CHVPEI** and **CHVPOI** are a bi-objective optimization acquisition functions that exploit the inexpensiveness of only the second objective that is always assumed to be inexpensive [95]. For the first objective, the expected improvement or the probability of improvement are computed depending on the infill criteria. This infill criteria however still needs to be extended for more than 2 objectives, and cannot deal with constraints yet.

However, an algorithm that can deal with a mix of expensive and inexpensive objectives and constraints has not been proposed yet. It is for this reason that in this section a parallel constraint multi-objective optimization algorithm is proposed that is capable of dealing with mixed expensiveness of objective and constraint functions.

In the following subsections, the closely related relevant methods IC-SA-NSGA-II that is used as reference algorithm is described in more detail.

IC-SA-NSGA-II

An extension of SA-NSGA-II has been proposed to address optimization problems where objectives are computationally expensive, but the constraints are not [19]. For such problems, the optimization method shall exploit the asymmetry of expensiveness, or in other words, the fact that one can collect significantly more information regarding the feasibility of a solution before having to run an expensive simulation. The novelty of the proposed method is the constraint sampling for finding feasible designs in the first optimization cycle. The challenge of finding a feasible yet diverse set of solutions is addressed by incorporating a Riesz s-energy [77] based sampling method [21] modified for constraint search spaces. Furthermore, to make IC-SA-NSGA-II more efficient for the optimization of highly constraint problems (still with inexpensive constraints), the embedded surrogate-based optimization loop has been extended by a repair operator applied to each solution after mating [87]. The repair operator ensures that only

feasible solutions are evaluated (on the surrogates and on the expensive functions) and has demonstrated to be effective for problems with complex constraints. The novel evaluated solutions are added to the archive which is used in the next iteration to retrain the surrogates. This continues until the objective evaluation budget has been exhausted. A more extensive explanation of the IC-SA-NSGA-II algorithm is given in [19].

5.3.2 Inexpensive Function Exploitation

In the original SAMO-COBRA algorithm for every objective and constraint function, an RBF surrogate is used during the search for new candidate solutions. In the IOC-SAMO-COBRA extension, one or more of the RBFs can be replaced with the real inexpensive constraint or objective function. Instead of finding good solutions on the RBFs, in IOC-SAMO-COBRA the inexpensive constraints and objectives are used directly during the search for feasible Pareto efficient solutions that contribute HV to the Pareto front. The direct use of inexpensive functions can be beneficial because the real functions do not make approximation errors like RBF surrogates do in unseen regions. This should, especially in the early iterations, lead to better results compared to the use of RBFs since in early iterations the RBF approximation error might still be large. Besides a benefit during the early iterations, inexpensive constraints can also be exploited when finding the Pareto fronts of optimization problems with very few feasible solutions. The pseudocode of the IOC-SAMO-COBRA algorithm is given in Algorithm. 3.

Hypervolume Maximization

The IOC-SAMO-COBRA algorithm uses the same acquisition function as presented in Section 5.2.2 (line 12 in Algorithm 3). While the original SAMO-COBRA algorithm with the multi-point acquisition function used the RBF surrogates for each constraint and each objective, the IOC-SAMO-COBRA algorithm does this differently. If one or more of the constraints or objectives are inexpensive to evaluate, then this can be indicated by the user. This allows the IOC-SAMO-COBRA algorithm to directly use them to compute the corresponding function values. These inexpensive functions in combination with the RBF approximations of the expensive functions are used by COBYLA to find the most promising solution set that is expected to contribute the most hypervolume.

When optimization of the acquisition function with COBYLA, also with the use

Algorithm 3: IOC-SAMO-COBRA. **Input:** Number of decision variables d, objective functions $\mathbf{f}(\mathbf{x})$, split where required into expensive objective function(s) $\mathbf{f}_e(\mathbf{x})$, computationally inexpensive objective function(s) $\mathbf{f}_c(\mathbf{x})$, constraint function(s) $\mathbf{g}(\mathbf{x})$, split where required into expensive constraint function(s) $\mathbf{g}_e(\mathbf{x})$, computationally inexpensive constraint function(s) $\mathbf{g}_c(\mathbf{x})$, decision parameters' lower and upper bounds $[\mathbf{x_{lb}}, \mathbf{x_{ub}}] \subset \mathbb{R}^d$, reference point $\mathbf{ref} \in \mathbb{R}^k$, number of initial samples N_{init}, maximum evaluation budget N_{max}, RBF strategy domain $\Phi = \{Cubic, Gaussian, Multiquadric, InverseQuadratic, InverseMultiquadric, ThinPlateSpline\} \times \{\text{PLOG}, standardized\}$, acquisition function HV.
Output: Evaluated solutions.

1 **Function** IOC-SAMO-COBRA(d, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{x_{lb}}, \mathbf{x_{ub}}$, $\mathbf{ref}$, N, N_{max}, $RBF_{kernels}$):
2 $\quad \mathbf{X} \leftarrow \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ ▷ Generate initial design, $\mathbf{X} \in \mathbb{R}^{d \times N}$
3 $\quad \mathbf{F} \leftarrow \mathbf{f}(\mathbf{X})$ ▷ Evaluate objective functions, $\mathbf{F} \in \mathbb{R}^{k \times N}$
4 $\quad \mathbf{G} \leftarrow \mathbf{g}(\mathbf{X})$ ▷ Evaluate constraint functions, $\mathbf{G} \in \mathbb{R}^{m \times N}$
5 $\quad \mathbf{h} \leftarrow \{\mathbf{f}_e \cup \mathbf{g}_e\}$ ▷ Union of expensive obj. and constr. functions
6 $\quad \varphi^* \leftarrow \{(Cubic, standardized) \mid \forall\, h \in \mathbf{h}\}$ ▷ Init best RBF, $\varphi^* \in \Phi$
7 $\quad \mathbf{E} \leftarrow \{0 \mid \forall\, h \in \{\mathbf{h} \times \Phi\}\}$ ▷ Init RBF approx. errors for each configuration/
8 $\quad j \leftarrow N$ ▷ Initialize expensive evaluation counter
9 $\quad$ **while** $j < N_{max}$ **do**
10 $\quad\quad \mathbf{S}^{\Phi} \leftarrow \{\text{FITRBF}(\mathbf{X}, h, \Phi, \mathbf{x_{lb}}, \mathbf{x_{ub}}) \mid \forall\, h \in \mathbf{h}\}$ ▷ Fit RBF with all Φ strategies for all $\mathbf{h}$
11 $\quad\quad \mathbf{S}^{\varphi^*} \leftarrow \left\{\mathbf{S}^{\varphi^*} \mid \forall\, h \in \mathbf{h}\right\}$ ▷ Select best RBF surrogate based on line 6 or 17
12 $\quad\quad \mathbf{x}_1^*, \ldots, \mathbf{x}_p^* \leftarrow \text{MAX}(\text{HV}, p, \mathbf{ref}, \mathbf{S}^{\varphi^*}, \mathbf{f}_c, \mathbf{g}_c)$ ▷ Get p new solutions based on HV
13 $\quad\quad j \leftarrow j + p$ ▷ Increase iteration counter to new matrix sizes
14 $\quad\quad \mathbf{X} \leftarrow [\mathbf{X}, \mathbf{x}_1^*, \ldots, \mathbf{x}_p^*]$ ▷ Add p new solution vectors, $\mathbf{X} \in \mathbb{R}^{d \times j}$
15 $\quad\quad \mathbf{F} \leftarrow [\mathbf{F}, \mathbf{f}(\mathbf{x}_1^*), \ldots, \mathbf{f}(\mathbf{x}_p^*)]$ ▷ Add vectors of evaluated objectives, $\mathbf{F} \in \mathbb{R}^{k \times j}$
16 $\quad\quad \mathbf{G} \leftarrow [\mathbf{G}, \mathbf{g}(\mathbf{x}_1^*), \ldots, \mathbf{g}(\mathbf{x}_p^*)]$ ▷ Add vectors of evaluated constraints, $\mathbf{G} \in \mathbb{R}^{m \times j}$
17 $\quad\quad \text{HV}, \varphi^*, \mathbf{E} \leftarrow \text{SELECTBESTSTRATEGY}(\mathbf{E}, S^{\Phi}, \mathbf{X}, \mathbf{F}, \mathbf{G})$ ▷ Update HV, RBF approx. errors $\mathbf{E}$, and new best RBF configuraiton φ^* based on $\mathbf{E}$
18 $\quad$ **end**
19 **return** ($\mathbf{F}$, $\mathbf{G}$, $\mathbf{X}$)

of inexpensive functions where possible, COBYLA can get stuck in local optima. To overcome this problem also in the IOC-SAMO-COBRA algorithm, COBYLA is run in parallel starting from multiple random starting points.

After all COBYLA instances have converged, all feasible solutions found are 10000 times randomly combined in groups of size p. Since there are $\binom{16 \cdot p \cdot d}{p}$ such groups, for small values of p and d fewer combinations are sufficient. However, due to the negligible computational effort, it is decided to fix this number to 10000. The set of p solutions which together contribute the most HV are selected for evaluation on the expensive objective and constraint functions.

After the parallel evaluation of the solutions on the real functions (line 14, 15, 16 of algorithm 3), the RBF approximation errors (**E**) are stored for each RBF modeling strategy (line 17 of algorithm 3), the RBFs are updated (line 10 of algorithm 3), the best RBF modeling strategy is selected based on the historic approximation errors (line 11 of algorithm 3) and COBYLA is used again to find the next set of optimal solutions (line 12 of algorithm 3). This optimization process continues until the expensive evaluation budget is exhausted (line 9 of algorithm 3).

Acquisition Function Switching

IOC-SAMO-COBRA maximizes the predicted HV contribution every iteration, meaning that by default it does not use any uncertainty quantification of the RBF models for the objectives. Just like the RBF functions, by default, the inexpensive objectives also do not have an uncertainty quantification method. Other Bayesian optimization algorithms, however, often use Kriging or Gaussian process regression models, which provide an uncertainty quantification method for the objectives to encourage exploration [154, 118, 85]. Earlier experiments from Section 5.1.3, showed that the use of uncertainty quantification is in many cases redundant because by maximizing the HV, the algorithm is naturally forced to explore the objective space [148]. If, however, IOC-SAMO-COBRA gets stuck and does not find any HV improvement for three consecutive iterations, an uncertainty quantification method for RBFs (see Section 2.3.2 and Equation 2.5 or [12]) is enabled to help with exploration (this is part of line 17 of algorithm 3, but for space reasons not explicitly formulated in the pseudocode). By enabling the uncertainty quantification method, the acquisition function changes to an RBF variant of the S-Metric selection criterion [118]. Note that the inexpensive objectives still do not have an uncertainty quantification and therefore, only for the objectives modeled with RBFs the uncertainty is calculated.

5.3.3 IOC-SAMO-COBRA Experiments

The four algorithms (SA-NSGA-II, IC-SA-NSGA-II, SAMO-COBRA, IOC-SAMO-COBRA) are compared on the complete set of diverse test functions from Table 2.1. The surrogate-assisted algorithm and the Inexpensive function exploiting counterparts are compared to confirm our hypothesis that exploiting inexpensive functions in the optimization process directly is beneficial. The metrics used to compare the algorithms' performances are the HV and the IGD+ performance metrics which are described in more detail in Section 2.5.1.

Experimental Settings

The allowed number of function evaluations for the different algorithms is set to $40 \cdot d$ for all functions experimented with. The performances of the algorithms are checked with a different number of candidate solutions per iteration (In the SA-NSGA-II variants also referred to as population sizes) $p \in \{1, 2, 3, 4, 5, 6, 10, 20\}$. To get statistically significant results on all test functions, each test function is optimized 10 times per algorithm configuration.

All benchmark test functions are inexpensive to evaluate. However, SA-NSGA-II and SAMO-COBRA are developed to optimize computationally expensive problems. To test this functionality, in the experiments done with SA-NSGA-II and SAMO-COBRA all constraints and objectives are assumed to be expensive and are therefore modeled with the RBF surrogates. To test the functionality where inexpensive functions are directly used instead of a surrogate with IC-SA-NSGA-II and IOC-SAMO-COBRA, a decision needs to be made concerning the expensiveness of the objective and constraint functions. To be able to compare IOC-SAMO-COBRA to IC-SA-NSGA-II as fairly as possible, the assumption from IC-SA-NSGA-II that the constraints are inexpensive and the objectives are expensive to evaluate is also adopted in the experiments with IOC-SAMO-COBRA. A description and implementation of the test functions, the obtained Pareto frontiers for the IGD+ performance metric, all raw experiment results, and implementation of the IOC-SAMO-COBRA algorithm can be found on a dedicated Github page [145].

5.3.4 Results

The results obtained from the four algorithm variants are presented in tables, empirical cumulative distribution function plots, and empirical attainment function difference plots. Special attention is given to the problems with a very small feasibility ratio since these test problems benefit the most from using the inexpensive constraint functions directly in the optimization algorithms.

Performance Metrics Results

The two performance metrics used to assess and compare the performance of the different algorithms are the IGD+ metric and the HV metric. Table 5.9 and Table 5.10, respectively, report the mean and standard deviation of the HV and the IGD+ performance metric after $40 \cdot d$ function evaluations. The HV is computed between the Nadir point and the obtained Pareto fronts, the IGD+ metric is computed between

a well-spread Pareto front approximation and the obtained Pareto fronts by the different algorithms. The performance metrics for the SA-NSGA-II, IC-SA-NSGA-II, and SAMO-COBRA are statistically compared with a Wilcoxon rank sum test to the results of IOC-SAMO-COBRA at a 5% confidence level. A $(-)$ in the tables indicates significantly worse results, $(\approx)$ indicates indifference between the results, and $(+)$ indicates significantly better results of the given algorithm, compared to IOC-SAMO-COBRA. In the second last row of Table 5.9 and Table 5.10 a summary is given of the results of the significance test. Inspection of this summary shows that IOC-SAMO-COBRA in most cases achieves the best or statistically indistinguishable results after the number of function evaluations is exhausted for both the HV and IGD+ metric. On 14 out of 22 test problems, IOC-SAMO-COBRA outperforms the other algorithms when the performance is aggregated on data for all values of p that were tested. On 4 out of 22 test problems, SAMO-COBRA achieves a larger HV compared to IOC-SAMO-COBRA, however, these results are often not significant and differences are too small to be captured in the table with only two numbers after the decimal point. On the remaining 4 out of 22 test problems, the IC-SA-NSGA-II algorithm performs better compared to IOC-SAMO-COBRA, especially on BICOP1 and MW2. The mean Friedman rank test confirmed (with $p = 1 \cdot 10^{-16}$) the alternative hypothesis which states that there is a significant difference in the mean ranks of the algorithms. In the last rows of Table 5.9 and Table 5.10, respectively, the mean ranks of the algorithms are reported (a low rank indicates a better rank for both performance metrics).

Empirical Cumulative Distribution Function Results

Table 5.9 and Table 5.10 do not tell us anything about the convergence rate or how fast the different algorithms are able to find Pareto efficient solutions. Empirical Cumulative Distribution Functions (ECDF) from Section 2.5.2 visualize the convergence of the different algorithms. The aggregated results of the HV and IGD+ metric of the four different algorithm variants are visualized in Figure 5.7 and figure 5.8 by means of their ECDF, based on a fixed-target perspective. For each algorithm, the corresponding ECDF curve is aggregated over all functions and the number of candidate solutions per iteration. The four curves illustrate the advantage of the *Inexpensive Constraint* approach, independently of the base algorithm. This finding highlights the importance of using as accurate as possible models (by IOC-SAMO-COBRA's approach to evaluate and compare all RBF configurations in the configuration space Φ) and shows the relevance of using the constraint and objective functions directly if

Table 5.9: Hypervolume score ± standard deviation of hypervolume, Wilcoxon rank sum test with probability value = 0.05 (reference algorithm: IOC-SAMO-COBRA), per test function and candidate solutions size p. The highest HV per row is reported in **bold**, best scoring algorithm per test function is highlighted.

Function	p	SA-NSGA-II	IC-SA-NSGA-II	SAMO-COBRA	IOC-SAMO-COBRA
BNH	1	$4.89 \cdot 10^{3} \pm 3.06 \cdot 10^{1}$ (−)	$4.85 \cdot 10^{3} \pm 3.85 \cdot 10^{1}$ (−)	$\mathbf{5.07 \cdot 10^{3} \pm 4.10 \cdot 10^{-2}}$ (≈)	$5.07 \cdot 10^{3} \pm 2.44 \cdot 10^{-2}$
	2	$4.86 \cdot 10^{3} \pm 1.54 \cdot 10^{1}$ (−)	$4.83 \cdot 10^{3} \pm 3.36 \cdot 10^{1}$ (−)	$5.07 \cdot 10^{3} \pm 3.47 \cdot 10^{-2}$ (≈)	$\mathbf{5.07 \cdot 10^{3} \pm 3.85 \cdot 10^{-2}}$
	3	$4.88 \cdot 10^{3} \pm 3.12 \cdot 10^{1}$ (−)	$4.85 \cdot 10^{3} \pm 3.04 \cdot 10^{1}$ (−)	$\mathbf{5.07 \cdot 10^{3} \pm 5.44 \cdot 10^{-2}}$ (≈)	$5.07 \cdot 10^{3} \pm 2.86 \cdot 10^{-2}$
	4	$4.89 \cdot 10^{3} \pm 1.99 \cdot 10^{1}$ (−)	$4.84 \cdot 10^{3} \pm 2.61 \cdot 10^{1}$ (−)	$\mathbf{5.07 \cdot 10^{3} \pm 1.57 \cdot 10^{-1}}$ (+)	$5.07 \cdot 10^{3} \pm 9.29 \cdot 10^{-2}$
	5	$4.86 \cdot 10^{3} \pm 2.52 \cdot 10^{1}$ (−)	$4.85 \cdot 10^{3} \pm 2.46 \cdot 10^{1}$ (−)	$5.07 \cdot 10^{3} \pm 1.83 \cdot 10^{-1}$ (≈)	$\mathbf{5.07 \cdot 10^{3} \pm 2.24 \cdot 10^{-1}}$
	6	$4.88 \cdot 10^{3} \pm 1.37 \cdot 10^{1}$ (−)	$4.88 \cdot 10^{3} \pm 3.05 \cdot 10^{1}$ (−)	$5.07 \cdot 10^{3} \pm 1.25 \cdot 10^{-1}$ (≈)	$\mathbf{5.07 \cdot 10^{3} \pm 1.94 \cdot 10^{-1}}$
	10	$4.87 \cdot 10^{3} \pm 1.92 \cdot 10^{1}$ (−)	$4.86 \cdot 10^{3} \pm 2.68 \cdot 10^{1}$ (−)	$\mathbf{5.07 \cdot 10^{3} \pm 2.71 \cdot 10^{-1}}$ (≈)	$5.07 \cdot 10^{3} \pm 1.53 \cdot 10^{-1}$
	20	$4.90 \cdot 10^{3} \pm 3.03 \cdot 10^{1}$ (−)	$4.87 \cdot 10^{3} \pm 2.54 \cdot 10^{1}$ (−)	$\mathbf{5.06 \cdot 10^{3} \pm 4.25 \cdot 10^{-1}}$ (≈)	$5.06 \cdot 10^{3} \pm 3.94 \cdot 10^{-1}$
CEXP	1	$3.65 \cdot 10^{0} \pm 2.23 \cdot 10^{-2}$ (−)	$3.64 \cdot 10^{0} \pm 6.21 \cdot 10^{-2}$ (−)	$3.80 \cdot 10^{0} \pm 4.36 \cdot 10^{-4}$ (−)	$\mathbf{3.80 \cdot 10^{0} \pm 8.37 \cdot 10^{-5}}$
	2	$3.58 \cdot 10^{0} \pm 4.48 \cdot 10^{-2}$ (−)	$3.57 \cdot 10^{0} \pm 5.72 \cdot 10^{-2}$ (−)	$3.80 \cdot 10^{0} \pm 1.16 \cdot 10^{-3}$ (−)	$\mathbf{3.80 \cdot 10^{0} \pm 3.89 \cdot 10^{-4}}$
	3	$3.58 \cdot 10^{0} \pm 3.02 \cdot 10^{-2}$ (−)	$3.57 \cdot 10^{0} \pm 3.76 \cdot 10^{-2}$ (−)	$3.80 \cdot 10^{0} \pm 1.73 \cdot 10^{-4}$ (≈)	$\mathbf{3.80 \cdot 10^{0} \pm 5.62 \cdot 10^{-5}}$
	4	$3.57 \cdot 10^{0} \pm 3.56 \cdot 10^{-2}$ (−)	$3.56 \cdot 10^{0} \pm 3.62 \cdot 10^{-2}$ (−)	$3.80 \cdot 10^{0} \pm 3.68 \cdot 10^{-5}$ (−)	$\mathbf{3.80 \cdot 10^{0} \pm 2.65 \cdot 10^{-4}}$
	5	$3.58 \cdot 10^{0} \pm 3.18 \cdot 10^{-2}$ (−)	$3.56 \cdot 10^{0} \pm 4.78 \cdot 10^{-2}$ (−)	$\mathbf{3.80 \cdot 10^{0} \pm 2.35 \cdot 10^{-4}}$ (≈)	$3.80 \cdot 10^{0} \pm 1.09 \cdot 10^{-4}$
	6	$3.58 \cdot 10^{0} \pm 2.40 \cdot 10^{-2}$ (−)	$3.58 \cdot 10^{0} \pm 3.35 \cdot 10^{-2}$ (−)	$3.80 \cdot 10^{0} \pm 2.67 \cdot 10^{-4}$ (≈)	$\mathbf{3.80 \cdot 10^{0} \pm 1.54 \cdot 10^{-4}}$
	10	$3.56 \cdot 10^{0} \pm 4.87 \cdot 10^{-2}$ (−)	$3.60 \cdot 10^{0} \pm 2.43 \cdot 10^{-2}$ (−)	$3.79 \cdot 10^{0} \pm 6.69 \cdot 10^{-4}$ (≈)	$\mathbf{3.79 \cdot 10^{0} \pm 7.53 \cdot 10^{-4}}$
	20	$3.55 \cdot 10^{0} \pm 2.89 \cdot 10^{-2}$ (−)	$3.59 \cdot 10^{0} \pm 3.41 \cdot 10^{-2}$ (−)	$\mathbf{3.77 \cdot 10^{0} \pm 3.43 \cdot 10^{-3}}$ (≈)	$3.77 \cdot 10^{0} \pm 4.06 \cdot 10^{-3}$
SRN	1	$2.38 \cdot 10^{4} \pm 1.14 \cdot 10^{2}$ (−)	$2.40 \cdot 10^{4} \pm 1.53 \cdot 10^{2}$ (−)	$2.50 \cdot 10^{4} \pm 3.86 \cdot 10^{0}$ (≈)	$\mathbf{2.50 \cdot 10^{4} \pm 2.28 \cdot 10^{0}}$
	2	$2.29 \cdot 10^{4} \pm 2.99 \cdot 10^{2}$ (−)	$2.34 \cdot 10^{4} \pm 1.97 \cdot 10^{2}$ (−)	$2.50 \cdot 10^{4} \pm 1.13 \cdot 10^{1}$ (−)	$\mathbf{2.50 \cdot 10^{4} \pm 3.96 \cdot 10^{0}}$
	3	$2.34 \cdot 10^{4} \pm 2.55 \cdot 10^{2}$ (−)	$2.35 \cdot 10^{4} \pm 2.62 \cdot 10^{2}$ (−)	$2.50 \cdot 10^{4} \pm 6.39 \cdot 10^{0}$ (−)	$\mathbf{2.50 \cdot 10^{4} \pm 2.56 \cdot 10^{0}}$
	4	$2.30 \cdot 10^{4} \pm 2.84 \cdot 10^{2}$ (−)	$2.33 \cdot 10^{4} \pm 2.10 \cdot 10^{2}$ (−)	$\mathbf{2.50 \cdot 10^{4} \pm 2.80 \cdot 10^{0}}$ (+)	$2.50 \cdot 10^{4} \pm 2.14 \cdot 10^{0}$
	5	$2.33 \cdot 10^{4} \pm 2.32 \cdot 10^{2}$ (−)	$2.35 \cdot 10^{4} \pm 2.89 \cdot 10^{2}$ (−)	$2.50 \cdot 10^{4} \pm 2.62 \cdot 10^{0}$ (≈)	$\mathbf{2.50 \cdot 10^{4} \pm 2.78 \cdot 10^{0}}$
	6	$2.33 \cdot 10^{4} \pm 1.64 \cdot 10^{2}$ (−)	$2.36 \cdot 10^{4} \pm 1.43 \cdot 10^{2}$ (−)	$2.50 \cdot 10^{4} \pm 7.42 \cdot 10^{0}$ (≈)	$\mathbf{2.50 \cdot 10^{4} \pm 3.92 \cdot 10^{0}}$
	10	$2.33 \cdot 10^{4} \pm 2.03 \cdot 10^{2}$ (−)	$2.37 \cdot 10^{4} \pm 2.46 \cdot 10^{2}$ (−)	$\mathbf{2.49 \cdot 10^{4} \pm 3.07 \cdot 10^{1}}$ (≈)	$2.49 \cdot 10^{4} \pm 2.99 \cdot 10^{1}$
	20	$2.31 \cdot 10^{4} \pm 3.95 \cdot 10^{2}$ (−)	$2.37 \cdot 10^{4} \pm 1.39 \cdot 10^{2}$ (−)	$\mathbf{2.48 \cdot 10^{4} \pm 1.18 \cdot 10^{1}}$ (≈)	$2.48 \cdot 10^{4} \pm 2.05 \cdot 10^{1}$
TNK	1	$2.05 \cdot 10^{-1} \pm 1.38 \cdot 10^{-2}$ (−)	$2.87 \cdot 10^{-1} \pm 3.37 \cdot 10^{-3}$ (−)	$2.96 \cdot 10^{-1} \pm 1.65 \cdot 10^{-3}$ (−)	$\mathbf{3.03 \cdot 10^{-1} \pm 5.49 \cdot 10^{-4}}$
	2	$2.31 \cdot 10^{-1} \pm 1.75 \cdot 10^{-2}$ (−)	$2.75 \cdot 10^{-1} \pm 5.24 \cdot 10^{-3}$ (−)	$2.96 \cdot 10^{-1} \pm 1.99 \cdot 10^{-3}$ (−)	$\mathbf{3.05 \cdot 10^{-1} \pm 4.94 \cdot 10^{-4}}$
	3	$2.49 \cdot 10^{-1} \pm 1.70 \cdot 10^{-2}$ (−)	$2.84 \cdot 10^{-1} \pm 3.80 \cdot 10^{-3}$ (−)	$2.95 \cdot 10^{-1} \pm 3.09 \cdot 10^{-3}$ (−)	$\mathbf{3.06 \cdot 10^{-1} \pm 2.40 \cdot 10^{-4}}$
	4	$2.47 \cdot 10^{-1} \pm 1.34 \cdot 10^{-2}$ (−)	$2.71 \cdot 10^{-1} \pm 8.31 \cdot 10^{-3}$ (−)	$2.97 \cdot 10^{-1} \pm 1.80 \cdot 10^{-3}$ (−)	$\mathbf{3.06 \cdot 10^{-1} \pm 2.68 \cdot 10^{-4}}$
	5	$2.48 \cdot 10^{-1} \pm 1.13 \cdot 10^{-2}$ (−)	$2.77 \cdot 10^{-1} \pm 7.15 \cdot 10^{-3}$ (−)	$2.95 \cdot 10^{-1} \pm 2.26 \cdot 10^{-3}$ (−)	$\mathbf{3.06 \cdot 10^{-1} \pm 1.34 \cdot 10^{-4}}$
	6	$2.48 \cdot 10^{-1} \pm 1.47 \cdot 10^{-2}$ (−)	$2.81 \cdot 10^{-1} \pm 2.97 \cdot 10^{-3}$ (−)	$2.94 \cdot 10^{-1} \pm 1.25 \cdot 10^{-3}$ (−)	$\mathbf{3.06 \cdot 10^{-1} \pm 2.19 \cdot 10^{-4}}$
	10	$2.35 \cdot 10^{-1} \pm 1.21 \cdot 10^{-2}$ (−)	$2.73 \cdot 10^{-1} \pm 5.91 \cdot 10^{-3}$ (−)	$2.93 \cdot 10^{-1} \pm 2.56 \cdot 10^{-3}$ (−)	$\mathbf{3.06 \cdot 10^{-1} \pm 1.49 \cdot 10^{-4}}$
	20	$2.16 \cdot 10^{-1} \pm 1.11 \cdot 10^{-2}$ (−)	$2.72 \cdot 10^{-1} \pm 7.76 \cdot 10^{-3}$ (−)	$2.83 \cdot 10^{-1} \pm 3.23 \cdot 10^{-3}$ (−)	$\mathbf{3.01 \cdot 10^{-1} \pm 8.46 \cdot 10^{-4}}$
CTP1	1	$2.86 \cdot 10^{-1} \pm 4.12 \cdot 10^{-3}$ (−)	$2.89 \cdot 10^{-1} \pm 2.64 \cdot 10^{-4}$ (−)	$\mathbf{3.02 \cdot 10^{-1} \pm 1.29 \cdot 10^{-4}}$ (≈)	$3.02 \cdot 10^{-1} \pm 2.34 \cdot 10^{-4}$
	2	$2.76 \cdot 10^{-1} \pm 2.99 \cdot 10^{-3}$ (−)	$2.74 \cdot 10^{-1} \pm 7.15 \cdot 10^{-3}$ (−)	$3.00 \cdot 10^{-1} \pm 1.63 \cdot 10^{-3}$ (≈)	$\mathbf{3.01 \cdot 10^{-1} \pm 1.45 \cdot 10^{-3}}$
	3	$2.78 \cdot 10^{-1} \pm 5.88 \cdot 10^{-3}$ (−)	$2.81 \cdot 10^{-1} \pm 6.28 \cdot 10^{-3}$ (−)	$3.02 \cdot 10^{-1} \pm 4.18 \cdot 10^{-4}$ (≈)	$\mathbf{3.02 \cdot 10^{-1} \pm 8.68 \cdot 10^{-4}}$
	4	$2.80 \cdot 10^{-1} \pm 2.48 \cdot 10^{-3}$ (−)	$2.77 \cdot 10^{-1} \pm 4.06 \cdot 10^{-3}$ (−)	$3.02 \cdot 10^{-1} \pm 4.10 \cdot 10^{-4}$ (≈)	$\mathbf{3.02 \cdot 10^{-1} \pm 3.57 \cdot 10^{-4}}$
	5	$2.74 \cdot 10^{-1} \pm 6.52 \cdot 10^{-3}$ (−)	$2.76 \cdot 10^{-1} \pm 4.81 \cdot 10^{-3}$ (−)	$3.02 \cdot 10^{-1} \pm 3.58 \cdot 10^{-4}$ (≈)	$\mathbf{3.02 \cdot 10^{-1} \pm 3.58 \cdot 10^{-4}}$
	6	$2.78 \cdot 10^{-1} \pm 4.94 \cdot 10^{-3}$ (−)	$2.79 \cdot 10^{-1} \pm 2.59 \cdot 10^{-3}$ (−)	$\mathbf{3.02 \cdot 10^{-1} \pm 3.14 \cdot 10^{-4}}$ (≈)	$3.02 \cdot 10^{-1} \pm 3.14 \cdot 10^{-4}$
	10	$2.76 \cdot 10^{-1} \pm 5.45 \cdot 10^{-3}$ (−)	$2.81 \cdot 10^{-1} \pm 3.46 \cdot 10^{-3}$ (−)	$\mathbf{3.01 \cdot 10^{-1} \pm 2.53 \cdot 10^{-4}}$ (≈)	$3.01 \cdot 10^{-1} \pm 2.82 \cdot 10^{-4}$
	20	$2.74 \cdot 10^{-1} \pm 4.44 \cdot 10^{-3}$ (−)	$2.81 \cdot 10^{-1} \pm 4.28 \cdot 10^{-3}$ (−)	$2.99 \cdot 10^{-1} \pm 8.59 \cdot 10^{-4}$ (≈)	$\mathbf{2.99 \cdot 10^{-1} \pm 1.05 \cdot 10^{-3}}$
C3DTLZ4	1	$1.54 \cdot 10^{0} \pm 9.41 \cdot 10^{-2}$ (−)	$1.23 \cdot 10^{0} \pm 2.01 \cdot 10^{-1}$ (−)	$1.44 \cdot 10^{0} \pm 5.31 \cdot 10^{-2}$ (−)	$\mathbf{1.74 \cdot 10^{0} \pm 5.19 \cdot 10^{-3}}$
	2	$1.54 \cdot 10^{0} \pm 9.22 \cdot 10^{-2}$ (−)	$1.54 \cdot 10^{0} \pm 1.07 \cdot 10^{-1}$ (−)	$1.27 \cdot 10^{0} \pm 5.91 \cdot 10^{-2}$ (−)	$\mathbf{1.75 \cdot 10^{0} \pm 8.81 \cdot 10^{-3}}$
	3	$1.64 \cdot 10^{0} \pm 2.19 \cdot 10^{-2}$ (−)	$1.65 \cdot 10^{0} \pm 3.30 \cdot 10^{-2}$ (−)	$1.40 \cdot 10^{0} \pm 5.46 \cdot 10^{-2}$ (−)	$\mathbf{1.76 \cdot 10^{0} \pm 1.01 \cdot 10^{-3}}$
	4	$1.66 \cdot 10^{0} \pm 1.50 \cdot 10^{-2}$ (−)	$1.69 \cdot 10^{0} \pm 1.12 \cdot 10^{-2}$ (−)	$1.39 \cdot 10^{0} \pm 3.50 \cdot 10^{-2}$ (−)	$\mathbf{1.77 \cdot 10^{0} \pm 1.37 \cdot 10^{-3}}$
	5	$1.66 \cdot 10^{0} \pm 2.01 \cdot 10^{-2}$ (−)	$1.69 \cdot 10^{0} \pm 1.20 \cdot 10^{-2}$ (−)	$1.43 \cdot 10^{0} \pm 4.12 \cdot 10^{-2}$ (−)	$\mathbf{1.77 \cdot 10^{0} \pm 8.47 \cdot 10^{-4}}$
	6	$1.67 \cdot 10^{0} \pm 1.57 \cdot 10^{-2}$ (−)	$1.71 \cdot 10^{0} \pm 7.88 \cdot 10^{-3}$ (−)	$1.44 \cdot 10^{0} \pm 5.84 \cdot 10^{-2}$ (−)	$\mathbf{1.77 \cdot 10^{0} \pm 5.92 \cdot 10^{-4}}$
	10	$1.66 \cdot 10^{0} \pm 1.84 \cdot 10^{-2}$ (−)	$1.72 \cdot 10^{0} \pm 4.84 \cdot 10^{-3}$ (−)	$1.46 \cdot 10^{0} \pm 6.98 \cdot 10^{-2}$ (−)	$\mathbf{1.77 \cdot 10^{0} \pm 1.42 \cdot 10^{-3}}$
	20	$1.64 \cdot 10^{0} \pm 2.17 \cdot 10^{-2}$ (−)	$1.72 \cdot 10^{0} \pm 3.89 \cdot 10^{-3}$ (−)	$1.52 \cdot 10^{0} \pm 3.39 \cdot 10^{-2}$ (−)	$\mathbf{1.76 \cdot 10^{0} \pm 1.46 \cdot 10^{-3}}$
OSY	1	$9.62 \cdot 10^{3} \pm 1.98 \cdot 10^{3}$ (−)	$1.13 \cdot 10^{4} \pm 4.75 \cdot 10^{2}$ (−)	$\mathbf{1.26 \cdot 10^{4} \pm 4.21 \cdot 10^{0}}$ (≈)	$1.26 \cdot 10^{4} \pm 2.78 \cdot 10^{0}$
	2	$1.18 \cdot 10^{4} \pm 3.45 \cdot 10^{2}$ (−)	$1.18 \cdot 10^{4} \pm 2.70 \cdot 10^{2}$ (−)	$1.26 \cdot 10^{4} \pm 3.34 \cdot 10^{0}$ (−)	$\mathbf{1.26 \cdot 10^{4} \pm 3.66 \cdot 10^{0}}$
	3	$1.21 \cdot 10^{4} \pm 2.35 \cdot 10^{2}$ (−)	$1.23 \cdot 10^{4} \pm 6.57 \cdot 10^{1}$ (−)	$1.26 \cdot 10^{4} \pm 3.02 \cdot 10^{0}$ (≈)	$\mathbf{1.26 \cdot 10^{4} \pm 2.63 \cdot 10^{0}}$
	4	$1.22 \cdot 10^{4} \pm 1.36 \cdot 10^{2}$ (−)	$1.23 \cdot 10^{4} \pm 8.03 \cdot 10^{1}$ (−)	$\mathbf{1.26 \cdot 10^{4} \pm 2.79 \cdot 10^{0}}$ (≈)	$1.26 \cdot 10^{4} \pm 5.11 \cdot 10^{0}$
	5	$1.23 \cdot 10^{4} \pm 6.76 \cdot 10^{1}$ (−)	$1.23 \cdot 10^{4} \pm 7.50 \cdot 10^{1}$ (−)	$\mathbf{1.26 \cdot 10^{4} \pm 4.56 \cdot 10^{0}}$ (≈)	$1.26 \cdot 10^{4} \pm 3.79 \cdot 10^{0}$
	6	$1.23 \cdot 10^{4} \pm 4.06 \cdot 10^{1}$ (−)	$1.24 \cdot 10^{4} \pm 4.16 \cdot 10^{1}$ (−)	$1.26 \cdot 10^{4} \pm 6.01 \cdot 10^{0}$ (≈)	$\mathbf{1.26 \cdot 10^{4} \pm 6.76 \cdot 10^{0}}$
	10	$1.24 \cdot 10^{4} \pm 1.06 \cdot 10^{2}$ (−)	$1.24 \cdot 10^{4} \pm 5.34 \cdot 10^{1}$ (−)	$1.24 \cdot 10^{4} \pm 3.24 \cdot 10^{1}$ (≈)	$\mathbf{1.24 \cdot 10^{4} \pm 2.87 \cdot 10^{1}}$
	20	$1.23 \cdot 10^{4} \pm 1.40 \cdot 10^{2}$ (+)	$\mathbf{1.23 \cdot 10^{4} \pm 1.92 \cdot 10^{2}}$ (+)	$1.13 \cdot 10^{4} \pm 3.43 \cdot 10^{2}$ (−)	$1.16 \cdot 10^{4} \pm 1.67 \cdot 10^{2}$
TBTD	1	$3.46 \cdot 10^{2} \pm 9.91 \cdot 10^{1}$ (−)	$3.92 \cdot 10^{2} \pm 4.59 \cdot 10^{1}$ (−)	$4.95 \cdot 10^{2} \pm 3.40 \cdot 10^{0}$ (≈)	$\mathbf{4.96 \cdot 10^{2} \pm 9.50 \cdot 10^{0}}$
	2	$4.00 \cdot 10^{2} \pm 3.40 \cdot 10^{1}$ (−)	$4.37 \cdot 10^{2} \pm 1.41 \cdot 10^{1}$ (−)	$4.88 \cdot 10^{2} \pm 6.06 \cdot 10^{0}$ (≈)	$\mathbf{4.89 \cdot 10^{2} \pm 8.70 \cdot 10^{0}}$
	3	$4.18 \cdot 10^{2} \pm 1.40 \cdot 10^{1}$ (−)	$4.44 \cdot 10^{2} \pm 1.56 \cdot 10^{1}$ (−)	$4.73 \cdot 10^{2} \pm 9.80 \cdot 10^{0}$ (−)	$\mathbf{4.90 \cdot 10^{2} \pm 6.64 \cdot 10^{0}}$
	4	$4.17 \cdot 10^{2} \pm 1.91 \cdot 10^{1}$ (−)	$4.42 \cdot 10^{2} \pm 2.26 \cdot 10^{1}$ (−)	$4.70 \cdot 10^{2} \pm 9.65 \cdot 10^{0}$ (−)	$\mathbf{4.86 \cdot 10^{2} \pm 8.77 \cdot 10^{0}}$
	5	$4.16 \cdot 10^{2} \pm 1.47 \cdot 10^{1}$ (−)	$4.38 \cdot 10^{2} \pm 1.54 \cdot 10^{1}$ (−)	$4.77 \cdot 10^{2} \pm 7.71 \cdot 10^{0}$ (−)	$\mathbf{4.86 \cdot 10^{2} \pm 5.72 \cdot 10^{0}}$
	6	$4.25 \cdot 10^{2} \pm 1.80 \cdot 10^{1}$ (−)	$4.43 \cdot 10^{2} \pm 1.44 \cdot 10^{1}$ (−)	$4.72 \cdot 10^{2} \pm 1.09 \cdot 10^{1}$ (≈)	$\mathbf{4.76 \cdot 10^{2} \pm 1.03 \cdot 10^{1}}$
	10	$4.15 \cdot 10^{2} \pm 2.92 \cdot 10^{1}$ (−)	$4.46 \cdot 10^{2} \pm 1.34 \cdot 10^{1}$ (−)	$4.71 \cdot 10^{2} \pm 6.75 \cdot 10^{0}$ (≈)	$\mathbf{4.76 \cdot 10^{2} \pm 5.01 \cdot 10^{0}}$
	20	$4.26 \cdot 10^{2} \pm 1.70 \cdot 10^{1}$ (−)	$4.50 \cdot 10^{2} \pm 1.19 \cdot 10^{1}$ (≈)	$\mathbf{4.68 \cdot 10^{2} \pm 4.84 \cdot 10^{0}}$ (≈)	$4.61 \cdot 10^{2} \pm 9.27 \cdot 10^{0}$
NBP	1	$7.71 \cdot 10^{5} \pm 4.45 \cdot 10^{3}$ (−)	$7.72 \cdot 10^{5} \pm 8.82 \cdot 10^{3}$ (−)	$7.98 \cdot 10^{5} \pm 4.53 \cdot 10^{2}$ (−)	$\mathbf{8.01 \cdot 10^{5} \pm 8.88 \cdot 10^{0}}$
	2	$7.62 \cdot 10^{5} \pm 7.06 \cdot 10^{3}$ (−)	$7.63 \cdot 10^{5} \pm 5.29 \cdot 10^{3}$ (−)	$7.99 \cdot 10^{5} \pm 8.82 \cdot 10^{2}$ (−)	$\mathbf{8.01 \cdot 10^{5} \pm 6.72 \cdot 10^{1}}$
	3	$7.67 \cdot 10^{5} \pm 6.99 \cdot 10^{3}$ (−)	$7.69 \cdot 10^{5} \pm 3.09 \cdot 10^{3}$ (−)	$7.99 \cdot 10^{5} \pm 3.30 \cdot 10^{2}$ (−)	$\mathbf{8.01 \cdot 10^{5} \pm 1.03 \cdot 10^{1}}$
	4	$7.56 \cdot 10^{5} \pm 7.57 \cdot 10^{3}$ (−)	$7.65 \cdot 10^{5} \pm 7.17 \cdot 10^{3}$ (−)	$7.98 \cdot 10^{5} \pm 5.00 \cdot 10^{2}$ (−)	$\mathbf{8.01 \cdot 10^{5} \pm 3.08 \cdot 10^{1}}$
	5	$7.63 \cdot 10^{5} \pm 5.21 \cdot 10^{3}$ (−)	$7.69 \cdot 10^{5} \pm 3.60 \cdot 10^{3}$ (−)	$7.97 \cdot 10^{5} \pm 8.66 \cdot 10^{2}$ (−)	$\mathbf{8.00 \cdot 10^{5} \pm 1.36 \cdot 10^{2}}$
	6	$7.66 \cdot 10^{5} \pm 4.83 \cdot 10^{3}$ (−)	$7.68 \cdot 10^{5} \pm 6.03 \cdot 10^{3}$ (−)	$7.98 \cdot 10^{5} \pm 6.20 \cdot 10^{2}$ (−)	$\mathbf{8.00 \cdot 10^{5} \pm 1.48 \cdot 10^{2}}$
	10	$7.66 \cdot 10^{5} \pm 5.68 \cdot 10^{3}$ (−)	$7.72 \cdot 10^{5} \pm 3.84 \cdot 10^{3}$ (−)	$7.96 \cdot 10^{5} \pm 1.01 \cdot 10^{3}$ (−)	$\mathbf{7.99 \cdot 10^{5} \pm 5.26 \cdot 10^{2}}$
	20	$7.61 \cdot 10^{5} \pm 6.10 \cdot 10^{3}$ (−)	$7.68 \cdot 10^{5} \pm 4.32 \cdot 10^{3}$ (−)	$7.78 \cdot 10^{5} \pm 6.92 \cdot 10^{3}$ (−)	$\mathbf{7.95 \cdot 10^{5} \pm 5.81 \cdot 10^{2}}$
DBD	1	$3.38 \cdot 10^{1} \pm 2.84 \cdot 10^{-1}$ (−)	$3.29 \cdot 10^{1} \pm 1.12 \cdot 10^{0}$ (−)	$\mathbf{3.46 \cdot 10^{1} \pm 2.66 \cdot 10^{-2}}$ (+)	$3.46 \cdot 10^{1} \pm 1.06 \cdot 10^{-1}$
	2	$3.37 \cdot 10^{1} \pm 2.18 \cdot 10^{-1}$ (−)	$3.32 \cdot 10^{1} \pm 3.25 \cdot 10^{-1}$ (−)	$\mathbf{3.46 \cdot 10^{1} \pm 2.00 \cdot 10^{-2}}$ (+)	$3.44 \cdot 10^{1} \pm 1.15 \cdot 10^{-1}$
	3	$3.40 \cdot 10^{1} \pm 1.14 \cdot 10^{-1}$ (−)	$3.32 \cdot 10^{1} \pm 4.90 \cdot 10^{-1}$ (−)	$3.46 \cdot 10^{1} \pm 2.35 \cdot 10^{-2}$ (−)	$\mathbf{3.47 \cdot 10^{1} \pm 2.46 \cdot 10^{-3}}$
	4	$3.37 \cdot 10^{1} \pm 2.15 \cdot 10^{-1}$ (−)	$3.34 \cdot 10^{1} \pm 2.14 \cdot 10^{-1}$ (−)	$3.45 \cdot 10^{1} \pm 1.17 \cdot 10^{-1}$ (−)	$\mathbf{3.47 \cdot 10^{1} \pm 4.22 \cdot 10^{-3}}$
	5	$3.37 \cdot 10^{1} \pm 3.16 \cdot 10^{-1}$ (−)	$3.34 \cdot 10^{1} \pm 2.82 \cdot 10^{-1}$ (−)	$3.46 \cdot 10^{1} \pm 8.13 \cdot 10^{-2}$ (≈)	$\mathbf{3.46 \cdot 10^{1} \pm 5.64 \cdot 10^{-3}}$
	6	$3.39 \cdot 10^{1} \pm 9.60 \cdot 10^{-2}$ (−)	$3.31 \cdot 10^{1} \pm 6.13 \cdot 10^{-1}$ (−)	$\mathbf{3.46 \cdot 10^{1} \pm 5.80 \cdot 10^{-2}}$ (≈)	$3.46 \cdot 10^{1} \pm 5.86 \cdot 10^{-2}$
	10	$3.38 \cdot 10^{1} \pm 1.96 \cdot 10^{-1}$ (−)	$3.34 \cdot 10^{1} \pm 3.22 \cdot 10^{-1}$ (−)	$3.45 \cdot 10^{1} \pm 4.46 \cdot 10^{-2}$ (≈)	$\mathbf{3.46 \cdot 10^{1} \pm 1.62 \cdot 10^{-2}}$
	20	$3.37 \cdot 10^{1} \pm 2.51 \cdot 10^{-1}$ (−)	$3.31 \cdot 10^{1} \pm 3.73 \cdot 10^{-1}$ (−)	$\mathbf{3.45 \cdot 10^{1} \pm 1.72 \cdot 10^{-2}}$ (+)	$3.44 \cdot 10^{1} \pm 2.75 \cdot 10^{-2}$
SRD	1	$3.04 \cdot 10^{6} \pm 1.91 \cdot 10^{4}$ (≈)	$2.96 \cdot 10^{6} \pm 5.49 \cdot 10^{4}$ (−)	$\mathbf{3.07 \cdot 10^{6} \pm 5.91 \cdot 10^{2}}$ (+)	$3.06 \cdot 10^{6} \pm 6.25 \cdot 10^{3}$
	2	$3.06 \cdot 10^{6} \pm 8.30 \cdot 10^{3}$ (≈)	$3.06 \cdot 10^{6} \pm 4.81 \cdot 10^{3}$ (−)	$\mathbf{3.06 \cdot 10^{6} \pm 7.83 \cdot 10^{2}}$ (≈)	$3.06 \cdot 10^{6} \pm 1.02 \cdot 10^{3}$
	3	$3.06 \cdot 10^{6} \pm 2.73 \cdot 10^{3}$ (+)	$\mathbf{3.07 \cdot 10^{6} \pm 1.45 \cdot 10^{3}}$ (+)	$3.06 \cdot 10^{6} \pm 8.53 \cdot 10^{2}$ (−)	$3.06 \cdot 10^{6} \pm 7.05 \cdot 10^{2}$
	4	$3.06 \cdot 10^{6} \pm 9.27 \cdot 10^{2}$ (+)	$\mathbf{3.06 \cdot 10^{6} \pm 9.80 \cdot 10^{2}}$ (+)	$3.06 \cdot 10^{6} \pm 1.90 \cdot 10^{3}$ (−)	$3.06 \cdot 10^{6} \pm 8.32 \cdot 10^{2}$
	5	$3.06 \cdot 10^{6} \pm 9.63 \cdot 10^{2}$ (+)	$\mathbf{3.07 \cdot 10^{6} \pm 5.04 \cdot 10^{2}}$ (+)	$3.06 \cdot 10^{6} \pm 2.08 \cdot 10^{3}$ (−)	$3.06 \cdot 10^{6} \pm 6.44 \cdot 10^{2}$
	6	$3.06 \cdot 10^{6} \pm 1.18 \cdot 10^{3}$ (+)	$\mathbf{3.07 \cdot 10^{6} \pm 6.02 \cdot 10^{2}}$ (+)	$3.06 \cdot 10^{6} \pm 1.90 \cdot 10^{3}$ (−)	$3.06 \cdot 10^{6} \pm 7.64 \cdot 10^{2}$
	10	$3.06 \cdot 10^{6} \pm 8.15 \cdot 10^{2}$ (+)	$\mathbf{3.07 \cdot 10^{6} \pm 5.43 \cdot 10^{2}}$ (+)	$3.05 \cdot 10^{6} \pm 1.71 \cdot 10^{3}$ (−)	$3.06 \cdot 10^{6} \pm 1.25 \cdot 10^{3}$
	20	$3.06 \cdot 10^{6} \pm 9.27 \cdot 10^{2}$ (+)	$\mathbf{3.06 \cdot 10^{6} \pm 7.54 \cdot 10^{2}}$ (+)	$3.04 \cdot 10^{6} \pm 2.83 \cdot 10^{3}$ (−)	$3.05 \cdot 10^{6} \pm 1.61 \cdot 10^{3}$

Table continues on next page.

5.3. Expensive and Inexpensive Function Optimization

Continuation of Table 5.9.

Function	p	SA-NSGA-II	IC-SA-NSGA-II	SAMO-COBRA	IOC-SAMO-COBRA
WB	1	$2.46 \cdot 10^{-1} \pm 5.47 \cdot 10^{-2}$ (−)	$\mathbf{4.19 \cdot 10^{-1} \pm 2.11 \cdot 10^{-2}}$ (+)	$3.77 \cdot 10^{-1} \pm 1.01 \cdot 10^{-2}$ (−)	$4.15 \cdot 10^{-1} \pm 1.43 \cdot 10^{-3}$
	2	$3.46 \cdot 10^{-1} \pm 4.48 \cdot 10^{-2}$ (−)	$\mathbf{4.20 \cdot 10^{-1} \pm 3.02 \cdot 10^{-3}}$ (≈)	$3.87 \cdot 10^{-1} \pm 1.70 \cdot 10^{-2}$ (−)	$4.15 \cdot 10^{-1} \pm 8.41 \cdot 10^{-3}$
	3	$3.73 \cdot 10^{-1} \pm 3.96 \cdot 10^{-2}$ (−)	$\mathbf{4.23 \cdot 10^{-1} \pm 3.73 \cdot 10^{-3}}$ (+)	$4.06 \cdot 10^{-1} \pm 1.32 \cdot 10^{-2}$ (≈)	$4.14 \cdot 10^{-1} \pm 5.39 \cdot 10^{-3}$
	4	$3.96 \cdot 10^{-1} \pm 1.95 \cdot 10^{-2}$ (−)	$\mathbf{4.23 \cdot 10^{-1} \pm 1.86 \cdot 10^{-3}}$ (+)	$3.86 \cdot 10^{-1} \pm 1.39 \cdot 10^{-2}$ (−)	$4.11 \cdot 10^{-1} \pm 7.66 \cdot 10^{-3}$
	5	$3.72 \cdot 10^{-1} \pm 6.19 \cdot 10^{-2}$ (−)	$\mathbf{4.22 \cdot 10^{-1} \pm 2.55 \cdot 10^{-3}}$ (+)	$3.84 \cdot 10^{-1} \pm 2.23 \cdot 10^{-2}$ (−)	$4.14 \cdot 10^{-1} \pm 1.11 \cdot 10^{-2}$
	6	$3.83 \cdot 10^{-1} \pm 3.70 \cdot 10^{-2}$ (≈)	$\mathbf{4.24 \cdot 10^{-1} \pm 1.84 \cdot 10^{-3}}$ (+)	$3.79 \cdot 10^{-1} \pm 1.73 \cdot 10^{-2}$ (−)	$4.02 \cdot 10^{-1} \pm 1.64 \cdot 10^{-2}$
	10	$3.92 \cdot 10^{-1} \pm 9.35 \cdot 10^{-3}$ (≈)	$\mathbf{4.25 \cdot 10^{-1} \pm 2.64 \cdot 10^{-3}}$ (+)	$3.76 \cdot 10^{-1} \pm 1.69 \cdot 10^{-2}$ (−)	$3.96 \cdot 10^{-1} \pm 5.10 \cdot 10^{-3}$
	20	$3.67 \cdot 10^{-1} \pm 7.21 \cdot 10^{-2}$ (≈)	$\mathbf{4.24 \cdot 10^{-1} \pm 2.30 \cdot 10^{-3}}$ (+)	$3.72 \cdot 10^{-1} \pm 1.20 \cdot 10^{-2}$ (≈)	$3.78 \cdot 10^{-1} \pm 1.24 \cdot 10^{-2}$
BICOP1	1	$6.38 \cdot 10^{-2} \pm 9.98 \cdot 10^{-2}$ (≈)	$9.60 \cdot 10^{-2} \pm 1.05 \cdot 10^{-1}$ (≈)	$\mathbf{1.23 \cdot 10^{-1} \pm 1.62 \cdot 10^{-1}}$ (≈)	$7.91 \cdot 10^{-2} \pm 1.16 \cdot 10^{-1}$
	2	$5.98 \cdot 10^{-1} \pm 1.92 \cdot 10^{-2}$ (+)	$\mathbf{6.07 \cdot 10^{-1} \pm 1.34 \cdot 10^{-2}}$ (+)	$3.17 \cdot 10^{-1} \pm 2.60 \cdot 10^{-1}$ (≈)	$4.16 \cdot 10^{-1} \pm 2.11 \cdot 10^{-1}$
	3	$6.29 \cdot 10^{-1} \pm 1.03 \cdot 10^{-2}$ (≈)	$\mathbf{6.36 \cdot 10^{-1} \pm 4.84 \cdot 10^{-3}}$ (≈)	$5.06 \cdot 10^{-1} \pm 2.54 \cdot 10^{-1}$ (≈)	$5.79 \cdot 10^{-1} \pm 8.34 \cdot 10^{-2}$
	4	$6.41 \cdot 10^{-1} \pm 6.41 \cdot 10^{-3}$ (≈)	$\mathbf{6.43 \cdot 10^{-1} \pm 6.09 \cdot 10^{-3}}$ (≈)	$6.34 \cdot 10^{-1} \pm 1.06 \cdot 10^{-2}$ (≈)	$6.09 \cdot 10^{-1} \pm 7.65 \cdot 10^{-2}$
	5	$6.49 \cdot 10^{-1} \pm 4.38 \cdot 10^{-3}$ (+)	$\mathbf{6.50 \cdot 10^{-1} \pm 5.78 \cdot 10^{-3}}$ (+)	$6.25 \cdot 10^{-1} \pm 1.39 \cdot 10^{-2}$ (≈)	$6.20 \cdot 10^{-1} \pm 1.31 \cdot 10^{-2}$
	6	$6.53 \cdot 10^{-1} \pm 4.50 \cdot 10^{-3}$ (+)	$\mathbf{6.53 \cdot 10^{-1} \pm 3.46 \cdot 10^{-3}}$ (+)	$5.89 \cdot 10^{-1} \pm 1.88 \cdot 10^{-2}$ (≈)	$5.99 \cdot 10^{-1} \pm 1.37 \cdot 10^{-2}$
	10	$\mathbf{6.60 \cdot 10^{-1} \pm 1.08 \cdot 10^{-3}}$ (+)	$6.59 \cdot 10^{-1} \pm 1.88 \cdot 10^{-3}$ (+)	$4.91 \cdot 10^{-1} \pm 4.87 \cdot 10^{-2}$ (≈)	$5.08 \cdot 10^{-1} \pm 3.28 \cdot 10^{-2}$
	20	$6.60 \cdot 10^{-1} \pm 8.35 \cdot 10^{-4}$ (+)	$\mathbf{6.60 \cdot 10^{-1} \pm 7.77 \cdot 10^{-4}}$ (+)	$2.98 \cdot 10^{-1} \pm 9.13 \cdot 10^{-2}$ (≈)	$2.68 \cdot 10^{-1} \pm 7.79 \cdot 10^{-2}$
BICOP2	1	$1.04 \cdot 10^{-1} \pm 2.31 \cdot 10^{-2}$ (−)	$1.17 \cdot 10^{-1} \pm 2.88 \cdot 10^{-2}$ (−)	$2.16 \cdot 10^{-1} \pm 4.01 \cdot 10^{-2}$ (−)	$\mathbf{2.82 \cdot 10^{-1} \pm 1.79 \cdot 10^{-2}}$
	2	$1.06 \cdot 10^{-1} \pm 3.53 \cdot 10^{-2}$ (−)	$1.76 \cdot 10^{-1} \pm 3.44 \cdot 10^{-2}$ (−)	$2.15 \cdot 10^{-1} \pm 4.28 \cdot 10^{-2}$ (−)	$\mathbf{3.11 \cdot 10^{-1} \pm 3.03 \cdot 10^{-2}}$
	3	$1.22 \cdot 10^{-1} \pm 3.01 \cdot 10^{-2}$ (−)	$1.53 \cdot 10^{-1} \pm 4.97 \cdot 10^{-2}$ (−)	$2.23 \cdot 10^{-1} \pm 4.93 \cdot 10^{-2}$ (−)	$\mathbf{3.01 \cdot 10^{-1} \pm 5.25 \cdot 10^{-2}}$
	4	$1.21 \cdot 10^{-1} \pm 3.67 \cdot 10^{-2}$ (−)	$1.67 \cdot 10^{-1} \pm 5.22 \cdot 10^{-2}$ (≈)	$2.34 \cdot 10^{-1} \pm 5.56 \cdot 10^{-2}$ (≈)	$\mathbf{2.50 \cdot 10^{-1} \pm 7.37 \cdot 10^{-2}}$
	5	$1.27 \cdot 10^{-1} \pm 4.19 \cdot 10^{-2}$ (−)	$1.77 \cdot 10^{-1} \pm 4.21 \cdot 10^{-2}$ (≈)	$\mathbf{2.53 \cdot 10^{-1} \pm 3.15 \cdot 10^{-2}}$ (≈)	$2.24 \cdot 10^{-1} \pm 6.76 \cdot 10^{-2}$
	6	$1.26 \cdot 10^{-1} \pm 3.79 \cdot 10^{-2}$ (−)	$1.55 \cdot 10^{-1} \pm 4.65 \cdot 10^{-2}$ (≈)	$\mathbf{2.65 \cdot 10^{-1} \pm 1.68 \cdot 10^{-2}}$ (≈)	$2.13 \cdot 10^{-1} \pm 6.50 \cdot 10^{-2}$
	10	$1.53 \cdot 10^{-1} \pm 3.98 \cdot 10^{-2}$ (−)	$1.45 \cdot 10^{-1} \pm 3.91 \cdot 10^{-2}$ (−)	$2.38 \cdot 10^{-1} \pm 2.77 \cdot 10^{-2}$ (≈)	$\mathbf{2.44 \cdot 10^{-1} \pm 4.47 \cdot 10^{-2}}$
	20	$1.54 \cdot 10^{-1} \pm 4.41 \cdot 10^{-2}$ (−)	$1.50 \cdot 10^{-1} \pm 4.22 \cdot 10^{-2}$ (−)	$2.25 \cdot 10^{-1} \pm 2.07 \cdot 10^{-2}$ (−)	$\mathbf{2.72 \cdot 10^{-1} \pm 1.60 \cdot 10^{-2}}$
MW1	1	$0.00 \cdot 10^{0} \pm 0.00 \cdot 10^{0}$ (−)	$2.73 \cdot 10^{-1} \pm 4.54 \cdot 10^{-2}$ (−)	$1.66 \cdot 10^{-2} \pm 3.27 \cdot 10^{-2}$ (−)	$\mathbf{3.99 \cdot 10^{-1} \pm 5.85 \cdot 10^{-5}}$
	2	$2.40 \cdot 10^{-1} \pm 6.35 \cdot 10^{-2}$ (−)	$3.37 \cdot 10^{-1} \pm 5.49 \cdot 10^{-3}$ (−)	$1.92 \cdot 10^{-1} \pm 1.13 \cdot 10^{-1}$ (−)	$\mathbf{3.98 \cdot 10^{-1} \pm 8.02 \cdot 10^{-5}}$
	3	$2.82 \cdot 10^{-1} \pm 4.11 \cdot 10^{-2}$ (−)	$3.40 \cdot 10^{-1} \pm 1.00 \cdot 10^{-2}$ (−)	$3.10 \cdot 10^{-1} \pm 7.11 \cdot 10^{-2}$ (−)	$\mathbf{3.98 \cdot 10^{-1} \pm 1.55 \cdot 10^{-4}}$
	4	$3.24 \cdot 10^{-1} \pm 3.86 \cdot 10^{-2}$ (−)	$3.51 \cdot 10^{-1} \pm 1.15 \cdot 10^{-2}$ (−)	$2.20 \cdot 10^{-1} \pm 1.61 \cdot 10^{-1}$ (−)	$\mathbf{3.98 \cdot 10^{-1} \pm 1.56 \cdot 10^{-4}}$
	5	$3.49 \cdot 10^{-1} \pm 1.34 \cdot 10^{-2}$ (−)	$3.66 \cdot 10^{-1} \pm 6.28 \cdot 10^{-3}$ (−)	$2.09 \cdot 10^{-1} \pm 1.73 \cdot 10^{-1}$ (−)	$\mathbf{3.98 \cdot 10^{-1} \pm 1.70 \cdot 10^{-4}}$
	6	$3.56 \cdot 10^{-1} \pm 1.61 \cdot 10^{-2}$ (−)	$3.80 \cdot 10^{-1} \pm 5.43 \cdot 10^{-3}$ (−)	$2.55 \cdot 10^{-1} \pm 1.25 \cdot 10^{-1}$ (−)	$\mathbf{3.98 \cdot 10^{-1} \pm 5.34 \cdot 10^{-4}}$
	10	$3.56 \cdot 10^{-1} \pm 2.92 \cdot 10^{-2}$ (−)	$3.90 \cdot 10^{-1} \pm 1.81 \cdot 10^{-3}$ (−)	$1.63 \cdot 10^{-1} \pm 1.23 \cdot 10^{-1}$ (−)	$\mathbf{3.97 \cdot 10^{-1} \pm 1.10 \cdot 10^{-3}}$
	20	$3.72 \cdot 10^{-1} \pm 1.15 \cdot 10^{-2}$ (≈)	$\mathbf{3.93 \cdot 10^{-1} \pm 1.16 \cdot 10^{-3}}$ (+)	$2.09 \cdot 10^{-1} \pm 1.15 \cdot 10^{-1}$ (≈)	$2.73 \cdot 10^{-1} \pm 1.46 \cdot 10^{-1}$
MW2	1	$2.86 \cdot 10^{-2} \pm 5.73 \cdot 10^{-2}$ (−)	$\mathbf{4.24 \cdot 10^{-1} \pm 1.51 \cdot 10^{-2}}$ (+)	$1.60 \cdot 10^{-1} \pm 6.38 \cdot 10^{-2}$ (−)	$3.85 \cdot 10^{-1} \pm 2.32 \cdot 10^{-2}$
	2	$2.63 \cdot 10^{-1} \pm 6.16 \cdot 10^{-2}$ (−)	$\mathbf{4.33 \cdot 10^{-1} \pm 6.25 \cdot 10^{-3}}$ (≈)	$1.82 \cdot 10^{-1} \pm 1.22 \cdot 10^{-1}$ (−)	$4.19 \cdot 10^{-1} \pm 2.06 \cdot 10^{-2}$
	3	$2.93 \cdot 10^{-1} \pm 8.71 \cdot 10^{-2}$ (−)	$\mathbf{4.41 \cdot 10^{-1} \pm 8.60 \cdot 10^{-3}}$ (≈)	$1.98 \cdot 10^{-1} \pm 1.12 \cdot 10^{-1}$ (−)	$4.00 \cdot 10^{-1} \pm 6.68 \cdot 10^{-2}$
	4	$3.42 \cdot 10^{-1} \pm 8.05 \cdot 10^{-2}$ (≈)	$\mathbf{4.40 \cdot 10^{-1} \pm 8.97 \cdot 10^{-3}}$ (+)	$1.66 \cdot 10^{-1} \pm 1.03 \cdot 10^{-1}$ (−)	$3.47 \cdot 10^{-1} \pm 7.45 \cdot 10^{-2}$
	5	$3.38 \cdot 10^{-1} \pm 7.90 \cdot 10^{-2}$ (−)	$\mathbf{4.42 \cdot 10^{-1} \pm 8.72 \cdot 10^{-3}}$ (+)	$1.35 \cdot 10^{-1} \pm 7.24 \cdot 10^{-2}$ (−)	$3.96 \cdot 10^{-1} \pm 5.05 \cdot 10^{-2}$
	6	$3.40 \cdot 10^{-1} \pm 7.84 \cdot 10^{-2}$ (−)	$\mathbf{4.42 \cdot 10^{-1} \pm 7.95 \cdot 10^{-3}}$ (+)	$1.43 \cdot 10^{-1} \pm 9.95 \cdot 10^{-2}$ (−)	$4.11 \cdot 10^{-1} \pm 2.91 \cdot 10^{-2}$
	10	$3.20 \cdot 10^{-1} \pm 1.07 \cdot 10^{-1}$ (≈)	$\mathbf{4.45 \cdot 10^{-1} \pm 1.08 \cdot 10^{-2}}$ (+)	$1.04 \cdot 10^{-1} \pm 8.28 \cdot 10^{-2}$ (−)	$3.78 \cdot 10^{-1} \pm 3.47 \cdot 10^{-2}$
	20	$3.33 \cdot 10^{-1} \pm 9.66 \cdot 10^{-2}$ (≈)	$\mathbf{4.49 \cdot 10^{-1} \pm 9.99 \cdot 10^{-3}}$ (+)	$1.31 \cdot 10^{-1} \pm 1.09 \cdot 10^{-1}$ (−)	$3.10 \cdot 10^{-1} \pm 4.45 \cdot 10^{-2}$
MW3	1	$1.04 \cdot 10^{-1} \pm 1.48 \cdot 10^{-1}$ (−)	$4.10 \cdot 10^{-1} \pm 9.41 \cdot 10^{-3}$ (−)	$3.72 \cdot 10^{-1} \pm 2.46 \cdot 10^{-2}$ (−)	$\mathbf{4.50 \cdot 10^{-1} \pm 9.80 \cdot 10^{-4}}$
	2	$4.06 \cdot 10^{-1} \pm 1.30 \cdot 10^{-2}$ (−)	$4.22 \cdot 10^{-1} \pm 3.38 \cdot 10^{-3}$ (−)	$4.07 \cdot 10^{-1} \pm 8.56 \cdot 10^{-3}$ (−)	$\mathbf{4.51 \cdot 10^{-1} \pm 1.70 \cdot 10^{-3}}$
	3	$4.22 \cdot 10^{-1} \pm 6.32 \cdot 10^{-3}$ (−)	$4.29 \cdot 10^{-1} \pm 3.08 \cdot 10^{-3}$ (−)	$4.29 \cdot 10^{-1} \pm 1.08 \cdot 10^{-2}$ (−)	$\mathbf{4.52 \cdot 10^{-1} \pm 4.92 \cdot 10^{-4}}$
	4	$4.22 \cdot 10^{-1} \pm 2.48 \cdot 10^{-3}$ (−)	$4.32 \cdot 10^{-1} \pm 3.15 \cdot 10^{-3}$ (−)	$4.43 \cdot 10^{-1} \pm 5.59 \cdot 10^{-3}$ (−)	$\mathbf{4.52 \cdot 10^{-1} \pm 2.32 \cdot 10^{-4}}$
	5	$4.26 \cdot 10^{-1} \pm 4.97 \cdot 10^{-3}$ (−)	$4.36 \cdot 10^{-1} \pm 2.39 \cdot 10^{-3}$ (−)	$4.43 \cdot 10^{-1} \pm 3.76 \cdot 10^{-3}$ (−)	$\mathbf{4.51 \cdot 10^{-1} \pm 1.01 \cdot 10^{-3}}$
	6	$4.25 \cdot 10^{-1} \pm 2.38 \cdot 10^{-3}$ (−)	$4.37 \cdot 10^{-1} \pm 3.88 \cdot 10^{-3}$ (−)	$4.42 \cdot 10^{-1} \pm 3.55 \cdot 10^{-3}$ (−)	$\mathbf{4.50 \cdot 10^{-1} \pm 7.45 \cdot 10^{-4}}$
	10	$4.29 \cdot 10^{-1} \pm 3.28 \cdot 10^{-3}$ (−)	$4.41 \cdot 10^{-1} \pm 1.21 \cdot 10^{-3}$ (−)	$4.36 \cdot 10^{-1} \pm 1.97 \cdot 10^{-3}$ (−)	$\mathbf{4.48 \cdot 10^{-1} \pm 6.46 \cdot 10^{-4}}$
	20	$4.28 \cdot 10^{-1} \pm 4.92 \cdot 10^{-3}$ (−)	$4.40 \cdot 10^{-1} \pm 1.41 \cdot 10^{-3}$ (−)	$4.29 \cdot 10^{-1} \pm 2.55 \cdot 10^{-3}$ (−)	$\mathbf{4.44 \cdot 10^{-1} \pm 7.06 \cdot 10^{-4}}$
MW11	1	$6.65 \cdot 10^{-1} \pm 2.63 \cdot 10^{-1}$ (−)	$\mathbf{1.36 \cdot 10^{0} \pm 4.41 \cdot 10^{-2}}$ (+)	$9.80 \cdot 10^{-1} \pm 3.80 \cdot 10^{-1}$ (≈)	$1.10 \cdot 10^{0} \pm 1.99 \cdot 10^{-1}$
	2	$1.17 \cdot 10^{0} \pm 1.75 \cdot 10^{-1}$ (≈)	$\mathbf{1.42 \cdot 10^{0} \pm 2.43 \cdot 10^{-2}}$ (+)	$9.82 \cdot 10^{-1} \pm 1.74 \cdot 10^{-1}$ (−)	$1.17 \cdot 10^{0} \pm 1.55 \cdot 10^{-1}$
	3	$1.09 \cdot 10^{0} \pm 2.30 \cdot 10^{-1}$ (−)	$1.44 \cdot 10^{0} \pm 1.83 \cdot 10^{-2}$ (−)	$9.92 \cdot 10^{-1} \pm 1.97 \cdot 10^{-1}$ (−)	$\mathbf{1.49 \cdot 10^{0} \pm 4.23 \cdot 10^{-2}}$
	4	$1.03 \cdot 10^{0} \pm 2.44 \cdot 10^{-1}$ (−)	$1.46 \cdot 10^{0} \pm 1.81 \cdot 10^{-2}$ (−)	$9.99 \cdot 10^{-1} \pm 1.23 \cdot 10^{-1}$ (−)	$\mathbf{1.51 \cdot 10^{0} \pm 1.40 \cdot 10^{-2}}$
	5	$1.04 \cdot 10^{0} \pm 2.41 \cdot 10^{-1}$ (−)	$1.46 \cdot 10^{0} \pm 9.42 \cdot 10^{-3}$ (−)	$1.06 \cdot 10^{0} \pm 1.80 \cdot 10^{-1}$ (−)	$\mathbf{1.52 \cdot 10^{0} \pm 8.07 \cdot 10^{-3}}$
	6	$9.08 \cdot 10^{-1} \pm 1.57 \cdot 10^{-1}$ (−)	$1.48 \cdot 10^{0} \pm 8.50 \cdot 10^{-3}$ (−)	$9.75 \cdot 10^{-1} \pm 2.81 \cdot 10^{-1}$ (−)	$\mathbf{1.52 \cdot 10^{0} \pm 1.26 \cdot 10^{-2}}$
	10	$9.52 \cdot 10^{-1} \pm 2.21 \cdot 10^{-1}$ (−)	$1.49 \cdot 10^{0} \pm 8.09 \cdot 10^{-3}$ (−)	$8.27 \cdot 10^{-1} \pm 1.73 \cdot 10^{-1}$ (−)	$\mathbf{1.52 \cdot 10^{0} \pm 5.60 \cdot 10^{-3}}$
	20	$8.02 \cdot 10^{-1} \pm 1.80 \cdot 10^{-1}$ (−)	$1.49 \cdot 10^{0} \pm 1.53 \cdot 10^{-2}$ (−)	$8.78 \cdot 10^{-1} \pm 5.96 \cdot 10^{-2}$ (−)	$\mathbf{1.50 \cdot 10^{0} \pm 6.86 \cdot 10^{-3}}$
TRICOP	1	$4.47 \cdot 10^{1} \pm 2.03 \cdot 10^{0}$ (−)	$4.57 \cdot 10^{1} \pm 1.19 \cdot 10^{0}$ (−)	$\mathbf{4.97 \cdot 10^{1} \pm 6.30 \cdot 10^{-3}}$ (≈)	$4.97 \cdot 10^{1} \pm 3.81 \cdot 10^{-2}$
	2	$4.19 \cdot 10^{1} \pm 1.56 \cdot 10^{0}$ (−)	$4.55 \cdot 10^{1} \pm 7.37 \cdot 10^{-1}$ (−)	$4.96 \cdot 10^{1} \pm 2.76 \cdot 10^{-2}$ (≈)	$\mathbf{4.97 \cdot 10^{1} \pm 3.93 \cdot 10^{-2}}$
	3	$4.31 \cdot 10^{1} \pm 1.75 \cdot 10^{0}$ (−)	$4.63 \cdot 10^{1} \pm 5.46 \cdot 10^{-1}$ (−)	$\mathbf{4.97 \cdot 10^{1} \pm 1.93 \cdot 10^{-2}}$ (+)	$4.96 \cdot 10^{1} \pm 3.41 \cdot 10^{-2}$
	4	$4.31 \cdot 10^{1} \pm 1.50 \cdot 10^{0}$ (−)	$4.63 \cdot 10^{1} \pm 6.92 \cdot 10^{-1}$ (−)	$\mathbf{4.96 \cdot 10^{1} \pm 4.30 \cdot 10^{-2}}$ (≈)	$4.96 \cdot 10^{1} \pm 3.22 \cdot 10^{-2}$
	5	$4.26 \cdot 10^{1} \pm 1.45 \cdot 10^{0}$ (−)	$4.66 \cdot 10^{1} \pm 3.57 \cdot 10^{-1}$ (−)	$\mathbf{4.97 \cdot 10^{1} \pm 2.46 \cdot 10^{-2}}$ (≈)	$4.97 \cdot 10^{1} \pm 2.45 \cdot 10^{-2}$
	6	$4.36 \cdot 10^{1} \pm 1.48 \cdot 10^{0}$ (−)	$4.63 \cdot 10^{1} \pm 5.04 \cdot 10^{-1}$ (−)	$\mathbf{4.97 \cdot 10^{1} \pm 3.34 \cdot 10^{-2}}$ (≈)	$4.97 \cdot 10^{1} \pm 5.05 \cdot 10^{-2}$
	10	$4.40 \cdot 10^{1} \pm 1.43 \cdot 10^{0}$ (−)	$4.71 \cdot 10^{1} \pm 3.92 \cdot 10^{-1}$ (−)	$\mathbf{4.97 \cdot 10^{1} \pm 3.00 \cdot 10^{-2}}$ (≈)	$4.97 \cdot 10^{1} \pm 1.95 \cdot 10^{-2}$
	20	$4.55 \cdot 10^{1} \pm 8.60 \cdot 10^{-1}$ (−)	$4.76 \cdot 10^{1} \pm 3.49 \cdot 10^{-1}$ (−)	$4.95 \cdot 10^{1} \pm 4.39 \cdot 10^{-2}$ (≈)	$\mathbf{4.95 \cdot 10^{1} \pm 2.77 \cdot 10^{-2}}$
SPD	1	$5.04 \cdot 10^{9} \pm 8.11 \cdot 10^{7}$ (−)	$4.95 \cdot 10^{9} \pm 1.07 \cdot 10^{8}$ (−)	$5.87 \cdot 10^{9} \pm 1.68 \cdot 10^{7}$ (−)	$\mathbf{6.01 \cdot 10^{9} \pm 1.95 \cdot 10^{6}}$
	2	$4.88 \cdot 10^{9} \pm 1.58 \cdot 10^{8}$ (−)	$5.05 \cdot 10^{9} \pm 6.73 \cdot 10^{7}$ (−)	$5.91 \cdot 10^{9} \pm 2.36 \cdot 10^{7}$ (−)	$\mathbf{6.02 \cdot 10^{9} \pm 3.06 \cdot 10^{6}}$
	3	$5.02 \cdot 10^{9} \pm 7.48 \cdot 10^{7}$ (−)	$5.08 \cdot 10^{9} \pm 8.10 \cdot 10^{7}$ (−)	$5.93 \cdot 10^{9} \pm 9.35 \cdot 10^{6}$ (−)	$\mathbf{6.01 \cdot 10^{9} \pm 2.47 \cdot 10^{6}}$
	4	$4.97 \cdot 10^{9} \pm 1.19 \cdot 10^{8}$ (−)	$5.02 \cdot 10^{9} \pm 7.63 \cdot 10^{7}$ (−)	$5.93 \cdot 10^{9} \pm 6.67 \cdot 10^{6}$ (−)	$\mathbf{6.01 \cdot 10^{9} \pm 3.71 \cdot 10^{6}}$
	5	$5.06 \cdot 10^{9} \pm 8.50 \cdot 10^{7}$ (−)	$5.03 \cdot 10^{9} \pm 1.44 \cdot 10^{8}$ (−)	$5.91 \cdot 10^{9} \pm 1.68 \cdot 10^{7}$ (−)	$\mathbf{6.01 \cdot 10^{9} \pm 3.20 \cdot 10^{6}}$
	6	$5.07 \cdot 10^{9} \pm 5.48 \cdot 10^{7}$ (−)	$5.05 \cdot 10^{9} \pm 5.88 \cdot 10^{7}$ (−)	$5.91 \cdot 10^{9} \pm 1.17 \cdot 10^{7}$ (−)	$\mathbf{6.00 \cdot 10^{9} \pm 5.75 \cdot 10^{6}}$
	10	$5.08 \cdot 10^{9} \pm 6.05 \cdot 10^{7}$ (−)	$5.06 \cdot 10^{9} \pm 9.73 \cdot 10^{7}$ (−)	$5.86 \cdot 10^{9} \pm 2.25 \cdot 10^{7}$ (−)	$\mathbf{5.99 \cdot 10^{9} \pm 3.85 \cdot 10^{6}}$
	20	$5.08 \cdot 10^{9} \pm 1.15 \cdot 10^{8}$ (−)	$5.04 \cdot 10^{9} \pm 8.77 \cdot 10^{7}$ (−)	$5.80 \cdot 10^{9} \pm 2.77 \cdot 10^{7}$ (−)	$\mathbf{5.93 \cdot 10^{9} \pm 1.02 \cdot 10^{7}}$
CSI	1	$7.31 \cdot 10^{0} \pm 9.10 \cdot 10^{-2}$ (−)	$6.06 \cdot 10^{0} \pm 3.59 \cdot 10^{-1}$ (−)	$8.33 \cdot 10^{0} \pm 6.84 \cdot 10^{-2}$ (≈)	$\mathbf{8.35 \cdot 10^{0} \pm 9.34 \cdot 10^{-2}}$
	2	$7.11 \cdot 10^{0} \pm 1.34 \cdot 10^{-1}$ (−)	$7.04 \cdot 10^{0} \pm 7.90 \cdot 10^{-2}$ (−)	$8.45 \cdot 10^{0} \pm 1.70 \cdot 10^{-2}$ (≈)	$\mathbf{8.46 \cdot 10^{0} \pm 2.16 \cdot 10^{-2}}$
	3	$7.20 \cdot 10^{0} \pm 1.01 \cdot 10^{-1}$ (−)	$7.12 \cdot 10^{0} \pm 1.45 \cdot 10^{-1}$ (−)	$8.47 \cdot 10^{0} \pm 1.32 \cdot 10^{-2}$ (−)	$\mathbf{8.49 \cdot 10^{0} \pm 5.44 \cdot 10^{-3}}$
	4	$7.13 \cdot 10^{0} \pm 6.49 \cdot 10^{-2}$ (−)	$7.14 \cdot 10^{0} \pm 7.68 \cdot 10^{-2}$ (−)	$8.46 \cdot 10^{0} \pm 1.36 \cdot 10^{-2}$ (−)	$\mathbf{8.48 \cdot 10^{0} \pm 9.85 \cdot 10^{-3}}$
	5	$7.09 \cdot 10^{0} \pm 6.13 \cdot 10^{-2}$ (−)	$7.10 \cdot 10^{0} \pm 1.75 \cdot 10^{-1}$ (−)	$8.46 \cdot 10^{0} \pm 1.11 \cdot 10^{-2}$ (−)	$\mathbf{8.48 \cdot 10^{0} \pm 7.05 \cdot 10^{-3}}$
	6	$7.14 \cdot 10^{0} \pm 1.04 \cdot 10^{-1}$ (−)	$7.17 \cdot 10^{0} \pm 9.63 \cdot 10^{-2}$ (−)	$8.45 \cdot 10^{0} \pm 1.00 \cdot 10^{-2}$ (≈)	$\mathbf{8.46 \cdot 10^{0} \pm 1.21 \cdot 10^{-2}}$
	10	$7.21 \cdot 10^{0} \pm 1.05 \cdot 10^{-1}$ (−)	$7.19 \cdot 10^{0} \pm 6.00 \cdot 10^{-2}$ (−)	$8.33 \cdot 10^{0} \pm 1.89 \cdot 10^{-2}$ (−)	$\mathbf{8.35 \cdot 10^{0} \pm 1.28 \cdot 10^{-2}}$
	20	$7.10 \cdot 10^{0} \pm 1.68 \cdot 10^{-1}$ (−)	$7.22 \cdot 10^{0} \pm 1.10 \cdot 10^{-1}$ (−)	$\mathbf{7.90 \cdot 10^{0} \pm 8.04 \cdot 10^{-2}}$ (≈)	$7.86 \cdot 10^{0} \pm 4.02 \cdot 10^{-2}$
WP	1	$1.03 \cdot 10^{18} \pm 3.31 \cdot 10^{17}$ (−)	$1.00 \cdot 10^{18} \pm 2.70 \cdot 10^{17}$ (−)	$3.42 \cdot 10^{18} \pm 8.57 \cdot 10^{15}$ (≈)	$\mathbf{3.42 \cdot 10^{18} \pm 7.12 \cdot 10^{15}}$
	2	$1.47 \cdot 10^{18} \pm 2.17 \cdot 10^{17}$ (−)	$1.36 \cdot 10^{18} \pm 4.05 \cdot 10^{17}$ (−)	$\mathbf{3.44 \cdot 10^{18} \pm 1.62 \cdot 10^{15}}$ (+)	$3.42 \cdot 10^{18} \pm 5.17 \cdot 10^{15}$
	3	$1.61 \cdot 10^{18} \pm 2.10 \cdot 10^{17}$ (−)	$1.47 \cdot 10^{18} \pm 2.37 \cdot 10^{17}$ (−)	$3.42 \cdot 10^{18} \pm 1.04 \cdot 10^{16}$ (≈)	$\mathbf{3.42 \cdot 10^{18} \pm 8.90 \cdot 10^{15}}$
	4	$1.70 \cdot 10^{18} \pm 1.89 \cdot 10^{17}$ (−)	$1.53 \cdot 10^{18} \pm 2.55 \cdot 10^{17}$ (−)	$\mathbf{3.42 \cdot 10^{18} \pm 7.99 \cdot 10^{15}}$ (≈)	$3.42 \cdot 10^{18} \pm 5.15 \cdot 10^{15}$
	5	$1.69 \cdot 10^{18} \pm 2.27 \cdot 10^{17}$ (−)	$1.57 \cdot 10^{18} \pm 3.08 \cdot 10^{17}$ (−)	$\mathbf{3.42 \cdot 10^{18} \pm 6.03 \cdot 10^{15}}$ (+)	$3.41 \cdot 10^{18} \pm 1.46 \cdot 10^{16}$
	6	$1.86 \cdot 10^{18} \pm 1.69 \cdot 10^{17}$ (−)	$1.58 \cdot 10^{18} \pm 3.50 \cdot 10^{17}$ (−)	$\mathbf{3.42 \cdot 10^{18} \pm 5.69 \cdot 10^{15}}$ (+)	$3.40 \cdot 10^{18} \pm 1.47 \cdot 10^{16}$
	10	$1.97 \cdot 10^{18} \pm 2.78 \cdot 10^{17}$ (−)	$1.70 \cdot 10^{18} \pm 2.86 \cdot 10^{17}$ (−)	$3.37 \cdot 10^{18} \pm 1.51 \cdot 10^{16}$ (−)	$\mathbf{3.40 \cdot 10^{18} \pm 6.14 \cdot 10^{15}}$
	20	$1.90 \cdot 10^{18} \pm 2.51 \cdot 10^{17}$ (−)	$1.79 \cdot 10^{18} \pm 2.84 \cdot 10^{17}$ (−)	$\mathbf{3.37 \cdot 10^{18} \pm 7.01 \cdot 10^{15}}$ (≈)	$3.37 \cdot 10^{18} \pm 1.09 \cdot 10^{16}$
Wilcoxon summary		151−, 13 ≈, 12+	138−, 10 ≈, 28+	95−, 71 ≈, 10+	reference algorithm
Mean Friedman rank		3.35	2.79	2.30	**1.56**

Table 5.10: IGD+ score ± standard deviation of IGD+, Wilcoxon rank sum test with probability value = 0.05 (reference algorithm: IOC-SAMO-COBRA), per test function and candidate solutions size p. The lowest IGD+ per row is reported in **bold**, best scoring algorithm per test function is highlighted.

Function	p	SA-NSGA-II	IC-SA-NSGA-II	SAMO-COBRA	IOC-SAMO-COBRA
BNH	1	$1.77 \cdot 10^{-2} \pm 2.89 \cdot 10^{-3}$ (–)	$2.15 \cdot 10^{-2} \pm 3.60 \cdot 10^{-3}$ (–)	$2.06 \cdot 10^{-3} \pm 1.44 \cdot 10^{-5}$ (≈)	$\mathbf{2.06 \cdot 10^{-3} \pm 1.08 \cdot 10^{-5}}$
	2	$1.95 \cdot 10^{-2} \pm 1.39 \cdot 10^{-3}$ (–)	$2.21 \cdot 10^{-2} \pm 2.89 \cdot 10^{-3}$ (–)	$\mathbf{2.12 \cdot 10^{-3} \pm 1.14 \cdot 10^{-5}}$ (≈)	$2.12 \cdot 10^{-3} \pm 1.52 \cdot 10^{-5}$
	3	$1.81 \cdot 10^{-2} \pm 2.50 \cdot 10^{-3}$ (–)	$2.03 \cdot 10^{-2} \pm 2.92 \cdot 10^{-3}$ (–)	$2.13 \cdot 10^{-3} \pm 3.93 \cdot 10^{-5}$ (≈)	$\mathbf{2.12 \cdot 10^{-3} \pm 2.83 \cdot 10^{-5}}$
	4	$1.75 \cdot 10^{-2} \pm 1.61 \cdot 10^{-3}$ (–)	$2.11 \cdot 10^{-2} \pm 1.95 \cdot 10^{-3}$ (–)	$\mathbf{2.12 \cdot 10^{-3} \pm 4.18 \cdot 10^{-5}}$ (≈)	$2.15 \cdot 10^{-3} \pm 4.39 \cdot 10^{-5}$
	5	$1.94 \cdot 10^{-2} \pm 2.28 \cdot 10^{-3}$ (–)	$2.05 \cdot 10^{-2} \pm 2.19 \cdot 10^{-3}$ (–)	$2.14 \cdot 10^{-3} \pm 2.59 \cdot 10^{-5}$ (≈)	$\mathbf{2.13 \cdot 10^{-3} \pm 3.13 \cdot 10^{-5}}$
	6	$1.84 \cdot 10^{-2} \pm 1.03 \cdot 10^{-3}$ (–)	$1.79 \cdot 10^{-2} \pm 2.52 \cdot 10^{-3}$ (–)	$2.09 \cdot 10^{-3} \pm 5.02 \cdot 10^{-5}$ (≈)	$\mathbf{2.09 \cdot 10^{-3} \pm 3.86 \cdot 10^{-5}}$
	10	$1.85 \cdot 10^{-2} \pm 1.72 \cdot 10^{-3}$ (–)	$1.97 \cdot 10^{-2} \pm 2.28 \cdot 10^{-3}$ (–)	$\mathbf{2.40 \cdot 10^{-3} \pm 5.77 \cdot 10^{-5}}$ (≈)	$2.42 \cdot 10^{-3} \pm 5.79 \cdot 10^{-5}$
	20	$1.67 \cdot 10^{-2} \pm 2.40 \cdot 10^{-3}$ (–)	$1.89 \cdot 10^{-2} \pm 2.32 \cdot 10^{-3}$ (–)	$\mathbf{3.03 \cdot 10^{-3} \pm 5.58 \cdot 10^{-5}}$ (≈)	$3.06 \cdot 10^{-3} \pm 6.72 \cdot 10^{-5}$
CEXP	1	$1.79 \cdot 10^{-2} \pm 2.35 \cdot 10^{-3}$ (–)	$1.83 \cdot 10^{-2} \pm 6.60 \cdot 10^{-3}$ (–)	$2.54 \cdot 10^{-3} \pm 4.97 \cdot 10^{-5}$ (–)	$\mathbf{2.17 \cdot 10^{-3} \pm 1.17 \cdot 10^{-5}}$
	2	$2.50 \cdot 10^{-2} \pm 4.41 \cdot 10^{-3}$ (–)	$2.60 \cdot 10^{-2} \pm 6.07 \cdot 10^{-3}$ (–)	$2.43 \cdot 10^{-3} \pm 1.16 \cdot 10^{-4}$ (–)	$\mathbf{2.35 \cdot 10^{-3} \pm 4.84 \cdot 10^{-5}}$
	3	$2.50 \cdot 10^{-2} \pm 3.26 \cdot 10^{-3}$ (–)	$2.66 \cdot 10^{-2} \pm 3.94 \cdot 10^{-3}$ (–)	$2.17 \cdot 10^{-3} \pm 5.72 \cdot 10^{-5}$ (≈)	$\mathbf{2.15 \cdot 10^{-3} \pm 8.76 \cdot 10^{-6}}$
	4	$2.57 \cdot 10^{-2} \pm 3.90 \cdot 10^{-3}$ (–)	$2.69 \cdot 10^{-2} \pm 3.73 \cdot 10^{-3}$ (–)	$\mathbf{2.36 \cdot 10^{-3} \pm 1.49 \cdot 10^{-5}}$ (≈)	$2.38 \cdot 10^{-3} \pm 4.91 \cdot 10^{-5}$
	5	$2.51 \cdot 10^{-2} \pm 3.15 \cdot 10^{-3}$ (–)	$2.73 \cdot 10^{-2} \pm 5.23 \cdot 10^{-3}$ (–)	$\mathbf{2.45 \cdot 10^{-3} \pm 3.29 \cdot 10^{-5}}$ (≈)	$2.46 \cdot 10^{-3} \pm 3.09 \cdot 10^{-5}$
	6	$2.52 \cdot 10^{-2} \pm 2.47 \cdot 10^{-3}$ (–)	$2.46 \cdot 10^{-2} \pm 3.33 \cdot 10^{-3}$ (–)	$2.34 \cdot 10^{-3} \pm 5.45 \cdot 10^{-5}$ (≈)	$\mathbf{2.33 \cdot 10^{-3} \pm 4.39 \cdot 10^{-5}}$
	10	$2.67 \cdot 10^{-2} \pm 5.05 \cdot 10^{-3}$ (–)	$2.31 \cdot 10^{-2} \pm 2.66 \cdot 10^{-3}$ (–)	$2.88 \cdot 10^{-3} \pm 8.27 \cdot 10^{-5}$ (≈)	$\mathbf{2.84 \cdot 10^{-3} \pm 6.72 \cdot 10^{-5}}$
	20	$2.81 \cdot 10^{-2} \pm 3.01 \cdot 10^{-3}$ (–)	$2.39 \cdot 10^{-2} \pm 3.53 \cdot 10^{-3}$ (–)	$5.00 \cdot 10^{-3} \pm 3.61 \cdot 10^{-4}$ (≈)	$\mathbf{4.99 \cdot 10^{-3} \pm 4.43 \cdot 10^{-4}}$
SRN	1	$1.89 \cdot 10^{-2} \pm 1.70 \cdot 10^{-3}$ (–)	$1.54 \cdot 10^{-2} \pm 1.73 \cdot 10^{-3}$ (–)	$3.47 \cdot 10^{-3} \pm 4.37 \cdot 10^{-5}$ (–)	$\mathbf{3.39 \cdot 10^{-3} \pm 3.39 \cdot 10^{-5}}$
	2	$3.06 \cdot 10^{-2} \pm 6.16 \cdot 10^{-3}$ (–)	$2.23 \cdot 10^{-2} \pm 4.60 \cdot 10^{-3}$ (–)	$3.66 \cdot 10^{-3} \pm 1.52 \cdot 10^{-4}$ (–)	$\mathbf{3.32 \cdot 10^{-3} \pm 5.62 \cdot 10^{-5}}$
	3	$2.09 \cdot 10^{-2} \pm 2.54 \cdot 10^{-3}$ (–)	$2.02 \cdot 10^{-2} \pm 2.50 \cdot 10^{-3}$ (–)	$3.56 \cdot 10^{-3} \pm 5.90 \cdot 10^{-5}$ (–)	$\mathbf{3.31 \cdot 10^{-3} \pm 3.23 \cdot 10^{-5}}$
	4	$2.45 \cdot 10^{-2} \pm 3.00 \cdot 10^{-3}$ (–)	$2.25 \cdot 10^{-2} \pm 2.36 \cdot 10^{-3}$ (–)	$\mathbf{3.82 \cdot 10^{-3} \pm 5.09 \cdot 10^{-5}}$ (+)	$4.02 \cdot 10^{-3} \pm 3.30 \cdot 10^{-5}$
	5	$2.24 \cdot 10^{-2} \pm 2.40 \cdot 10^{-3}$ (–)	$1.98 \cdot 10^{-2} \pm 3.02 \cdot 10^{-3}$ (–)	$3.25 \cdot 10^{-3} \pm 3.31 \cdot 10^{-5}$ (≈)	$\mathbf{3.23 \cdot 10^{-3} \pm 4.57 \cdot 10^{-5}}$
	6	$2.17 \cdot 10^{-2} \pm 1.66 \cdot 10^{-3}$ (–)	$1.89 \cdot 10^{-2} \pm 1.60 \cdot 10^{-3}$ (–)	$3.25 \cdot 10^{-3} \pm 9.15 \cdot 10^{-5}$ (≈)	$\mathbf{3.20 \cdot 10^{-3} \pm 7.84 \cdot 10^{-5}}$
	10	$2.20 \cdot 10^{-2} \pm 2.10 \cdot 10^{-3}$ (–)	$1.82 \cdot 10^{-2} \pm 2.36 \cdot 10^{-3}$ (–)	$\mathbf{4.97 \cdot 10^{-3} \pm 3.33 \cdot 10^{-4}}$ (≈)	$5.01 \cdot 10^{-3} \pm 2.88 \cdot 10^{-4}$
	20	$2.37 \cdot 10^{-2} \pm 4.11 \cdot 10^{-3}$ (–)	$1.75 \cdot 10^{-2} \pm 1.39 \cdot 10^{-3}$ (–)	$\mathbf{5.58 \cdot 10^{-3} \pm 1.63 \cdot 10^{-4}}$ (≈)	$5.68 \cdot 10^{-3} \pm 2.12 \cdot 10^{-4}$
TNK	1	$1.01 \cdot 10^{-1} \pm 2.12 \cdot 10^{-2}$ (–)	$1.42 \cdot 10^{-2} \pm 2.16 \cdot 10^{-3}$ (–)	$9.36 \cdot 10^{-3} \pm 1.10 \cdot 10^{-3}$ (–)	$\mathbf{3.81 \cdot 10^{-3} \pm 3.15 \cdot 10^{-4}}$
	2	$6.63 \cdot 10^{-2} \pm 2.19 \cdot 10^{-2}$ (–)	$2.13 \cdot 10^{-2} \pm 3.89 \cdot 10^{-3}$ (–)	$9.14 \cdot 10^{-3} \pm 1.52 \cdot 10^{-3}$ (–)	$\mathbf{2.68 \cdot 10^{-3} \pm 2.64 \cdot 10^{-4}}$
	3	$4.75 \cdot 10^{-2} \pm 1.65 \cdot 10^{-2}$ (–)	$1.97 \cdot 10^{-2} \pm 4.01 \cdot 10^{-3}$ (–)	$1.03 \cdot 10^{-2} \pm 2.01 \cdot 10^{-3}$ (–)	$\mathbf{2.34 \cdot 10^{-3} \pm 1.24 \cdot 10^{-4}}$
	4	$4.29 \cdot 10^{-2} \pm 8.54 \cdot 10^{-3}$ (–)	$2.12 \cdot 10^{-2} \pm 3.24 \cdot 10^{-3}$ (–)	$8.88 \cdot 10^{-3} \pm 1.46 \cdot 10^{-3}$ (–)	$\mathbf{2.26 \cdot 10^{-3} \pm 1.64 \cdot 10^{-4}}$
	5	$3.56 \cdot 10^{-2} \pm 6.84 \cdot 10^{-3}$ (–)	$1.99 \cdot 10^{-2} \pm 4.01 \cdot 10^{-3}$ (–)	$1.04 \cdot 10^{-2} \pm 1.77 \cdot 10^{-3}$ (–)	$\mathbf{2.31 \cdot 10^{-3} \pm 7.63 \cdot 10^{-5}}$
	6	$3.27 \cdot 10^{-2} \pm 7.32 \cdot 10^{-3}$ (–)	$1.72 \cdot 10^{-2} \pm 2.30 \cdot 10^{-3}$ (–)	$1.14 \cdot 10^{-2} \pm 1.07 \cdot 10^{-3}$ (–)	$\mathbf{2.33 \cdot 10^{-3} \pm 1.28 \cdot 10^{-4}}$
	10	$3.84 \cdot 10^{-2} \pm 5.80 \cdot 10^{-3}$ (–)	$2.14 \cdot 10^{-2} \pm 3.90 \cdot 10^{-3}$ (–)	$1.11 \cdot 10^{-2} \pm 1.54 \cdot 10^{-3}$ (–)	$\mathbf{2.84 \cdot 10^{-3} \pm 1.19 \cdot 10^{-4}}$
	20	$4.53 \cdot 10^{-2} \pm 6.25 \cdot 10^{-3}$ (–)	$2.08 \cdot 10^{-2} \pm 3.36 \cdot 10^{-3}$ (–)	$1.85 \cdot 10^{-2} \pm 1.75 \cdot 10^{-3}$ (–)	$\mathbf{5.97 \cdot 10^{-3} \pm 5.56 \cdot 10^{-4}}$
CTP1	1	$2.29 \cdot 10^{-2} \pm 4.86 \cdot 10^{-3}$ (–)	$1.87 \cdot 10^{-2} \pm 2.91 \cdot 10^{-3}$ (–)	$\mathbf{4.39 \cdot 10^{-3} \pm 1.56 \cdot 10^{-4}}$ (≈)	$4.48 \cdot 10^{-3} \pm 2.87 \cdot 10^{-4}$
	2	$3.43 \cdot 10^{-2} \pm 3.52 \cdot 10^{-3}$ (–)	$3.62 \cdot 10^{-2} \pm 8.82 \cdot 10^{-3}$ (–)	$6.82 \cdot 10^{-3} \pm 1.72 \cdot 10^{-3}$ (≈)	$\mathbf{6.38 \cdot 10^{-3} \pm 1.41 \cdot 10^{-3}}$
	3	$3.13 \cdot 10^{-2} \pm 6.48 \cdot 10^{-3}$ (–)	$2.75 \cdot 10^{-2} \pm 6.81 \cdot 10^{-3}$ (–)	$\mathbf{4.93 \cdot 10^{-3} \pm 4.39 \cdot 10^{-4}}$ (≈)	$5.00 \cdot 10^{-3} \pm 8.95 \cdot 10^{-4}$
	4	$2.91 \cdot 10^{-2} \pm 2.52 \cdot 10^{-3}$ (–)	$3.26 \cdot 10^{-2} \pm 4.12 \cdot 10^{-3}$ (–)	$5.12 \cdot 10^{-3} \pm 4.82 \cdot 10^{-4}$ (≈)	$\mathbf{5.06 \cdot 10^{-3} \pm 4.51 \cdot 10^{-4}}$
	5	$3.56 \cdot 10^{-2} \pm 6.89 \cdot 10^{-3}$ (–)	$3.38 \cdot 10^{-2} \pm 5.08 \cdot 10^{-3}$ (–)	$\mathbf{5.24 \cdot 10^{-3} \pm 4.87 \cdot 10^{-4}}$ (≈)	$5.24 \cdot 10^{-3} \pm 4.87 \cdot 10^{-4}$
	6	$3.17 \cdot 10^{-2} \pm 5.38 \cdot 10^{-3}$ (–)	$2.98 \cdot 10^{-2} \pm 2.93 \cdot 10^{-3}$ (–)	$\mathbf{4.64 \cdot 10^{-3} \pm 2.99 \cdot 10^{-4}}$ (≈)	$4.64 \cdot 10^{-3} \pm 2.99 \cdot 10^{-4}$
	10	$3.43 \cdot 10^{-2} \pm 6.21 \cdot 10^{-3}$ (–)	$2.85 \cdot 10^{-2} \pm 3.84 \cdot 10^{-3}$ (–)	$\mathbf{5.59 \cdot 10^{-3} \pm 3.13 \cdot 10^{-4}}$ (≈)	$5.71 \cdot 10^{-3} \pm 3.34 \cdot 10^{-4}$
	20	$3.47 \cdot 10^{-2} \pm 4.60 \cdot 10^{-3}$ (–)	$2.91 \cdot 10^{-2} \pm 5.41 \cdot 10^{-3}$ (–)	$8.78 \cdot 10^{-3} \pm 1.04 \cdot 10^{-3}$ (≈)	$\mathbf{8.26 \cdot 10^{-3} \pm 1.26 \cdot 10^{-3}}$
C3DTLZ4	1	$3.69 \cdot 10^{-2} \pm 1.18 \cdot 10^{-2}$ (–)	$7.80 \cdot 10^{-2} \pm 3.13 \cdot 10^{-2}$ (–)	$4.38 \cdot 10^{-2} \pm 6.68 \cdot 10^{-3}$ (–)	$\mathbf{5.71 \cdot 10^{-3} \pm 6.34 \cdot 10^{-4}}$
	2	$4.23 \cdot 10^{-2} \pm 1.83 \cdot 10^{-2}$ (–)	$3.22 \cdot 10^{-2} \pm 1.46 \cdot 10^{-2}$ (–)	$6.59 \cdot 10^{-2} \pm 7.44 \cdot 10^{-3}$ (–)	$\mathbf{4.63 \cdot 10^{-3} \pm 1.08 \cdot 10^{-3}}$
	3	$2.12 \cdot 10^{-2} \pm 3.48 \cdot 10^{-3}$ (–)	$1.77 \cdot 10^{-2} \pm 4.18 \cdot 10^{-3}$ (–)	$4.79 \cdot 10^{-2} \pm 6.48 \cdot 10^{-3}$ (–)	$\mathbf{2.48 \cdot 10^{-3} \pm 1.45 \cdot 10^{-4}}$
	4	$1.98 \cdot 10^{-2} \pm 1.49 \cdot 10^{-3}$ (–)	$1.33 \cdot 10^{-2} \pm 1.59 \cdot 10^{-3}$ (–)	$5.18 \cdot 10^{-2} \pm 4.42 \cdot 10^{-3}$ (–)	$\mathbf{2.42 \cdot 10^{-3} \pm 1.64 \cdot 10^{-4}}$
	5	$1.86 \cdot 10^{-2} \pm 1.81 \cdot 10^{-3}$ (–)	$1.20 \cdot 10^{-2} \pm 1.58 \cdot 10^{-3}$ (–)	$4.75 \cdot 10^{-2} \pm 5.26 \cdot 10^{-3}$ (–)	$\mathbf{2.23 \cdot 10^{-3} \pm 1.31 \cdot 10^{-4}}$
	6	$1.68 \cdot 10^{-2} \pm 1.94 \cdot 10^{-3}$ (–)	$1.00 \cdot 10^{-2} \pm 1.02 \cdot 10^{-3}$ (–)	$4.51 \cdot 10^{-2} \pm 8.05 \cdot 10^{-3}$ (–)	$\mathbf{2.15 \cdot 10^{-3} \pm 7.18 \cdot 10^{-5}}$
	10	$1.66 \cdot 10^{-2} \pm 2.44 \cdot 10^{-3}$ (–)	$8.14 \cdot 10^{-3} \pm 7.08 \cdot 10^{-4}$ (–)	$5.22 \cdot 10^{-2} \pm 1.80 \cdot 10^{-2}$ (–)	$\mathbf{2.33 \cdot 10^{-3} \pm 1.67 \cdot 10^{-4}}$
	20	$1.91 \cdot 10^{-2} \pm 2.98 \cdot 10^{-3}$ (–)	$8.04 \cdot 10^{-3} \pm 5.96 \cdot 10^{-4}$ (–)	$6.22 \cdot 10^{-2} \pm 1.54 \cdot 10^{-2}$ (–)	$\mathbf{2.71 \cdot 10^{-3} \pm 1.66 \cdot 10^{-4}}$
OSY	1	$1.08 \cdot 10^{-1} \pm 7.19 \cdot 10^{-2}$ (–)	$4.87 \cdot 10^{-2} \pm 1.38 \cdot 10^{-2}$ (–)	$\mathbf{9.78 \cdot 10^{-4} \pm 1.23 \cdot 10^{-4}}$ (+)	$1.07 \cdot 10^{-3} \pm 4.00 \cdot 10^{-5}$
	2	$3.11 \cdot 10^{-2} \pm 1.13 \cdot 10^{-2}$ (–)	$3.23 \cdot 10^{-2} \pm 9.08 \cdot 10^{-3}$ (–)	$9.60 \cdot 10^{-4} \pm 4.80 \cdot 10^{-5}$ (–)	$\mathbf{8.35 \cdot 10^{-4} \pm 8.30 \cdot 10^{-5}}$
	3	$2.05 \cdot 10^{-2} \pm 6.24 \cdot 10^{-3}$ (–)	$1.61 \cdot 10^{-2} \pm 3.05 \cdot 10^{-3}$ (–)	$9.91 \cdot 10^{-4} \pm 6.88 \cdot 10^{-5}$ (–)	$\mathbf{9.38 \cdot 10^{-4} \pm 4.70 \cdot 10^{-5}}$
	4	$1.69 \cdot 10^{-2} \pm 4.00 \cdot 10^{-3}$ (–)	$1.54 \cdot 10^{-2} \pm 4.47 \cdot 10^{-3}$ (–)	$\mathbf{1.24 \cdot 10^{-3} \pm 7.74 \cdot 10^{-5}}$ (≈)	$1.26 \cdot 10^{-3} \pm 1.30 \cdot 10^{-4}$
	5	$1.35 \cdot 10^{-2} \pm 3.60 \cdot 10^{-3}$ (–)	$1.23 \cdot 10^{-2} \pm 2.71 \cdot 10^{-3}$ (–)	$1.54 \cdot 10^{-3} \pm 8.39 \cdot 10^{-5}$ (≈)	$\mathbf{1.54 \cdot 10^{-3} \pm 1.02 \cdot 10^{-4}}$
	6	$1.21 \cdot 10^{-2} \pm 2.18 \cdot 10^{-3}$ (–)	$1.17 \cdot 10^{-2} \pm 2.25 \cdot 10^{-3}$ (–)	$2.14 \cdot 10^{-3} \pm 1.76 \cdot 10^{-4}$ (≈)	$\mathbf{2.01 \cdot 10^{-3} \pm 1.61 \cdot 10^{-4}}$
	10	$1.14 \cdot 10^{-2} \pm 3.75 \cdot 10^{-3}$ (–)	$1.20 \cdot 10^{-2} \pm 3.15 \cdot 10^{-3}$ (–)	$7.60 \cdot 10^{-3} \pm 1.10 \cdot 10^{-3}$ (≈)	$\mathbf{7.26 \cdot 10^{-3} \pm 8.61 \cdot 10^{-4}}$
	20	$\mathbf{1.31 \cdot 10^{-2} \pm 7.56 \cdot 10^{-3}}$ (+)	$1.31 \cdot 10^{-2} \pm 5.45 \cdot 10^{-3}$ (+)	$4.22 \cdot 10^{-2} \pm 1.06 \cdot 10^{-2}$ (≈)	$3.67 \cdot 10^{-2} \pm 5.01 \cdot 10^{-3}$
TBTD	1	$4.43 \cdot 10^{-2} \pm 3.30 \cdot 10^{-2}$ (–)	$2.20 \cdot 10^{-2} \pm 8.82 \cdot 10^{-3}$ (–)	$6.43 \cdot 10^{-3} \pm 9.72 \cdot 10^{-4}$ (–)	$\mathbf{4.27 \cdot 10^{-3} \pm 2.41 \cdot 10^{-3}}$
	2	$2.87 \cdot 10^{-2} \pm 6.48 \cdot 10^{-3}$ (–)	$1.46 \cdot 10^{-2} \pm 5.57 \cdot 10^{-3}$ (–)	$1.10 \cdot 10^{-2} \pm 2.68 \cdot 10^{-3}$ (–)	$\mathbf{5.94 \cdot 10^{-3} \pm 2.08 \cdot 10^{-3}}$
	3	$2.17 \cdot 10^{-2} \pm 3.41 \cdot 10^{-3}$ (–)	$1.40 \cdot 10^{-2} \pm 4.45 \cdot 10^{-3}$ (–)	$1.37 \cdot 10^{-2} \pm 5.00 \cdot 10^{-3}$ (–)	$\mathbf{5.24 \cdot 10^{-3} \pm 1.22 \cdot 10^{-3}}$
	4	$1.56 \cdot 10^{-2} \pm 3.89 \cdot 10^{-3}$ (–)	$1.14 \cdot 10^{-2} \pm 2.67 \cdot 10^{-3}$ (–)	$1.46 \cdot 10^{-2} \pm 3.13 \cdot 10^{-3}$ (–)	$\mathbf{6.47 \cdot 10^{-3} \pm 1.29 \cdot 10^{-3}}$
	5	$1.72 \cdot 10^{-2} \pm 3.94 \cdot 10^{-3}$ (–)	$1.20 \cdot 10^{-2} \pm 2.80 \cdot 10^{-3}$ (–)	$1.19 \cdot 10^{-2} \pm 2.65 \cdot 10^{-3}$ (–)	$\mathbf{6.60 \cdot 10^{-3} \pm 1.12 \cdot 10^{-3}}$
	6	$1.38 \cdot 10^{-2} \pm 4.03 \cdot 10^{-3}$ (–)	$1.12 \cdot 10^{-2} \pm 2.78 \cdot 10^{-3}$ (≈)	$1.53 \cdot 10^{-2} \pm 4.81 \cdot 10^{-3}$ (–)	$\mathbf{8.85 \cdot 10^{-3} \pm 2.62 \cdot 10^{-3}}$
	10	$1.20 \cdot 10^{-2} \pm 3.09 \cdot 10^{-3}$ (≈)	$1.07 \cdot 10^{-2} \pm 3.27 \cdot 10^{-3}$ (≈)	$1.56 \cdot 10^{-2} \pm 6.24 \cdot 10^{-3}$ (–)	$\mathbf{1.02 \cdot 10^{-2} \pm 1.53 \cdot 10^{-3}}$
	20	$1.14 \cdot 10^{-2} \pm 1.54 \cdot 10^{-3}$ (≈)	$\mathbf{9.82 \cdot 10^{-3} \pm 2.00 \cdot 10^{-3}}$ (+)	$1.55 \cdot 10^{-2} \pm 3.37 \cdot 10^{-3}$ (≈)	$1.36 \cdot 10^{-2} \pm 3.73 \cdot 10^{-3}$
NBP	1	$1.83 \cdot 10^{-2} \pm 2.50 \cdot 10^{-3}$ (–)	$1.82 \cdot 10^{-2} \pm 4.90 \cdot 10^{-3}$ (–)	$3.76 \cdot 10^{-3} \pm 1.78 \cdot 10^{-4}$ (–)	$\mathbf{2.33 \cdot 10^{-3} \pm 3.94 \cdot 10^{-5}}$
	2	$2.32 \cdot 10^{-2} \pm 4.35 \cdot 10^{-3}$ (–)	$2.27 \cdot 10^{-2} \pm 2.88 \cdot 10^{-3}$ (–)	$3.64 \cdot 10^{-3} \pm 4.00 \cdot 10^{-4}$ (–)	$\mathbf{2.32 \cdot 10^{-3} \pm 5.64 \cdot 10^{-5}}$
	3	$2.00 \cdot 10^{-2} \pm 3.63 \cdot 10^{-3}$ (–)	$1.91 \cdot 10^{-2} \pm 1.80 \cdot 10^{-3}$ (–)	$3.64 \cdot 10^{-3} \pm 1.72 \cdot 10^{-4}$ (–)	$\mathbf{2.46 \cdot 10^{-3} \pm 1.46 \cdot 10^{-5}}$
	4	$2.60 \cdot 10^{-2} \pm 4.15 \cdot 10^{-3}$ (–)	$2.13 \cdot 10^{-2} \pm 3.90 \cdot 10^{-3}$ (–)	$3.87 \cdot 10^{-3} \pm 2.37 \cdot 10^{-4}$ (–)	$\mathbf{2.45 \cdot 10^{-3} \pm 2.87 \cdot 10^{-5}}$
	5	$2.23 \cdot 10^{-2} \pm 2.68 \cdot 10^{-3}$ (–)	$1.88 \cdot 10^{-2} \pm 2.01 \cdot 10^{-3}$ (–)	$4.41 \cdot 10^{-3} \pm 4.01 \cdot 10^{-4}$ (–)	$\mathbf{2.88 \cdot 10^{-3} \pm 9.64 \cdot 10^{-5}}$
	6	$2.09 \cdot 10^{-2} \pm 2.49 \cdot 10^{-3}$ (–)	$1.98 \cdot 10^{-2} \pm 3.48 \cdot 10^{-3}$ (–)	$4.03 \cdot 10^{-3} \pm 3.02 \cdot 10^{-4}$ (–)	$\mathbf{3.03 \cdot 10^{-3} \pm 7.37 \cdot 10^{-5}}$
	10	$2.11 \cdot 10^{-2} \pm 2.92 \cdot 10^{-3}$ (–)	$1.77 \cdot 10^{-2} \pm 2.28 \cdot 10^{-3}$ (–)	$5.38 \cdot 10^{-3} \pm 4.40 \cdot 10^{-4}$ (–)	$\mathbf{3.58 \cdot 10^{-3} \pm 2.04 \cdot 10^{-4}}$
	20	$2.29 \cdot 10^{-2} \pm 3.26 \cdot 10^{-3}$ (–)	$1.94 \cdot 10^{-2} \pm 2.35 \cdot 10^{-3}$ (–)	$1.44 \cdot 10^{-2} \pm 3.86 \cdot 10^{-3}$ (–)	$\mathbf{6.09 \cdot 10^{-3} \pm 3.19 \cdot 10^{-4}}$
DBD	1	$1.17 \cdot 10^{-2} \pm 3.63 \cdot 10^{-3}$ (–)	$2.43 \cdot 10^{-2} \pm 1.51 \cdot 10^{-2}$ (–)	$\mathbf{1.35 \cdot 10^{-3} \pm 3.36 \cdot 10^{-4}}$ (+)	$2.26 \cdot 10^{-3} \pm 1.40 \cdot 10^{-3}$
	2	$1.41 \cdot 10^{-2} \pm 2.84 \cdot 10^{-3}$ (–)	$2.06 \cdot 10^{-2} \pm 4.21 \cdot 10^{-3}$ (–)	$\mathbf{1.24 \cdot 10^{-3} \pm 2.49 \cdot 10^{-4}}$ (+)	$4.29 \cdot 10^{-3} \pm 1.55 \cdot 10^{-3}$
	3	$1.01 \cdot 10^{-2} \pm 1.49 \cdot 10^{-3}$ (–)	$2.01 \cdot 10^{-2} \pm 6.65 \cdot 10^{-3}$ (–)	$1.32 \cdot 10^{-3} \pm 3.12 \cdot 10^{-4}$ (–)	$\mathbf{9.60 \cdot 10^{-4} \pm 5.64 \cdot 10^{-5}}$
	4	$1.39 \cdot 10^{-2} \pm 2.75 \cdot 10^{-3}$ (–)	$1.69 \cdot 10^{-2} \pm 2.64 \cdot 10^{-3}$ (–)	$2.89 \cdot 10^{-3} \pm 1.54 \cdot 10^{-3}$ (–)	$\mathbf{1.18 \cdot 10^{-3} \pm 4.70 \cdot 10^{-5}}$
	5	$1.42 \cdot 10^{-2} \pm 4.50 \cdot 10^{-3}$ (–)	$1.75 \cdot 10^{-2} \pm 3.70 \cdot 10^{-3}$ (–)	$1.84 \cdot 10^{-3} \pm 1.04 \cdot 10^{-3}$ (≈)	$\mathbf{1.31 \cdot 10^{-3} \pm 8.45 \cdot 10^{-5}}$
	6	$1.07 \cdot 10^{-2} \pm 1.31 \cdot 10^{-3}$ (–)	$2.10 \cdot 10^{-2} \pm 8.34 \cdot 10^{-3}$ (–)	$\mathbf{2.08 \cdot 10^{-3} \pm 7.46 \cdot 10^{-4}}$ (≈)	$2.21 \cdot 10^{-3} \pm 7.33 \cdot 10^{-4}$
	10	$1.22 \cdot 10^{-2} \pm 2.49 \cdot 10^{-3}$ (–)	$1.72 \cdot 10^{-2} \pm 4.16 \cdot 10^{-3}$ (–)	$2.62 \cdot 10^{-3} \pm 5.78 \cdot 10^{-4}$ (≈)	$\mathbf{2.33 \cdot 10^{-3} \pm 2.25 \cdot 10^{-4}}$
	20	$1.30 \cdot 10^{-2} \pm 3.30 \cdot 10^{-3}$ (–)	$2.06 \cdot 10^{-2} \pm 5.11 \cdot 10^{-3}$ (–)	$\mathbf{3.74 \cdot 10^{-3} \pm 2.29 \cdot 10^{-4}}$ (+)	$4.09 \cdot 10^{-3} \pm 3.45 \cdot 10^{-4}$
SRD	1	$4.42 \cdot 10^{-3} \pm 3.15 \cdot 10^{-3}$ (≈)	$1.86 \cdot 10^{-2} \pm 9.63 \cdot 10^{-3}$ (–)	$\mathbf{6.51 \cdot 10^{-4} \pm 9.70 \cdot 10^{-5}}$ (+)	$2.45 \cdot 10^{-3} \pm 9.97 \cdot 10^{-4}$
	2	$2.26 \cdot 10^{-3} \pm 1.37 \cdot 10^{-3}$ (≈)	$2.09 \cdot 10^{-3} \pm 7.84 \cdot 10^{-4}$ (–)	$\mathbf{1.24 \cdot 10^{-3} \pm 1.31 \cdot 10^{-4}}$ (+)	$1.40 \cdot 10^{-3} \pm 1.77 \cdot 10^{-4}$
	3	$1.22 \cdot 10^{-3} \pm 4.50 \cdot 10^{-4}$ (≈)	$\mathbf{1.03 \cdot 10^{-3} \pm 2.35 \cdot 10^{-4}}$ (+)	$1.91 \cdot 10^{-3} \pm 1.38 \cdot 10^{-4}$ (–)	$1.50 \cdot 10^{-3} \pm 1.16 \cdot 10^{-4}$
	4	$1.15 \cdot 10^{-3} \pm 1.29 \cdot 10^{-4}$ (+)	$\mathbf{1.09 \cdot 10^{-3} \pm 1.64 \cdot 10^{-4}}$ (+)	$2.11 \cdot 10^{-3} \pm 3.10 \cdot 10^{-4}$ (–)	$1.58 \cdot 10^{-3} \pm 1.41 \cdot 10^{-4}$
	5	$1.09 \cdot 10^{-3} \pm 1.41 \cdot 10^{-4}$ (+)	$\mathbf{9.05 \cdot 10^{-4} \pm 8.43 \cdot 10^{-5}}$ (+)	$2.20 \cdot 10^{-3} \pm 3.38 \cdot 10^{-4}$ (–)	$1.56 \cdot 10^{-3} \pm 9.89 \cdot 10^{-5}$
	6	$1.13 \cdot 10^{-3} \pm 1.61 \cdot 10^{-4}$ (+)	$\mathbf{9.45 \cdot 10^{-4} \pm 1.19 \cdot 10^{-4}}$ (+)	$2.27 \cdot 10^{-3} \pm 3.09 \cdot 10^{-4}$ (–)	$1.70 \cdot 10^{-3} \pm 1.29 \cdot 10^{-4}$
	10	$1.19 \cdot 10^{-3} \pm 1.44 \cdot 10^{-4}$ (+)	$\mathbf{9.06 \cdot 10^{-4} \pm 9.72 \cdot 10^{-5}}$ (+)	$2.95 \cdot 10^{-3} \pm 2.89 \cdot 10^{-4}$ (–)	$2.26 \cdot 10^{-3} \pm 2.06 \cdot 10^{-4}$
	20	$1.25 \cdot 10^{-3} \pm 1.61 \cdot 10^{-4}$ (+)	$\mathbf{1.09 \cdot 10^{-3} \pm 1.24 \cdot 10^{-4}}$ (+)	$4.92 \cdot 10^{-3} \pm 4.88 \cdot 10^{-4}$ (–)	$3.20 \cdot 10^{-3} \pm 3.17 \cdot 10^{-4}$

Table continues on next page.

Continuation of Table 5.10.

Function	p	SA-NSGA-II	IC-SA-NSGA-II	SAMO-COBRA	IOC-SAMO-COBRA
WB	1	$2.57 \cdot 10^{-1} \pm 8.72 \cdot 10^{-2}$ (−)	**$2.10 \cdot 10^{-2} \pm 2.89 \cdot 10^{-2}$** (+)	$7.51 \cdot 10^{-2} \pm 1.35 \cdot 10^{-2}$ (−)	$2.63 \cdot 10^{-2} \pm 2.92 \cdot 10^{-3}$
	2	$1.10 \cdot 10^{-1} \pm 6.20 \cdot 10^{-2}$ (−)	**$1.99 \cdot 10^{-2} \pm 4.07 \cdot 10^{-3}$** (≈)	$6.06 \cdot 10^{-2} \pm 2.23 \cdot 10^{-2}$ (−)	$2.65 \cdot 10^{-2} \pm 1.23 \cdot 10^{-2}$
	3	$7.69 \cdot 10^{-2} \pm 5.10 \cdot 10^{-2}$ (−)	**$1.47 \cdot 10^{-2} \pm 4.74 \cdot 10^{-3}$** (+)	$3.65 \cdot 10^{-2} \pm 1.71 \cdot 10^{-2}$ (≈)	$2.68 \cdot 10^{-2} \pm 7.70 \cdot 10^{-3}$
	4	$4.66 \cdot 10^{-2} \pm 2.34 \cdot 10^{-2}$ (≈)	**$1.48 \cdot 10^{-2} \pm 2.35 \cdot 10^{-3}$** (+)	$6.29 \cdot 10^{-2} \pm 2.13 \cdot 10^{-2}$ (−)	$2.92 \cdot 10^{-2} \pm 1.05 \cdot 10^{-2}$
	5	$8.12 \cdot 10^{-2} \pm 8.78 \cdot 10^{-2}$ (−)	**$1.59 \cdot 10^{-2} \pm 3.51 \cdot 10^{-3}$** (+)	$6.44 \cdot 10^{-2} \pm 3.21 \cdot 10^{-2}$ (−)	$2.74 \cdot 10^{-2} \pm 1.57 \cdot 10^{-2}$
	6	$6.67 \cdot 10^{-2} \pm 4.83 \cdot 10^{-2}$ (≈)	**$1.41 \cdot 10^{-2} \pm 2.51 \cdot 10^{-3}$** (+)	$7.32 \cdot 10^{-2} \pm 2.59 \cdot 10^{-2}$ (−)	$4.19 \cdot 10^{-2} \pm 2.31 \cdot 10^{-2}$
	10	$5.06 \cdot 10^{-2} \pm 1.16 \cdot 10^{-2}$ (≈)	**$1.33 \cdot 10^{-2} \pm 3.94 \cdot 10^{-3}$** (+)	$7.73 \cdot 10^{-2} \pm 2.58 \cdot 10^{-2}$ (−)	$4.89 \cdot 10^{-2} \pm 8.56 \cdot 10^{-3}$
	20	$8.73 \cdot 10^{-2} \pm 1.03 \cdot 10^{-1}$ (≈)	**$1.38 \cdot 10^{-2} \pm 2.70 \cdot 10^{-3}$** (+)	$7.29 \cdot 10^{-2} \pm 1.18 \cdot 10^{-2}$ (≈)	$6.47 \cdot 10^{-2} \pm 1.55 \cdot 10^{-2}$
BICOP1	1	$6.41 \cdot 10^{-1} \pm 2.12 \cdot 10^{-1}$ (−)	$6.35 \cdot 10^{-1} \pm 3.17 \cdot 10^{-1}$ (−)	**$2.89 \cdot 10^{-1} \pm 9.21 \cdot 10^{-2}$** (≈)	$3.48 \cdot 10^{-1} \pm 7.81 \cdot 10^{-2}$
	2	$3.58 \cdot 10^{-2} \pm 1.03 \cdot 10^{-2}$ (+)	**$3.10 \cdot 10^{-2} \pm 6.98 \cdot 10^{-3}$** (+)	$2.45 \cdot 10^{-1} \pm 2.17 \cdot 10^{-1}$ (≈)	$1.23 \cdot 10^{-1} \pm 1.04 \cdot 10^{-1}$
	3	$1.88 \cdot 10^{-2} \pm 5.26 \cdot 10^{-3}$ (≈)	**$1.56 \cdot 10^{-2} \pm 2.56 \cdot 10^{-3}$** (≈)	$8.36 \cdot 10^{-2} \pm 1.34 \cdot 10^{-1}$ (≈)	$4.29 \cdot 10^{-2} \pm 4.04 \cdot 10^{-2}$
	4	$1.26 \cdot 10^{-2} \pm 3.18 \cdot 10^{-3}$ (≈)	**$1.15 \cdot 10^{-2} \pm 2.99 \cdot 10^{-3}$** (+)	$1.67 \cdot 10^{-2} \pm 5.16 \cdot 10^{-3}$ (≈)	$2.87 \cdot 10^{-2} \pm 3.66 \cdot 10^{-2}$
	5	$8.76 \cdot 10^{-3} \pm 2.13 \cdot 10^{-3}$ (+)	**$8.44 \cdot 10^{-3} \pm 2.83 \cdot 10^{-3}$** (+)	$2.13 \cdot 10^{-2} \pm 6.69 \cdot 10^{-3}$ (≈)	$2.44 \cdot 10^{-2} \pm 7.39 \cdot 10^{-3}$
	6	$6.67 \cdot 10^{-3} \pm 2.16 \cdot 10^{-3}$ (+)	**$6.62 \cdot 10^{-3} \pm 1.66 \cdot 10^{-3}$** (+)	$4.21 \cdot 10^{-2} \pm 1.06 \cdot 10^{-2}$ (≈)	$3.69 \cdot 10^{-2} \pm 8.02 \cdot 10^{-3}$
	10	**$3.38 \cdot 10^{-3} \pm 5.54 \cdot 10^{-4}$** (+)	$3.78 \cdot 10^{-3} \pm 9.19 \cdot 10^{-4}$ (+)	$1.03 \cdot 10^{-1} \pm 3.26 \cdot 10^{-2}$ (≈)	$9.08 \cdot 10^{-2} \pm 1.96 \cdot 10^{-2}$
	20	$3.41 \cdot 10^{-3} \pm 4.23 \cdot 10^{-4}$ (+)	**$3.36 \cdot 10^{-3} \pm 4.14 \cdot 10^{-4}$** (+)	$2.42 \cdot 10^{-1} \pm 6.91 \cdot 10^{-2}$ (≈)	$2.70 \cdot 10^{-1} \pm 5.84 \cdot 10^{-2}$
BICOP2	1	$1.83 \cdot 10^{-1} \pm 1.21 \cdot 10^{-2}$ (−)	$1.59 \cdot 10^{-1} \pm 2.57 \cdot 10^{-2}$ (−)	$7.70 \cdot 10^{-2} \pm 2.97 \cdot 10^{-2}$ (−)	**$2.93 \cdot 10^{-2} \pm 9.30 \cdot 10^{-3}$**
	2	$1.73 \cdot 10^{-1} \pm 3.16 \cdot 10^{-2}$ (−)	$1.07 \cdot 10^{-1} \pm 2.61 \cdot 10^{-2}$ (−)	$7.41 \cdot 10^{-2} \pm 3.19 \cdot 10^{-2}$ (−)	**$1.60 \cdot 10^{-2} \pm 1.77 \cdot 10^{-2}$**
	3	$1.58 \cdot 10^{-1} \pm 2.53 \cdot 10^{-2}$ (−)	$1.28 \cdot 10^{-1} \pm 4.18 \cdot 10^{-2}$ (−)	$7.19 \cdot 10^{-2} \pm 3.52 \cdot 10^{-2}$ (−)	**$2.31 \cdot 10^{-2} \pm 3.25 \cdot 10^{-2}$**
	4	$1.63 \cdot 10^{-1} \pm 2.98 \cdot 10^{-2}$ (−)	$1.17 \cdot 10^{-1} \pm 4.08 \cdot 10^{-2}$ (−)	$7.00 \cdot 10^{-2} \pm 4.01 \cdot 10^{-2}$ (≈)	**$5.61 \cdot 10^{-2} \pm 4.71 \cdot 10^{-2}$**
	5	$1.61 \cdot 10^{-1} \pm 2.69 \cdot 10^{-2}$ (−)	$1.08 \cdot 10^{-1} \pm 3.34 \cdot 10^{-2}$ (≈)	**$4.78 \cdot 10^{-2} \pm 1.21 \cdot 10^{-2}$** (≈)	$7.25 \cdot 10^{-2} \pm 4.46 \cdot 10^{-2}$
	6	$1.59 \cdot 10^{-1} \pm 2.69 \cdot 10^{-2}$ (−)	$1.26 \cdot 10^{-1} \pm 3.67 \cdot 10^{-2}$ (−)	**$4.46 \cdot 10^{-2} \pm 8.25 \cdot 10^{-3}$** (≈)	$7.96 \cdot 10^{-2} \pm 4.29 \cdot 10^{-2}$
	10	$1.30 \cdot 10^{-1} \pm 3.10 \cdot 10^{-2}$ (−)	$1.35 \cdot 10^{-1} \pm 3.17 \cdot 10^{-2}$ (−)	$5.90 \cdot 10^{-2} \pm 1.55 \cdot 10^{-2}$ (≈)	**$5.68 \cdot 10^{-2} \pm 2.92 \cdot 10^{-2}$**
	20	$1.35 \cdot 10^{-1} \pm 3.16 \cdot 10^{-2}$ (−)	$1.29 \cdot 10^{-1} \pm 3.42 \cdot 10^{-2}$ (−)	$7.75 \cdot 10^{-2} \pm 1.54 \cdot 10^{-2}$ (−)	**$3.76 \cdot 10^{-2} \pm 1.00 \cdot 10^{-2}$**
MW1	1	$1.00 \cdot 10^{+0} \pm 0.00 \cdot 10^{+0}$ (−)	$1.05 \cdot 10^{-1} \pm 3.91 \cdot 10^{-2}$ (−)	$7.18 \cdot 10^{-1} \pm 3.04 \cdot 10^{-1}$ (−)	**$6.09 \cdot 10^{-4} \pm 7.62 \cdot 10^{-5}$**
	2	$1.46 \cdot 10^{-1} \pm 7.65 \cdot 10^{-2}$ (−)	$4.35 \cdot 10^{-2} \pm 3.54 \cdot 10^{-3}$ (−)	$2.11 \cdot 10^{-1} \pm 1.19 \cdot 10^{-1}$ (−)	**$6.40 \cdot 10^{-4} \pm 1.02 \cdot 10^{-4}$**
	3	$9.49 \cdot 10^{-2} \pm 4.32 \cdot 10^{-2}$ (−)	$4.30 \cdot 10^{-2} \pm 7.40 \cdot 10^{-3}$ (−)	$8.58 \cdot 10^{-2} \pm 7.18 \cdot 10^{-2}$ (−)	**$7.57 \cdot 10^{-4} \pm 1.99 \cdot 10^{-4}$**
	4	$5.70 \cdot 10^{-2} \pm 3.73 \cdot 10^{-2}$ (−)	$3.46 \cdot 10^{-2} \pm 7.84 \cdot 10^{-3}$ (−)	$2.32 \cdot 10^{-1} \pm 2.49 \cdot 10^{-1}$ (−)	**$9.87 \cdot 10^{-4} \pm 1.55 \cdot 10^{-4}$**
	5	$3.47 \cdot 10^{-2} \pm 1.34 \cdot 10^{-2}$ (−)	$2.42 \cdot 10^{-2} \pm 4.97 \cdot 10^{-3}$ (−)	$2.38 \cdot 10^{-1} \pm 2.47 \cdot 10^{-1}$ (−)	**$1.07 \cdot 10^{-3} \pm 1.54 \cdot 10^{-4}$**
	6	$2.97 \cdot 10^{-2} \pm 1.20 \cdot 10^{-2}$ (−)	$1.47 \cdot 10^{-2} \pm 2.61 \cdot 10^{-3}$ (−)	$1.45 \cdot 10^{-1} \pm 1.31 \cdot 10^{-1}$ (−)	**$1.42 \cdot 10^{-3} \pm 3.54 \cdot 10^{-4}$**
	10	$3.23 \cdot 10^{-2} \pm 2.70 \cdot 10^{-2}$ (−)	$9.65 \cdot 10^{-3} \pm 1.91 \cdot 10^{-3}$ (−)	$2.63 \cdot 10^{-1} \pm 1.70 \cdot 10^{-1}$ (−)	**$2.31 \cdot 10^{-3} \pm 8.80 \cdot 10^{-4}$**
	20	$1.74 \cdot 10^{-2} \pm 1.23 \cdot 10^{-2}$ (≈)	**$8.06 \cdot 10^{-3} \pm 2.14 \cdot 10^{-3}$** (≈)	$1.91 \cdot 10^{-1} \pm 1.24 \cdot 10^{-1}$ (≈)	$1.81 \cdot 10^{-1} \pm 2.22 \cdot 10^{-1}$
MW2	1	$7.92 \cdot 10^{-1} \pm 2.55 \cdot 10^{-1}$ (−)	**$3.84 \cdot 10^{-2} \pm 9.30 \cdot 10^{-3}$** (+)	$3.14 \cdot 10^{-1} \pm 9.67 \cdot 10^{-2}$ (−)	$6.63 \cdot 10^{-2} \pm 1.40 \cdot 10^{-2}$
	2	$1.87 \cdot 10^{-1} \pm 7.99 \cdot 10^{-2}$ (−)	**$3.03 \cdot 10^{-2} \pm 3.62 \cdot 10^{-3}$** (+)	$3.06 \cdot 10^{-1} \pm 1.95 \cdot 10^{-1}$ (−)	$4.20 \cdot 10^{-2} \pm 1.15 \cdot 10^{-2}$
	3	$1.39 \cdot 10^{-1} \pm 7.28 \cdot 10^{-2}$ (−)	**$2.63 \cdot 10^{-2} \pm 5.49 \cdot 10^{-3}$** (+)	$2.74 \cdot 10^{-1} \pm 1.28 \cdot 10^{-1}$ (−)	$5.89 \cdot 10^{-2} \pm 4.97 \cdot 10^{-2}$
	4	$9.77 \cdot 10^{-2} \pm 6.52 \cdot 10^{-2}$ (≈)	**$2.59 \cdot 10^{-2} \pm 5.51 \cdot 10^{-3}$** (+)	$3.02 \cdot 10^{-1} \pm 1.31 \cdot 10^{-1}$ (−)	$1.00 \cdot 10^{-1} \pm 5.85 \cdot 10^{-2}$
	5	$1.12 \cdot 10^{-1} \pm 7.36 \cdot 10^{-2}$ (≈)	**$2.45 \cdot 10^{-2} \pm 5.14 \cdot 10^{-3}$** (+)	$3.25 \cdot 10^{-1} \pm 1.44 \cdot 10^{-1}$ (−)	$6.08 \cdot 10^{-2} \pm 3.79 \cdot 10^{-2}$
	6	$1.08 \cdot 10^{-1} \pm 7.13 \cdot 10^{-2}$ (−)	**$2.48 \cdot 10^{-2} \pm 5.52 \cdot 10^{-3}$** (+)	$3.52 \cdot 10^{-1} \pm 1.57 \cdot 10^{-1}$ (−)	$5.54 \cdot 10^{-2} \pm 3.19 \cdot 10^{-2}$
	10	$1.21 \cdot 10^{-1} \pm 8.27 \cdot 10^{-2}$ (≈)	**$2.34 \cdot 10^{-2} \pm 6.84 \cdot 10^{-3}$** (+)	$4.05 \cdot 10^{-1} \pm 1.08 \cdot 10^{-1}$ (−)	$7.55 \cdot 10^{-2} \pm 2.99 \cdot 10^{-2}$
	20	$1.21 \cdot 10^{-1} \pm 7.84 \cdot 10^{-2}$ (≈)	**$2.15 \cdot 10^{-2} \pm 5.98 \cdot 10^{-3}$** (+)	$3.82 \cdot 10^{-1} \pm 1.54 \cdot 10^{-1}$ (−)	$1.04 \cdot 10^{-1} \pm 3.26 \cdot 10^{-2}$
MW3	1	$6.07 \cdot 10^{-1} \pm 3.85 \cdot 10^{-1}$ (−)	$2.33 \cdot 10^{-2} \pm 4.57 \cdot 10^{-3}$ (−)	$4.14 \cdot 10^{-2} \pm 1.26 \cdot 10^{-2}$ (−)	**$2.53 \cdot 10^{-3} \pm 5.11 \cdot 10^{-4}$**
	2	$2.70 \cdot 10^{-2} \pm 8.21 \cdot 10^{-3}$ (−)	$1.70 \cdot 10^{-2} \pm 1.75 \cdot 10^{-3}$ (−)	$2.40 \cdot 10^{-2} \pm 4.48 \cdot 10^{-3}$ (−)	**$1.72 \cdot 10^{-3} \pm 8.36 \cdot 10^{-4}$**
	3	$1.78 \cdot 10^{-2} \pm 3.25 \cdot 10^{-3}$ (−)	$1.32 \cdot 10^{-2} \pm 1.40 \cdot 10^{-3}$ (−)	$1.32 \cdot 10^{-2} \pm 5.51 \cdot 10^{-3}$ (−)	**$1.32 \cdot 10^{-3} \pm 2.29 \cdot 10^{-4}$**
	4	$1.73 \cdot 10^{-2} \pm 1.18 \cdot 10^{-3}$ (−)	$1.12 \cdot 10^{-2} \pm 1.58 \cdot 10^{-3}$ (−)	$6.28 \cdot 10^{-3} \pm 2.94 \cdot 10^{-3}$ (−)	**$1.40 \cdot 10^{-3} \pm 1.12 \cdot 10^{-4}$**
	5	$1.55 \cdot 10^{-2} \pm 2.82 \cdot 10^{-3}$ (−)	$9.62 \cdot 10^{-3} \pm 1.17 \cdot 10^{-3}$ (−)	$6.39 \cdot 10^{-3} \pm 1.92 \cdot 10^{-3}$ (−)	**$1.93 \cdot 10^{-3} \pm 4.53 \cdot 10^{-4}$**
	6	$1.67 \cdot 10^{-2} \pm 1.46 \cdot 10^{-3}$ (−)	$8.82 \cdot 10^{-3} \pm 1.96 \cdot 10^{-3}$ (−)	$7.27 \cdot 10^{-3} \pm 2.02 \cdot 10^{-3}$ (−)	**$1.97 \cdot 10^{-3} \pm 3.23 \cdot 10^{-4}$**
	10	$1.38 \cdot 10^{-2} \pm 1.98 \cdot 10^{-3}$ (−)	$7.04 \cdot 10^{-3} \pm 5.85 \cdot 10^{-4}$ (−)	$1.03 \cdot 10^{-2} \pm 1.12 \cdot 10^{-3}$ (−)	**$2.91 \cdot 10^{-3} \pm 3.10 \cdot 10^{-4}$**
	20	$1.41 \cdot 10^{-2} \pm 2.59 \cdot 10^{-3}$ (−)	$7.15 \cdot 10^{-3} \pm 5.79 \cdot 10^{-4}$ (−)	$1.43 \cdot 10^{-2} \pm 1.63 \cdot 10^{-3}$ (−)	**$5.19 \cdot 10^{-3} \pm 3.65 \cdot 10^{-4}$**
MW11	1	$3.79 \cdot 10^{-1} \pm 2.23 \cdot 10^{-1}$ (−)	**$3.50 \cdot 10^{-2} \pm 8.38 \cdot 10^{-3}$** (+)	$1.75 \cdot 10^{-1} \pm 2.77 \cdot 10^{-1}$ (≈)	$7.91 \cdot 10^{-2} \pm 3.33 \cdot 10^{-2}$
	2	$1.02 \cdot 10^{-1} \pm 9.49 \cdot 10^{-2}$ (≈)	**$2.15 \cdot 10^{-2} \pm 4.27 \cdot 10^{-3}$** (+)	$1.65 \cdot 10^{-1} \pm 1.00 \cdot 10^{-1}$ (−)	$6.79 \cdot 10^{-2} \pm 2.65 \cdot 10^{-2}$
	3	$1.59 \cdot 10^{-1} \pm 1.21 \cdot 10^{-1}$ (−)	$1.88 \cdot 10^{-2} \pm 3.75 \cdot 10^{-3}$ (−)	$1.30 \cdot 10^{-1} \pm 7.44 \cdot 10^{-2}$ (−)	**$6.96 \cdot 10^{-3} \pm 4.95 \cdot 10^{-3}$**
	4	$1.99 \cdot 10^{-1} \pm 1.33 \cdot 10^{-1}$ (−)	$1.50 \cdot 10^{-2} \pm 2.65 \cdot 10^{-3}$ (−)	$1.44 \cdot 10^{-1} \pm 8.01 \cdot 10^{-2}$ (−)	**$4.78 \cdot 10^{-3} \pm 1.31 \cdot 10^{-3}$**
	5	$1.93 \cdot 10^{-1} \pm 1.31 \cdot 10^{-1}$ (−)	$1.49 \cdot 10^{-2} \pm 2.39 \cdot 10^{-3}$ (−)	$1.33 \cdot 10^{-1} \pm 8.63 \cdot 10^{-2}$ (−)	**$4.34 \cdot 10^{-3} \pm 6.51 \cdot 10^{-4}$**
	6	$2.62 \cdot 10^{-1} \pm 9.96 \cdot 10^{-2}$ (−)	$1.26 \cdot 10^{-2} \pm 2.12 \cdot 10^{-3}$ (−)	$1.53 \cdot 10^{-1} \pm 1.09 \cdot 10^{-1}$ (−)	**$4.29 \cdot 10^{-3} \pm 1.14 \cdot 10^{-3}$**
	10	$2.31 \cdot 10^{-1} \pm 1.23 \cdot 10^{-1}$ (−)	$9.83 \cdot 10^{-3} \pm 1.40 \cdot 10^{-3}$ (−)	$2.50 \cdot 10^{-1} \pm 8.82 \cdot 10^{-2}$ (−)	**$4.12 \cdot 10^{-3} \pm 1.00 \cdot 10^{-3}$**
	20	$3.02 \cdot 10^{-1} \pm 8.69 \cdot 10^{-2}$ (−)	$9.72 \cdot 10^{-3} \pm 1.20 \cdot 10^{-3}$ (−)	$2.28 \cdot 10^{-1} \pm 1.66 \cdot 10^{-2}$ (−)	**$8.07 \cdot 10^{-3} \pm 1.31 \cdot 10^{-3}$**
TRICOP	1	$6.21 \cdot 10^{-2} \pm 2.49 \cdot 10^{-2}$ (−)	$4.96 \cdot 10^{-2} \pm 1.36 \cdot 10^{-2}$ (−)	$1.04 \cdot 10^{-2} \pm 5.92 \cdot 10^{-5}$ (≈)	**$1.03 \cdot 10^{-2} \pm 5.14 \cdot 10^{-4}$**
	2	$9.52 \cdot 10^{-2} \pm 1.90 \cdot 10^{-2}$ (−)	$4.90 \cdot 10^{-2} \pm 9.31 \cdot 10^{-3}$ (−)	$1.08 \cdot 10^{-2} \pm 2.23 \cdot 10^{-4}$ (≈)	**$1.05 \cdot 10^{-2} \pm 5.52 \cdot 10^{-4}$**
	3	$8.17 \cdot 10^{-2} \pm 2.51 \cdot 10^{-2}$ (−)	$3.97 \cdot 10^{-2} \pm 5.04 \cdot 10^{-3}$ (−)	**$9.88 \cdot 10^{-3} \pm 1.41 \cdot 10^{-4}$** (+)	$1.06 \cdot 10^{-2} \pm 4.73 \cdot 10^{-4}$
	4	$7.72 \cdot 10^{-2} \pm 2.22 \cdot 10^{-2}$ (−)	$4.02 \cdot 10^{-2} \pm 6.24 \cdot 10^{-3}$ (−)	**$1.02 \cdot 10^{-2} \pm 3.85 \cdot 10^{-4}$** (+)	$1.07 \cdot 10^{-2} \pm 4.01 \cdot 10^{-4}$
	5	$8.22 \cdot 10^{-2} \pm 1.90 \cdot 10^{-2}$ (−)	$3.68 \cdot 10^{-2} \pm 2.98 \cdot 10^{-3}$ (−)	**$9.75 \cdot 10^{-3} \pm 3.34 \cdot 10^{-4}$** (≈)	$9.99 \cdot 10^{-3} \pm 4.32 \cdot 10^{-4}$
	6	$7.11 \cdot 10^{-2} \pm 2.22 \cdot 10^{-2}$ (−)	$3.76 \cdot 10^{-2} \pm 4.30 \cdot 10^{-3}$ (−)	**$1.02 \cdot 10^{-2} \pm 4.04 \cdot 10^{-4}$** (≈)	$1.03 \cdot 10^{-2} \pm 6.96 \cdot 10^{-4}$
	10	$6.58 \cdot 10^{-2} \pm 2.09 \cdot 10^{-2}$ (−)	$3.18 \cdot 10^{-2} \pm 4.24 \cdot 10^{-3}$ (−)	**$9.87 \cdot 10^{-3} \pm 3.53 \cdot 10^{-4}$** (≈)	$9.98 \cdot 10^{-3} \pm 3.69 \cdot 10^{-4}$
	20	$4.62 \cdot 10^{-2} \pm 1.10 \cdot 10^{-2}$ (−)	$2.63 \cdot 10^{-2} \pm 3.19 \cdot 10^{-3}$ (−)	$1.21 \cdot 10^{-2} \pm 6.52 \cdot 10^{-4}$ (≈)	**$1.20 \cdot 10^{-2} \pm 5.02 \cdot 10^{-4}$**
SPD	1	$5.95 \cdot 10^{-2} \pm 3.31 \cdot 10^{-3}$ (−)	$6.46 \cdot 10^{-2} \pm 5.16 \cdot 10^{-3}$ (−)	$1.55 \cdot 10^{-2} \pm 9.35 \cdot 10^{-4}$ (−)	**$8.78 \cdot 10^{-3} \pm 1.68 \cdot 10^{-4}$**
	2	$6.75 \cdot 10^{-2} \pm 9.27 \cdot 10^{-3}$ (−)	$5.78 \cdot 10^{-2} \pm 3.81 \cdot 10^{-3}$ (−)	$1.33 \cdot 10^{-2} \pm 1.15 \cdot 10^{-3}$ (−)	**$8.62 \cdot 10^{-3} \pm 2.00 \cdot 10^{-4}$**
	3	$6.02 \cdot 10^{-2} \pm 4.04 \cdot 10^{-3}$ (−)	$5.73 \cdot 10^{-2} \pm 5.38 \cdot 10^{-3}$ (−)	$1.22 \cdot 10^{-2} \pm 4.52 \cdot 10^{-4}$ (−)	**$8.61 \cdot 10^{-3} \pm 2.55 \cdot 10^{-4}$**
	4	$6.37 \cdot 10^{-2} \pm 5.26 \cdot 10^{-3}$ (−)	$6.03 \cdot 10^{-2} \pm 4.45 \cdot 10^{-3}$ (−)	$1.21 \cdot 10^{-2} \pm 3.87 \cdot 10^{-4}$ (−)	**$8.84 \cdot 10^{-3} \pm 1.78 \cdot 10^{-4}$**
	5	$5.96 \cdot 10^{-2} \pm 5.43 \cdot 10^{-3}$ (−)	$5.95 \cdot 10^{-2} \pm 7.26 \cdot 10^{-3}$ (−)	$1.29 \cdot 10^{-2} \pm 5.93 \cdot 10^{-4}$ (−)	**$8.77 \cdot 10^{-3} \pm 1.54 \cdot 10^{-4}$**
	6	$5.78 \cdot 10^{-2} \pm 2.68 \cdot 10^{-3}$ (−)	$5.87 \cdot 10^{-2} \pm 3.80 \cdot 10^{-3}$ (−)	$1.28 \cdot 10^{-2} \pm 5.15 \cdot 10^{-4}$ (−)	**$9.19 \cdot 10^{-3} \pm 1.91 \cdot 10^{-4}$**
	10	$5.90 \cdot 10^{-2} \pm 3.95 \cdot 10^{-3}$ (−)	$5.93 \cdot 10^{-2} \pm 5.02 \cdot 10^{-3}$ (−)	$1.51 \cdot 10^{-2} \pm 8.73 \cdot 10^{-4}$ (−)	**$9.68 \cdot 10^{-3} \pm 2.19 \cdot 10^{-4}$**
	20	$5.89 \cdot 10^{-2} \pm 6.15 \cdot 10^{-3}$ (−)	$6.10 \cdot 10^{-2} \pm 4.56 \cdot 10^{-3}$ (−)	$1.84 \cdot 10^{-2} \pm 1.02 \cdot 10^{-3}$ (−)	**$1.30 \cdot 10^{-2} \pm 3.34 \cdot 10^{-4}$**
CSI	1	$6.04 \cdot 10^{-2} \pm 3.48 \cdot 10^{-3}$ (−)	$1.10 \cdot 10^{-1} \pm 1.53 \cdot 10^{-2}$ (−)	$1.91 \cdot 10^{-2} \pm 2.07 \cdot 10^{-3}$ (≈)	**$1.82 \cdot 10^{-2} \pm 2.69 \cdot 10^{-3}$**
	2	$6.82 \cdot 10^{-2} \pm 5.25 \cdot 10^{-3}$ (−)	$7.00 \cdot 10^{-2} \pm 3.47 \cdot 10^{-3}$ (−)	$1.60 \cdot 10^{-2} \pm 5.08 \cdot 10^{-4}$ (−)	**$1.55 \cdot 10^{-2} \pm 4.95 \cdot 10^{-4}$**
	3	$6.37 \cdot 10^{-2} \pm 4.53 \cdot 10^{-3}$ (−)	$6.82 \cdot 10^{-2} \pm 6.44 \cdot 10^{-3}$ (−)	$1.54 \cdot 10^{-2} \pm 5.45 \cdot 10^{-4}$ (−)	**$1.48 \cdot 10^{-2} \pm 2.45 \cdot 10^{-4}$**
	4	$6.78 \cdot 10^{-2} \pm 3.57 \cdot 10^{-3}$ (−)	$6.84 \cdot 10^{-2} \pm 3.28 \cdot 10^{-3}$ (−)	$1.61 \cdot 10^{-2} \pm 4.98 \cdot 10^{-4}$ (−)	**$1.54 \cdot 10^{-2} \pm 3.11 \cdot 10^{-4}$**
	5	$6.93 \cdot 10^{-2} \pm 3.22 \cdot 10^{-3}$ (−)	$6.81 \cdot 10^{-2} \pm 7.55 \cdot 10^{-3}$ (−)	$1.60 \cdot 10^{-2} \pm 4.18 \cdot 10^{-4}$ (−)	**$1.55 \cdot 10^{-2} \pm 3.41 \cdot 10^{-4}$**
	6	$6.75 \cdot 10^{-2} \pm 3.50 \cdot 10^{-3}$ (−)	$6.59 \cdot 10^{-2} \pm 4.40 \cdot 10^{-3}$ (−)	$1.64 \cdot 10^{-2} \pm 3.98 \cdot 10^{-4}$ (≈)	**$1.61 \cdot 10^{-2} \pm 5.18 \cdot 10^{-4}$**
	10	$6.45 \cdot 10^{-2} \pm 5.78 \cdot 10^{-3}$ (−)	$6.60 \cdot 10^{-2} \pm 2.35 \cdot 10^{-3}$ (−)	$2.03 \cdot 10^{-2} \pm 7.23 \cdot 10^{-4}$ (≈)	**$2.00 \cdot 10^{-2} \pm 5.22 \cdot 10^{-4}$**
	20	$7.03 \cdot 10^{-2} \pm 7.85 \cdot 10^{-3}$ (−)	$6.45 \cdot 10^{-2} \pm 5.63 \cdot 10^{-3}$ (−)	**$3.78 \cdot 10^{-2} \pm 3.68 \cdot 10^{-3}$** (≈)	$3.94 \cdot 10^{-2} \pm 1.83 \cdot 10^{-3}$
WP	1	$2.77 \cdot 10^{-1} \pm 6.16 \cdot 10^{-2}$ (−)	$2.82 \cdot 10^{-1} \pm 4.32 \cdot 10^{-2}$ (−)	**$2.92 \cdot 10^{-2} \pm 9.66 \cdot 10^{-4}$** (≈)	$2.96 \cdot 10^{-2} \pm 1.52 \cdot 10^{-3}$
	2	$2.11 \cdot 10^{-1} \pm 2.80 \cdot 10^{-2}$ (−)	$2.19 \cdot 10^{-1} \pm 5.69 \cdot 10^{-2}$ (−)	**$2.85 \cdot 10^{-2} \pm 4.62 \cdot 10^{-4}$** (+)	$3.00 \cdot 10^{-2} \pm 9.23 \cdot 10^{-4}$
	3	$1.92 \cdot 10^{-1} \pm 2.40 \cdot 10^{-2}$ (−)	$2.02 \cdot 10^{-1} \pm 3.50 \cdot 10^{-2}$ (−)	**$3.03 \cdot 10^{-2} \pm 9.67 \cdot 10^{-4}$** (≈)	$3.07 \cdot 10^{-2} \pm 1.21 \cdot 10^{-3}$
	4	$1.79 \cdot 10^{-1} \pm 2.15 \cdot 10^{-2}$ (−)	$1.99 \cdot 10^{-1} \pm 2.96 \cdot 10^{-2}$ (−)	$2.97 \cdot 10^{-2} \pm 1.43 \cdot 10^{-3}$ (≈)	**$2.94 \cdot 10^{-2} \pm 9.56 \cdot 10^{-4}$**
	5	$1.82 \cdot 10^{-1} \pm 2.65 \cdot 10^{-2}$ (−)	$1.93 \cdot 10^{-1} \pm 3.50 \cdot 10^{-2}$ (−)	**$2.97 \cdot 10^{-2} \pm 7.08 \cdot 10^{-4}$** (≈)	$3.08 \cdot 10^{-2} \pm 1.36 \cdot 10^{-3}$
	6	$1.61 \cdot 10^{-1} \pm 1.77 \cdot 10^{-2}$ (−)	$1.94 \cdot 10^{-1} \pm 4.62 \cdot 10^{-2}$ (−)	**$2.95 \cdot 10^{-2} \pm 4.75 \cdot 10^{-4}$** (+)	$3.15 \cdot 10^{-2} \pm 1.15 \cdot 10^{-3}$
	10	$1.50 \cdot 10^{-1} \pm 3.08 \cdot 10^{-2}$ (−)	$1.76 \cdot 10^{-1} \pm 3.03 \cdot 10^{-2}$ (−)	$3.45 \cdot 10^{-2} \pm 1.80 \cdot 10^{-3}$ (−)	**$3.09 \cdot 10^{-2} \pm 1.14 \cdot 10^{-3}$**
	20	$1.57 \cdot 10^{-1} \pm 2.86 \cdot 10^{-2}$ (−)	$1.70 \cdot 10^{-1} \pm 3.11 \cdot 10^{-2}$ (−)	$3.38 \cdot 10^{-2} \pm 9.03 \cdot 10^{-4}$ (≈)	**$3.35 \cdot 10^{-2} \pm 1.01 \cdot 10^{-3}$**
Wilcoxon summary		148 − 17 ≈ 11+	139 − 6 ≈ 31+	99 − 66 ≈ 11+	reference algorithm
Mean Friedman rank		3.33	2.77	2.33	**1.56**

they are computationally inexpensive and available.

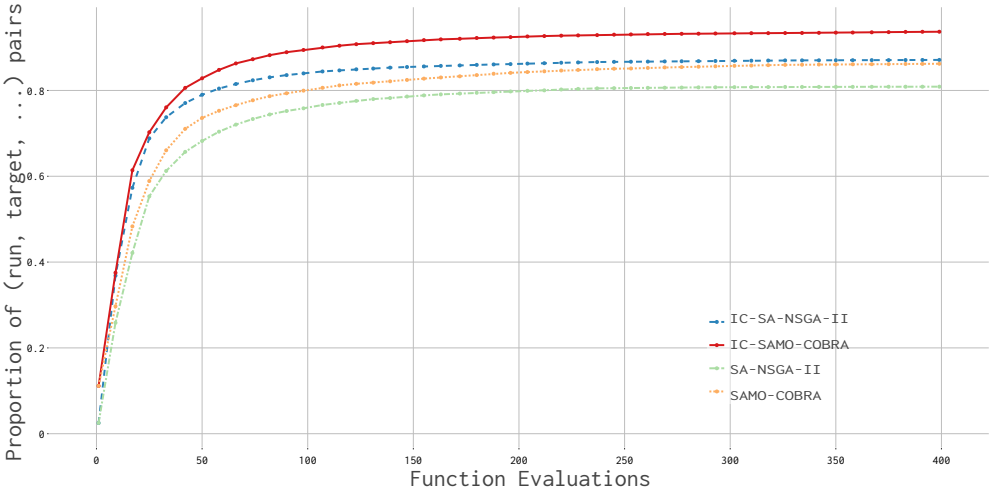


Figure 5.7: Empirical Cumulative Distribution Functions of hypervolume performance metric for SA-NSGA-II, IC-SA-NSGA-II, SAMO-COBRA and IC-SAMO-COBRA. All experiments with different numbers of candidate solutions per iteration and on different test functions are aggregated.

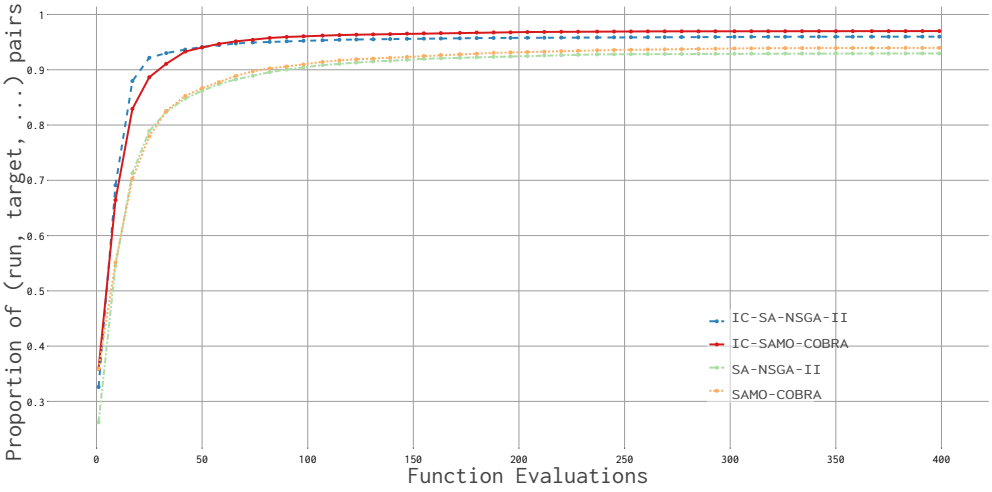


Figure 5.8: Empirical Cumulative Distribution Functions of IGD+ performance metric for SA-NSGA-II, IC-SA-NSGA-II, SAMO-COBRA and IC-SAMO-COBRA. All experiments with different numbers of candidate solutions per iteration and on different test functions are aggregated.

Visual Comparison

The Pareto fronts obtained by the IC-SA-NSGA-II, and the IOC-SAMO-COBRA algorithm can be visually compared with the Empirical Attainment Difference Functions [96]. In Figure 5.9 the EAF difference plot on the TBTD test function is given as an example, with all results per algorithm aggregated. The dark areas mark where the two algorithms obtained different results. As can be seen, the IOC-SAMO-COBRA algorithm manages to find the minimum values of objective 2 on the Pareto frontier, while IC-SA-NSGA-II found smaller values for objective 1. The EAF plots of other two objective test functions can be found in Appendix A.1.

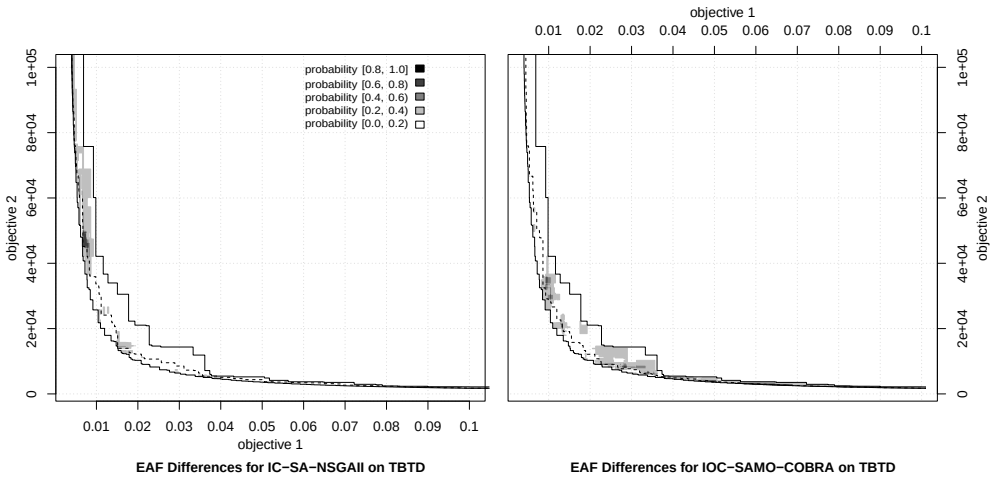


Figure 5.9: Visualization of the Empirical Attainment Function differences between IC-SA-NSGA-II and the IOC-SAMO-COBRA algorithm on the TBTD problem. The solid, dashed and solid lines from left to right represent the best, median and worst found Pareto frontier of both algorithms combined. The grey level in the plots encodes the probability that the corresponding algorithm outperforms the other algorithm in that region. In objective 1, IC-SA-NSGA-II finds smaller values while in objective 2, IOC-SAMO-COBRA finds smaller values. This can be seen in the plot at the bottom of the Pareto frontier at the right plot, IOC-SAMO-COBRA has a higher probability of domination compared to the left side of the Pareto frontier IC-SA-NSGA-II has a higher probability of domination.

5.3.5 Discussion of inexpensive function use

Table 5.9 and Table 5.10 show that IOC-SAMO-COBRA performs better in most cases compared to the other algorithms. The ECDF plot (Figure 5.7) also shows that the algorithm on average also finds good solutions faster since it is able to reach a higher

portion of the run target pairs. The EAF different plots from appendix A.1 also show in most cases that the IOC-SAMO-COBRA finds solutions closer to the Pareto frontier compared to the IC-SA-NSGA-II algorithm. However, two things become apparent when all results are analyzed in more detail.

- IC-SA-NSGA-II significantly outperforms the IOC-SAMO-COBRA algorithm on the BICOP1 and MW2 test problems. BICOP1 and MW2 do not have any active constraints on the Pareto front and the difference between the performances of the algorithms becomes even larger when the number of candidate solutions per iteration increases. This indicates that IC-SA-NSGA-II has more difficulty finding feasible solutions on the Pareto fronts with active constraints and IOC-SAMO-COBRA is directed too much towards the constraint boundaries and has more difficulty finding the Pareto front if the Pareto front is unconstraint.

- For test problems with a very low feasibility ratio (MW1, MW2, MW3, and MW11) the IC-SA-NSGA-II and IOC-SAMO-COBRA significantly outperform their original counterparts where the constraint functions are not directly used in the algorithm. In a few algorithm runs on the MW test functions not a single feasible solution was found. This indicates that the more strict and complex the constraints are, the more beneficial it is to directly use the constraints instead of attempting to learn them with surrogates.

5.3.6 Conclusion and Future Work on Inexpensive Function Exploitation

Measured in terms of HV and IGD+, the IOC-SAMO-COBRA algorithm outperforms the only real competitor IC-SA-NSGA-II in 78% of the benchmark problems. The key algorithmic components that are expected to be responsible for this advantage include:

1. It is beneficial to compute 12 RBF configurations for each expensive objective/constraint and pick the best as a surrogate model for the respective objective/constraint.

2. The use of COBYLA repeatedly and in parallel to find p solution candidates that maximize their joint HV contribution.

For two test functions (BICOP1 and MW2) that do not have any active constraints on the Pareto front, IC-SA-NSGA-II has outperformed IOC-SAMO-COBRA.

Further research is required to improve the IOC-SAMO-COBRA algorithm to be able to quickly find Pareto fronts not subject to active constraints. A crossover of the IOC-SAMO-COBRA algorithm and the IC-SA-NSGA-II algorithm could also potentially lead to even better results.

5.4 Overall Conclusions and Future Work

In this chapter, the following research question is answered: *How to find the Pareto frontier of computationally expensive problems?* Feasible Pareto efficient solutions can be found with the multi-objective SAMO-COBRA algorithm. Two extensions have been made to make the algorithm even more time-efficient. This is done by integrating a multi-point infill criterion, and an extension is made so that it can deal with a mix of expensive and inexpensive objectives and constraints. The resulting new IOC-SAMO-COBRA algorithm has been compared to other state-of-the-art algorithms. By testing on a diverse set of test functions, it has been shown that SAMO-COBRA by itself is fast, very efficient in terms of required function evaluations, and can find well-spread solutions along the Pareto frontier. Integration of the Multi-point infill criteria showed that more evaluations are required to find similar Pareto frontiers. However, a lot of iterations (and therefore in real-world scenarios with expensive function evaluations wall clock time) can be saved. Finally, exploiting the inexpensiveness of constraint functions was shown to be very beneficial since this will, in the majority of cases, lead to better Pareto front approximations.

A final open issue is handling mixed-integer decision parameters, as the extension to such design spaces is crucial for some real-world applications. Extending the IOC-SAMO-COBRA with the mixed-integer decision parameters is possible by introducing different surrogate modeling techniques and by replacing the COBYLA algorithm with a different optimization algorithm that can deal with mixed-integer decision parameters.

Chapter 6

Real World Applications

In this Chapter the last sub-question: *What is the performance of the proposed algorithms in real-world scenarios?* is answered. This is done by solving five real-world simulation-based ship design optimization problems using optimization algorithms. This is done to verify algorithm performance in real-world scenarios and with the algorithms that were readily available at the time the designs had to be optimized at C-Job Naval Architects. The first three problems are computationally expensive multi-objective problems with constraints that can be solved with the algorithms discussed in the previous chapters. The last two real-world optimization problems in this chapter required modifications to the optimization algorithm as these problems had one objective and the computational difference between the objective evaluation and the constraint evaluation was much larger compared to the other problems.

6.1 Trailing Suction Hopper Dredger

The first problem solved with the constraint multi-objective optimization algorithm is the Trailing Suction Hopper dredger as described in Chapter 3 and displayed in Figure 3.3. To repeat the optimization problem briefly, the problem is a two-objective problem where the steel weight and the resistance at operating speed should be minimized. The problem has 11 constraints concerning room and tank capacities, draft, trim, heel, the forepeak bulkhead, and intact stability criteria. All these constraints and objectives are evaluated in the commercial NAPA software.

The original design has a resistance coefficient of 1.08 and a steel weight of 2039 tonnes. This original design is parameterized and optimized by the ACD framework

from Chapter 3 using 200 ship design evaluations proposed by the CEGO algorithm from Section 5.1.1. The framework is allowed to do 200 evaluations to evaluate the trailing suction hopper dredger. The reference point is fixed and set to $[5000, 2]$. By setting the reference point to $[5000, 2]$ the algorithm is limited to solutions with a smaller steel weight than 5000 tonnes,and a resistance coefficient smaller than 2. The experiment has been repeated five times independently with different initial starting points to check for consistency.

In a small first experiment, the severity of the constraints is investigated. In this experiment, 200 random design variations are generated and evaluated. 24% of these design variations turned out to be feasible.

6.1.1 Results of Trailing Suction Hopper Dredger Design Experiment

The results of the five independent runs were very similar. The hypervolume metric as used in multi-objective optimization between the reference point and the Pareto optimal set was on average 3819 and the standard deviation of this volume was 3.3. The parameter combinations of a typical run of 200 evaluated design variations are displayed in the parallel coordinate plot in Figure 6.1. The red lines represent the obtained infeasible solutions, the blue lines represent the feasible solutions, and the green lines represent the Pareto optimal solutions.

The Pareto optimal results of a typical run are presented in Figure 6.2. During this run, the CEGO algorithm in combination with the ACD framework found a set of 10 non-dominated design variations where the most interesting solution has a resistance coefficient of 0.87 and a steel weight of 1748 tonnes. Therefore, compared to the original design, the improved design has a 19% smaller resistance coefficient and 14% less steel weight.

6.1.2 Analysis of the TSHD Results

In Figure 6.3a and Figure 6.3b the original and the improved design are shown respectively. From the first observation, there is not a lot of difference, but the optimized result is 9 meters longer, and 50 centimeters less wide. Typically when a ship is longer more steel is needed to fulfill the strength requirements. But in this case, the hopper is higher which eases the imposed longitudinal bending moment on the ship. The extra strength results in less thick required steel plates and smaller profiles required to meet the longitudinal strength.

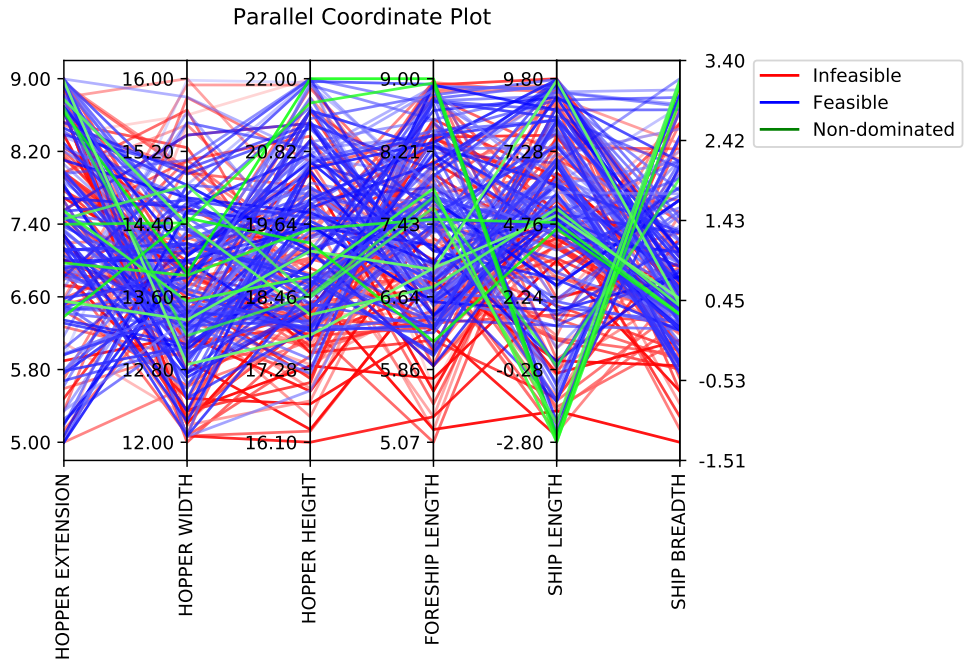


Figure 6.1: Parallel coordinate plot of the 200 different design variations. The red lines represent infeasible solutions, blue lines represent feasible solutions, green lines represent Pareto optimal solutions.

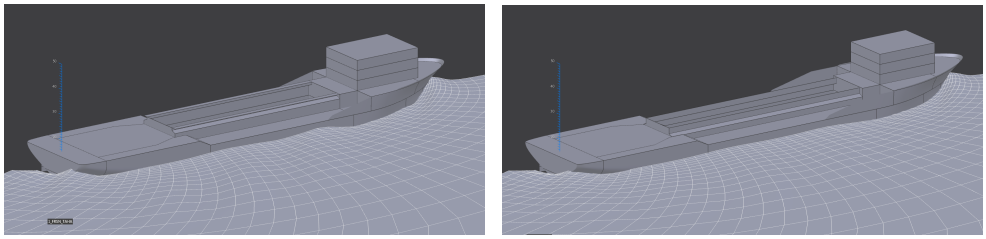
Furthermore, because the vessel is longer and less wide, the resistance also significantly decreased. This can also be seen from the wave pattern around the two design variations in the Figures 6.3a and Figure 6.3b.

6.1.3 Conclusion from Trailing Suction Hopper Dredger Study

From the results, it can be concluded that the Accelerated Concept Design framework is capable of optimizing parameterized vessels in a fully automated manner and in a very efficient way. On top of this, it is shown that this design process can reduce time and human effort while significantly improving ship designs. Furthermore, because of the use of surrogate-assisted models, and therefore objective and constrained prediction, the whole design space can be explored which would never have been an option for a human expert alone.

For future work, more software tools have to be automated and coupled to the Accelerated Concept Design framework to gather more high fidelity results. On top

Figure 6.2: Obtained non-dominated design variations and original design. The marked solutions are the original solution and the most interesting optimized solution.



(a) Original Trailing Suction Hopper Dredger design optimized by human experts.

(b) Trailing Suction Hopper Dredger design optimized with the CEGO algorithm.

Figure 6.3: Comparison of the two Trailing Suction Hopper Dredger designs

of this, a more in-depth analysis of ship parameterization should be done to achieve even better results.

6.2 Wind Feeder Vessel

One of the advancements of the accelerated concept design framework after the TSHD optimization problem is the implementation of the Operability Robustness Index to optimize a Wind Feeder vessel. The SAMO-COBRA algorithm has been used in practice to design a wind feeder vessel to support the installation of windmills at sea. Although high winds are good for power production, they usually also result in rough

seas. These rough seas around the wind park installation sites increase the demand for reliable vessels. The impression of such a wind feeder vessel is presented in Figure 6.4. This vessel has been designed and later optimized at C-Job Naval Architects [30]. This vessel is specifically designed to support the construction of wind farms and to transport the materials from the shore to the installation sites for the US market.

Figure 6.4: Impression of the Wind Feeder Vessel design by C-Job Naval Architects.

The objectives of the optimization case of the wind feeder vessel are to have a robust seakeeping performance to maximize the year-round operability, while also keeping the operational cost and capital expenses at a minimum. The operability can be optimized by maximizing the so-called Operability Robustness Index (ORI)[71]. The ORI objective takes the area of operation into account and therefore can be optimized for a certain wave spectrum. In this case the Pierson Moskowitz spectrum is used as recommended by the DNV-GL maritime classification bureau [52]. The seakeeping assessment is done with a strip theory code of NAPA[1]. Strip theory is proven to be fast and reliable with sufficient accuracy for conventional hull forms [16, 69]. The capital expenses can be translated into the cost of steel that is required to build the vessel, this is roughly equal to the Lightship Weight (LSW) of the vessel. The LSW is calculated by summing the weight of all the equipment plus the minimum amount of steel that is required to fulfill the longitudinal strength requirements. The operational

[1]Intelligent solutions for the maritime industry, https://www.napa.fi/

expenses can be dealt with by minimizing the ship resistance in the water at service speed (Rt[kN]). This resistance is calculated with the Holtrop & Mennen method [79].

More details about the wind feeder design study can be found in the paper and master thesis from Bronkhorst et al. [27, 28].

All objectives, practical constraints, relevant rules, regulations, and loading conditions can be evaluated with the modular Accelerated Concept Design framework as described in Chapter 3. In this software, a parametric 3D model of the ship is set up by a naval architect after which automated software tools can evaluate any design variation in one function call. In the wind feeder vessel case, five design parameters are defined: *Aftship Length*, *Midship Length*, *Foreship length*, *Beam at Waterline*, and *Draught*. Since sea-keeping and longitudinal strength are already captured in the objectives, only two constraints are needed. The two constraints are for space reservation of the wind turbine blades and the meta-centric height for intact stability of the vessel.

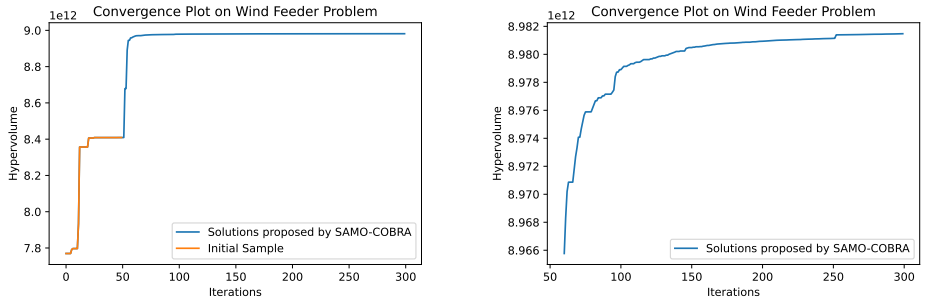
SAMO-COBRA from Section 5.1.2 is then used to optimize this ship design optimization problem. To enhance the exploration in this case study, SAMO-COBRA started with more than the advised 50 initial Halton samples. After evaluation of the initial sample, the SAMO-COBRA algorithm with the PHV infill criterion is used to propose 250 more solutions. On a desktop with an Intel Xeon Processor E3-1245 V3 quad-core processor with 16 GB of working memory, the 300 evaluations required three and a half hours of wall clock time.

6.2.1 Results of Wind Feeder Design Experiment

Based on the first 50 Halton samples, 36% of the design space is estimated to be feasible. Out of the total 300 design variants SAMO-COBRA was able to find 154 non dominated solutions, 35 feasible but dominated solutions, and proposed 111 infeasible solutions.

The convergence of the SAMO-COBRA optimization run is visualized in Figure 6.5a. This plot shows that after the Design of Experiments, SAMO-COBRA quickly finds the most promising solutions. When zooming in on the solutions after evaluation 60 (see Figure 6.5b) it can also be observed that the algorithm continues finding solutions that contribute hypervolume.

The Pareto frontier found on this problem is plotted in Figure 6.6. In this plot, it can be observed that the solutions on the Pareto frontier are nicely spread.



(a) Convergence Plot on Wind Feeder Problem.

(b) Convergence Plot on Wind Feeder Problem zoomed in on the last 240 iterations.

Figure 6.5: Wind feeder optimization process convergence plot (a) and zoomed in part (b).

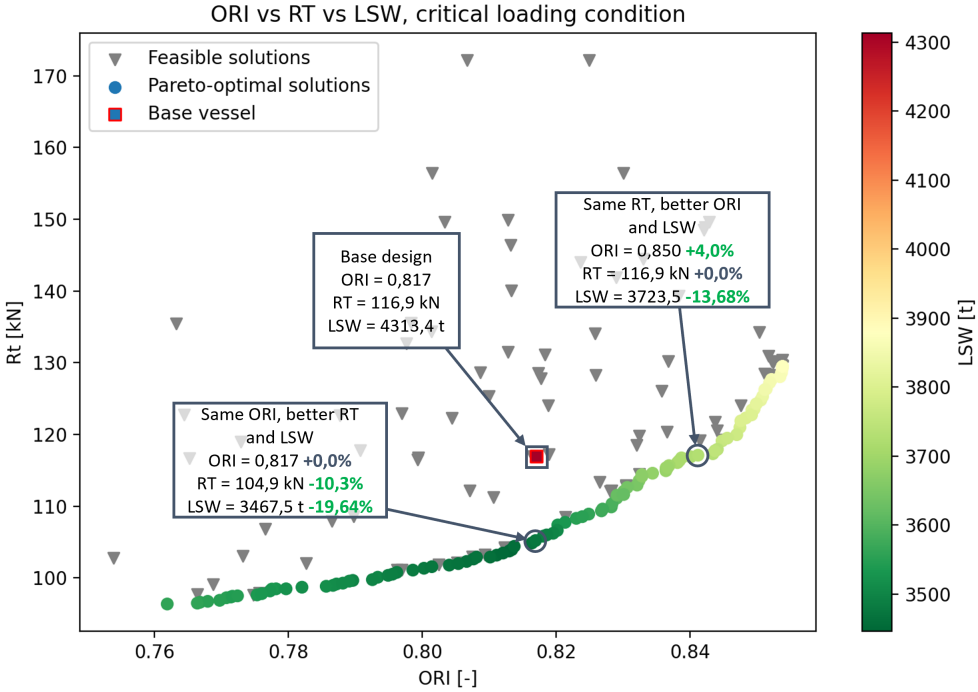


Figure 6.6: Pareto Frontier of Ship Design case with Original Design by human expert represented by a square. Objectives are maximize the Operability Robustness Index (ORI[-]), minimize ship resistance (Rt [kN]), and minimize Lightship Weight (LSW[t]).

6.2.2 Analysis of Wind Feeder Optimization Results

All evaluated, feasible solutions are visualized on the Pareto frontier in Figure 6.6. After analysing the results from the optimization study, the base design by the naval architect was shown to be much too large, causing the ship to be too heavy with a sub-optimal performance. When a few of the Pareto-optimal solutions are compared to the original, then the solution with the same ORI score has a 10.3% smaller resistance value, and 19.64% less light ship weight. The solution with the same resistance score has a 4% better ORI score, and 13.68% less light ship weight. The lightship weight reduction can be explained with that a significant reduction in length was possible.

6.2.3 Conclusion for Wind Feeder Vessel

SAMO-COBRA has been used in practice on a wind feeder optimization problem with three objectives, two constraints, and five decision variables. In this application, the algorithm demonstrates its ability to outperform the human expert in all objectives simultaneously mainly due to the fact that the original design was too large and could have been designed much smaller. The larger design did not contribute to a higher operability robustness index in the area of interest nor was it good for steel weight and resistance. However, a significant amount of wall-clock time would have been able to be saved if at the time of experimenting the IOC-SAMO-COBRA algorithm would have been available since a few of the constraints and the objectives are computationally inexpensive to evaluate.

6.3 Single Hold Cargo Ship Damage Stability Optimization

As a real-world application, the mid-ship section of a single-hold general cargo vessel design[2] as presented in Figure 6.7 is optimized for two conflicting objectives: stability (↑ max) after potential damage (survivability), and cargo hold capacity (↑ max).

Besides the conflicting objectives, the problem has three volumetric constraints and one regulatory constraint:

- Volumetric: The two fuel tanks should be of sufficient size so that enough fuel can be stored and the technical space should be large enough to host the equipment.

[2]Figure courtesy of C-Job Naval Architects, Hoofddorp, Netherlands.

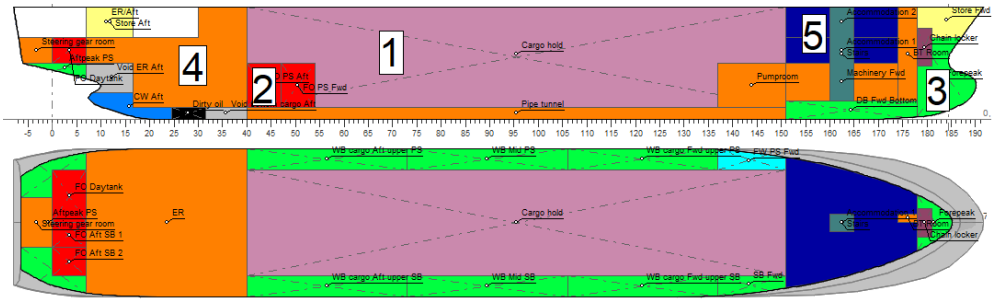


Figure 6.7: Longitudinal section and top view of cargo vessel design optimized for damage stability (survivability) and cargo hold capacity (both to be maximized). The pink compartments (1) annotates the cargo hold, the red compartments (2) are fuel tanks, the green compartments (3) are water ballast tanks, the orange compartments (4) are the technical spaces, and the blue compartments (5) are part of the crew accommodation.

- Regulatory: The attained damage stability index (survivability) score should be larger compared to the required damage stability index.

The objectives and constraints depend on 17 geometric parameters, which influence the longitudinal and transversal positioning of the bulkheads and the heights of openings. The bulkheads split the different compartments and tanks together with the height of decks and openings in the vicinity of the cargo hold.

The evaluation of the damage stability (survivability) objective and the corresponding comparison between the required damage stability constraint is computationally expensive. Evaluation of the damage stability index requires a run of the commercial maritime simulator Delftship pro[3]. The volumetric objective and the three volumetric constraints are inexpensive to evaluate. This offers the opportunity to optimize the design problem with the IOC-SAMO-COBRA algorithm from Section 5.3. The inexpensive constraints and objective are directly used in the IOC-SAMO-COBRA algorithm while for the expensive objective and constraint, RBF surrogates are updated and selected every iteration. More details about the ship design problem are given in [103, 102, 146].

This real-world problem is optimized in three different ways:

1. IOC-SAMO-COBRA with number of candidate solutions per iteration $p = 1$ and 300 function evaluations.

2. IOC-SAMO-COBRA with number of candidate solutions per iteration $p = 3$ and

[3]Version 14.20.343; see Delftship: Visual hull modeling and stability analysis. `https://www.delftship.net/`

501 function evaluations.

3. SA-NSGA-II with number of candidate solutions per iteration $p = 3$ and 501 function evaluations.

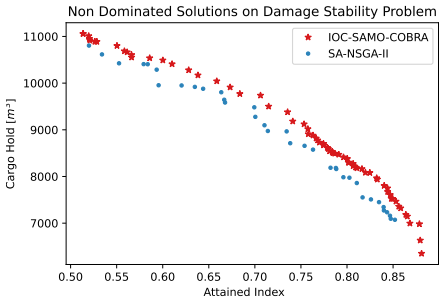
The experiments with $p = 3$ aim at investigating the potential benefit of parallelism in terms of wall-clock time provided that the corresponding number of simulator licenses is available. The HV metric is used to compare the performance of SA-NSGA-II and IOC-SAMO-COBRA. IC-SA-NSGA-II is not experimented with since one of the constraints is expensive to evaluate, and IC-SA-NSGA-II does not have a built-in option to use RBFs for that constraint.

6.3.1 Results of Cargo Vessel Design Experiment

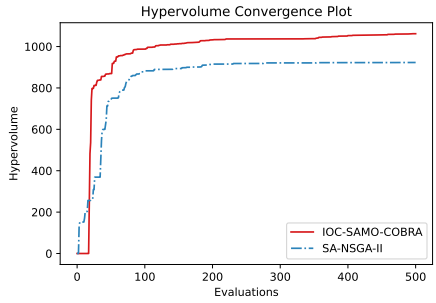
For the expensive objective (damage stability), requiring a Delftship pro simulator run, the median evaluation time was 248 seconds. In experiment 1 (IOC-SAMO-COBRA, $p = 1$, 300 evaluations), a HV of 9115 with respect to the reference point $(0, 0)$ was obtained. In experiment 2 (IOC-SAMO-COBRA, $p = 3$, 501 evaluations), the same HV was obtained in the 129^{th} iteration (after 385 function evaluations), saving a total wall-clock time of 682 minutes compared to experiment 1.

A comparison of the Pareto fronts resulting from experiments 2 and 3 (i.e., a direct comparison between IOC-SAMO-COBRA and SA-NSGA-II) is shown in Figure 6.8a, where cargo hold capacity ($\uparrow$ max) is shown on the y-axis and the attained damage stability index ($\uparrow$ max) on the x-axis. The Pareto front obtained by the SA-NSGA-II algorithm is dominated by the obtained Pareto front obtained by IOC-SAMO-COBRA, and the latter algorithm also finds more extreme solutions (especially for damage stability).

Figure 6.8b illustrates the convergence of the algorithms by showing the HV (measured between the Pareto fronts obtained by the two algorithms and the approximated Nadir point) over the number of function evaluations. The difference in the two Pareto fronts (Figure 6.8a) is also clearly visible in this illustration. The different behavior in the first few evaluations can be explained by the difference in the initial sampling strategies (Latin Hypercube Sampling [138] for SA-NSGA-II vs. Halton Sampling [73] for IOC-SAMO-COBRA).



(a) Comparison of Pareto fronts obtained by IOC-SAMO-COBRA and SA-NSGA-II ($p = 3$, each) on the ship design problem. x-axis: Survivability index; y-axis: Cargo hold (both $\uparrow$ max).



(b) Comparison of the hypervolume maximization progress between IOC-SAMO-COBRA and SA-NSGA-II ($p = 3$, each) on the ship design problem. x-axis: Number of function evaluations; y-axis: Hypervolume.

Figure 6.8: Obtained Pareto Frontier and Convergence plot of single hold cargo optimization problem.

6.3.2 Analysis of Cargo Vessel Design Results

The IOC-SAMO-COBRA results were further analyzed by naval architects to understand and interpret them in the light of vessel design expertise, resulting in the following observations:

- For every point on the Pareto front, the parameter that defines the tanktop height has converged to the minimum value. The algorithm learned that extra height in the double bottom of the vessel does not improve the damage stability index. The compartments above the tanktop benefited from this in terms of their size. Interestingly, this finding could be confirmed since it is also prescribed in the International Convention for the Safety of Life at Sea (SOLAS chapter II-1 part B-2 regulation 9) [81].

- The algorithm also found that a large space between the hull and cargo hold is beneficial for the damage stability criterion. This result can be explained well by the fact that a small distance between the hull and cargo hold makes it less likely for the design to survive in case of damage (flooding of the cargo hold will always lead to the loss of a single cargo hold vessel).

6.3.3 Conclusion Cargo Vessel

The single-hold cargo vessel has been optimized with the IOC-SAMO-COBRA algorithm. By using the IOC-SAMO-COBRA algorithm the available resources (3 available Delftship licences, 3 desktops) are perfectly exploited. The constraints and objective evaluations that where computationally very inexpensive have been used directly in the algorithm instead of also training an RBF. With this new functionality, it is illustrated that a significant amount of wall clock time can be saved and much better Pareto frontier approximations could be made.

6.4 Roll-on/Roll-off Ferry Hull Optimization

After an operational data analysis study and a feasibility study for alternative fuels, an initial design is made by C-Job naval Architects for a fully battery-powered Roll-on/-Roll-off (Ro-Ro) ferry with a capacity of 800 passengers [31, 32]. This vessel is designed to sail between the Saronic islands and the port of Piraeus. A render of this C-Job design is presented in Figure 6.9.

Figure 6.9: Render of Saronic Roll-on/Roll-off ferry designed and created by C-Job Naval Architects.

As this design is intended to sail powered on batteries the vessel requires an optimized hull. To accomplish the energy-efficient hull, the hull form is optimized for minimal resistance at design speed. However, three imposed restrictions limit the

search space of the to-be-designed hull:

1. The hull above the waterline is only allowed to be modified marginally.
2. The displacement of the new hull design should be equal to or larger than the original hull displacement to be able to carry the passengers, vehicles, parcels, and all equipment.
3. The Longitudinal Centre of Buoyancy (LCB) should remain within 1% of the original hull to keep a good trim, heel, and intact stability without the need to move heavy components around.

With these conditions, two completely independent experiments are set up. In the first experiment experienced naval architects with hydrodynamic experience manually optimized the hull for minimal resistance. In the second experiment, the vessel is parameterized and optimized with an optimization algorithm. The hull is parameterized below the waterline varying the following seven parameters ($d = 7$):

1. transom height,
2. the transom angle,
3. the start of the midship in the longitudinal direction,
4. the shoulder location in the longitudinal direction,
5. the bulb size, (can be 0 which results in a design without a bulb)
6. the bulb width,
7. and finally the foreship width.

The design variants are parameterized and generated in the Rhino software. After generation the three computationally relatively inexpensive constraints are also calculated in the Rhino software. Finally, the hull is exported to evaluate the computationally expensive objective in the Star-CCM+ software on a High Performance Computer with 120 cores in the Microsoft Azure cloud. The constraints are computationally relatively inexpensive but still require communication between a software package they can not be called directly in the optimization algorithm as seen in Section 5.3. However, the constraints are computationally much cheaper compared to the objective. Therefore, a variant of the IOC-SAMO-COBRA algorithm is developed that models the objectives and constraints with radial basis function surrogates. Since

we are optimizing a single objective problem, the hypervolume infill criteria of SAMO-COBRA is exchanged for the feasible predicted value infill criteria of the SACOBRA algorithm [13]. The most promising solution according to the infill criteria is evaluated first on the constraints, and only if the design variant is feasible according to the constraints the objective is evaluated. If the constraints are infeasible only the constraint surrogates are updated and the objective surrogate remains the same as in the previous iteration. This process continues until the algorithm has converged and no significant improvements are found for several consecutive iterations. More details and test results for this single objective optimization algorithm framework can be found in Appendix B.1 and in [147].

6.4.1 Results of Ferry Hull Optimization Experiment

After 10 initial samples, the algorithm was allowed to do 110 more evaluations. In the design of experiments, not a single feasible solution is found, however, the first solution proposed by the algorithm was feasible right away. After enough feasible solutions ($d + 1 = 8$) are found to fit a first surrogate model for the objectives, the algorithm started also optimizing the objective score. In total out of the 120 evaluations 21 feasible hulls are found. The best of the 21 feasible hulls had a 26% smaller resistance value compared to the original design.

The convergence plot of the feasible solutions are plotted in Figure 6.10. The convergence plot also shows exactly what was expected. The first 8 feasible designs still show a relatively high objective value as the algorithm is not minimizing the objective score yet but putting its attention to finding feasible solutions. After the algorithm has seen enough solutions to learn from (8 in this case since there are 7 parameters), the algorithm immediately started minimizing the objective value. The overall best hull found by the algorithm is given in Figure 6.11.

6.4.2 Analysis of Ferry Hull Results

As described earlier, besides the experiment with the algorithm, naval architects with hydrodynamic experience also optimized the hull of the ferry. In their experiment, they used their creativity to design the underwater part of the hull in a completely different way. The engineers chose a knuckle line and fitted the bulb under the bow flare instead of in front of the ship. After their choice, the naval architects used a total of 8 CFD evaluations and found a hull with almost identical objective value (a 26% reduction). The final optimized hull of the naval architects is displayed in Figure 6.12.

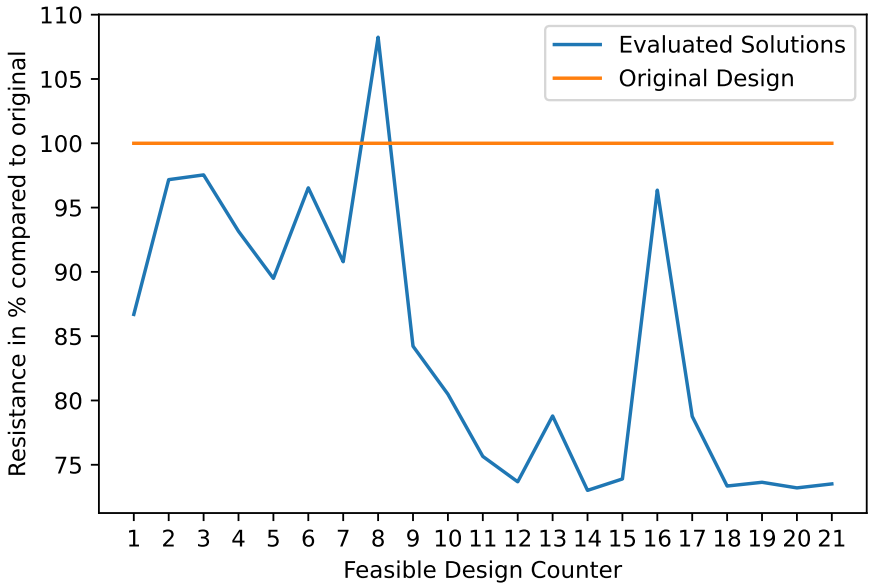


Figure 6.10: Convergence plot of the feasible solutions. Objective score in % compared to the original design.

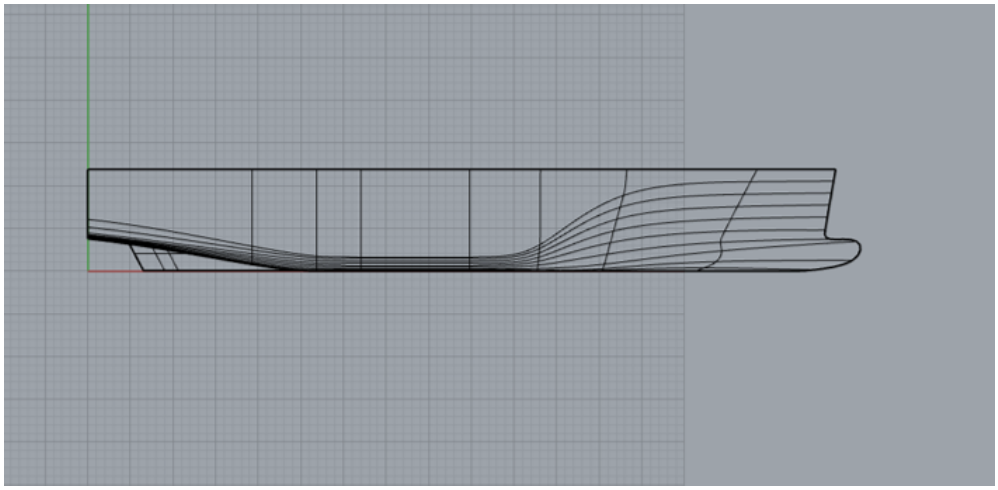


Figure 6.11: Optimized hull found by the optimization algorithm.

When compared to the design proposed by the optimization algorithm, the hull designed by the naval architects is better for a very practical reason. As this is a

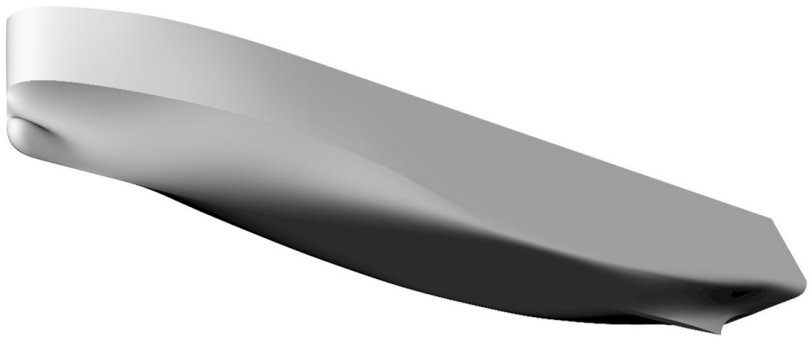


Figure 6.12: Proposed optimized hull design by naval architects.

Ro-Ro ferry, vehicles have to roll on and roll off the ship. When a bulb is in front of the ship, the ramp that the vehicles use to enter and exit the ship should be longer compared to when the bulb is under the bow flare. The downside of the design with the knuckle line is that it might be more complex to build.

6.4.3 Conclusion on Ferry Hull Optimization

The underwater part of a hull from a Ro-Ro ferry is optimized with an optimization algorithm by optimization experts and independently by naval architects with hydrodynamic experience. The best hulls from both the naval architects as the hull from the optimization algorithm showed a 26% smaller resistance compared to the original hull. The hull from the naval architects however had practical benefits compared to the hull from the naval architect. Therefore the hull of the naval architect is preferred above the hull proposed by the optimization algorithm. The conclusion that can be drawn from this is that the parameterization part of any optimization problem is the most important part when optimizing. If not all constraints are captured, the practical application is ignored, or the objective function does not perfectly represent the true goal then optimization algorithms can converge to less ideal solutions. Therefore, human experience remains important and the algorithm can only find as good solutions as the parameterization allows.

6.5 Bulb Optimization Problem

The optimization problem in this section is a refit of an existing container vessel with a container capacity around 10 000 standardized containers. Due to increasing fuel prices and more strict emission regulations, this vessel must reduce its operational speed since sailing at the design speed emits too much greenhouse gas and is way too costly. To sail efficiently at the different operational conditions, the vessel required a bulbous bow (in short bulb) refit. This bulb refit is done by cutting the current bulb off and welding a new optimally shaped bulb back in place. However, optimizing the bulb for all different loading conditions and speeds would lead to four different optimal bulb shapes. Therefore, the goal of this study is to find the bulb that performs well on (weighted)average on all four conditions.

Since the SAMO-COBRA algorithm and other algorithms discussed in this work are designed for multi-objective optimization problems and the problem at hand is a single objective problem, a single objective variant of the SAMO-COBRA algorithm has been introduced to optimize the bulb of a container vessel. The new algorithm variant uses an as small as possible initial Halton sample during the design of experiments, in every consecutive iteration the best RBF transformation strategy and kernel are chosen, and a purely exploiting infill criteria is used to find promising solutions in the adaptive sampling steps. If no improvements have been found for three consecutive iterations, the algorithm switches to an exploring infill criterion. The single objective SAMO-COBRA variant in this way is similar to the traditional Efficient Global Optimization (EGO) algorithm [85] but now with radial basis functions as surrogates and self-adjusting parameters. More details and test results for this single objective optimization algorithm framework can be found in Appendix B.1 and in [147].

The main cutting line to cut the current bulb off is at the design draft and 24 meters from the most forward part of the bulb. This allows us to change 24 meters of the fore hull and only the part below the design waterline. The bulb of the vessel is parameterized in Rhino, by varying 6 parameters that define the bulb length, height, width at two locations, and by changing the overall contour lines of the bulb. The objective is defined by a weighted sum of the required power that is needed to reach four different speeds with different loading conditions and therefore different drafts. Instead of searching for an optimal bulb for each condition, the weighted sum of the four different conditions is chosen as the objective function so that this bulb will perform well in all four conditions. However, this did mean that the complete hull had to be evaluated with RANSE calculations in the Star-CCM+ software for all

four conditions. The mesh for the vessel consists of between 1.7 and 3.7 million cells depending on the speed and draft of the calculation. For this optimization challenge the new algorithm is coupled with a high-performance computer on the Microsoft Azure cloud with 120 CPUs. One RANSE calculation in Star-CCM+ of the complete hull required approximately 20 minutes, since there are 4 different conditions, one iteration required 1 hour and 20 minutes of computation time on the high-performance machine.

6.5.1 Results of Bulb Design Experiment

After a design of experiments of 7 initial Halton Samples, and 38 adaptive sampling steps, the algorithm converged to several similar optimal solutions that were all found on the boundary of the design space. However, because the solutions were found on the boundary of the design space, a second parameterization setup was made that allowed a smaller bulb in terms of height and width. The second optimization run resulted in a bulb that in total for all conditions considered had 4.8% less required power compared to the original design.

In Figure 6.13 the free surface plot is made of the original bulb (left) and the new bulb (right) in one of the four operating conditions. The color indicates the height of the water at that location (Red shows a crest, while blue shows a trough). As can be concluded from the figure, the waves that are generated with the new proposed bulb are especially in the front less extreme compared to the original hull.

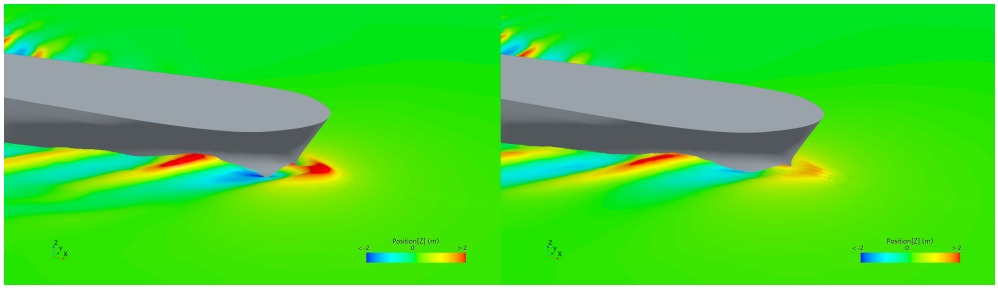


Figure 6.13: Free surface plot of the original bulb (left) and the new bulb (right). The color indicates the height of the water at that location (Red shows a crest, while blue shows a trough).

6.5.2 Analysis of Bulb Design Results

When designing completely new hulls for new build designs the hull surface is modeled with the Rapid Hull Modelling Methodology of Rhino [64]. In the rapid hull modeling methodology, the hull surface of the vessel consists of a loft that is fitted around multiple control lines. However, when refitting a bulb, wrapping a loft around a set of control lines is a bit less straightforward since the loft should exactly fit the current hull and there is only decision freedom after the cutoff line. Therefore some manual fairing was needed to create smooth and continuous hull lines without bumps and irregularities at the transition point between the hull and the bulb. After some manual fairing and checking a few extra operating conditions, the final optimized bulb showed to be able to reduce the power required the most in the slow steaming conditions and when sailing at a limited draught. This could have been expected upfront because the original bulb was designed with the principle of interacting waves for a much higher operating speed and in fully loaded conditions. This principle of interacting waves involves shaping the bulb to minimize wave resistance by strategically managing the interaction between waves generated by the bulb and the hull of the ship. The waves generated by the bulb are intended to cancel the other waves out so that in total, there is less wave resistance. Because of this principle of interacting waves, the original bulb worked best in that one condition (one draft and one speed) it was designed for.

However, When a bulb is optimized for multiple different drafts and different speeds, there is not one bulb design that generates the perfect wave for all different conditions. Therefore, the bulb that was proposed in this study does not cancel out all the waves in all conditions but it is designed to work better on the combined weighted conditions.

6.5.3 Conclusion on Bulb Optimization

To accommodate slower more energy-efficient trips a bulb refit is proposed for a container vessel with a capacity of approximately 10 000 containers. Up front, it was expected that during the optimization process a bulb would be found that generates the perfect wave to accommodate the principle of interacting waves. However, since the operating conditions to be optimized for were so different, it was difficult to find a bulb that cancels out the waves of the hull in all conditions. The newly developed single objective optimization algorithm proposed smaller than expected bulbs. The estimated performance of the small bulbs was better compared to the more traditional larger bulbs according to RANSE calculations executed with STAR-CCM+. The ini-

tial results showed that the search space was too narrow and it should be expanded to allow the algorithm to search for a bulb that was even smaller than initially thought. After the second optimization run, the optimal bulb required some manual fairing since the parameterization didn't align the new bulb perfectly with the already existing hull. Overall the optimization was a success since the hull with the new bulb showed to be 4.8% more efficient in terms of power required compared to the original hull. The largest reduction was found in the operating conditions that deviated the most from the original design condition which had a significantly higher speed.

6.6 Real World Optimization Conclusions and Future Work

Provided with the parameterization, constraint functions, and objective functions, the optimization algorithm variants were able to find feasible and optimal solutions. In the most complex design problems, feasible and optimal solutions are for naval architects often difficult to find as naval architects can't oversee all the interactions between the parameters, constraints, and objectives. The solutions found by the algorithms are however only as good as the parameterization allows, and typically after optimization require some additional practical modifications. In a few cases, it was realized that the parameterization did not lead to the expected result which therefore required a different parameterization setup, a different problem setup, or a change in objective function after which the optimization process is started again. This shows that optimization experts and naval architects should continue to work together to set up optimization problems.

In the future, more research is required in setting up the parameterization of optimization problems as usually the objective and constraint functions are quite clear but the ideal parameter and the parameter ranges that define the outcome are difficult to determine upfront. Setting up the optimization problems can be a labor-intensive and error-sensitive task. So instead of having to parameterize, define objective functions, define constraint functions, and finally optimize, it would be nice to use a generative model that could replace these steps. A start with such an approach is made with ShipHullGAN [86]. However, this thus far can only generate hulls and not complete designs including room arrangements for example.

Another future research direction that could be interesting to investigate is verifying if the proposed algorithms are also effective in other application domains.

Chapter 7

Conclusions and Future Work

In this thesis, research is presented on how constraint multi-objective problems can be optimized with as few function evaluations as possible. This final chapter provides a summary of all previous chapters, followed by an overall conclusion and answer to the main research question. The thesis is finalized by proposing directions for future work.

7.1 Summary

Chapter 1: In the introduction an overview and motivation for the study are provided. The main research question that is answered in this work is:

How to identify the Pareto frontier of constraint multi-objective optimization problems with only a few function evaluations?

The main question is divided into sub-questions that are addressed in the subsequent chapters. The secondary objective is to apply the newly developed algorithms to ship design optimization problems and show their applicability.

Chapter 2: The preliminary chapter presents the formal problem notations, the basic theory of expensive black-box optimization, the benchmark functions used in this work, performance metrics for validation, and visualization techniques for multi-objective optimization. This chapter forms the foundational knowledge and notations that are further developed in the remaining chapters.

Chapter 3: An investigation of ship design optimization problem characteristics is presented in the third chapter. The subquestion *What are typical ship design optimization problem characteristics?* is addressed, providing details about both empirical and simulated design methods. The empirical design method utilizes data from similar vessels for conceptualizing new designs, while the simulated design method is typically employed to create and optimize more detailed versions of ship designs. Alongside these design methods, important ship design software is described, and guidelines for parameterization and optimization problem setup are summarized. Finally, the holistic *accelerated concept design methodology* is introduced that can be used to evaluate ships for multiple key performance indicators at different levels of accuracy.

Chapter 4: The empirical design methodology is elaborated upon in the fourth chapter. A newly proposed empirical design methodology is the *reference finder*, which utilizes machine learning, optimization algorithms, and a dataset with static ship data to identify promising solutions. The reference finder is trained by fitting a random forest regressor to predict key performance indicators and an isolation forest to detect outliers. Finally, the NSGA-II algorithm is coupled with the random forest regressor and isolation forest to discover promising Pareto optimal ship design solutions that do not exist yet but are predicted to be feasible and favorable by the machine learning algorithms. These new preliminary designs can be further developed with the simulation-based design approach as shown in Chapter 6.

Chapter 5: The most significant scientific contribution and the key points of this work are detailed in Chapter 5. In this chapter, the *IOC-SAMO-COBRA* algorithm is introduced, providing the answer to the main research question. The IOC-SAMO-COBRA algorithm adeptly handles constraint multi-objective problems that could have both computationally expensive and inexpensive evaluation functions. It achieves this by fitting surrogates for the computationally expensive functions and directly utilizing the inexpensive functions during the search for promising feasible Pareto optimal solutions. The identification of Pareto optimal solutions is facilitated through the optimization of a multi-point acquisition function capable of proposing one or more feasible solutions per iteration. By proposing multiple solutions per iteration, the computationally expensive evaluations can be run in parallel. With each iteration, the algorithm learns from evaluated solutions by updating surrogates and subsequently continues the search for solutions that maximize the joint hypervolume.

Chapter 6: In the last content-focused chapter of this work, the algorithms introduced in Chapter 5 are used and validated in practice. Five different simulation-based ship design optimization problems are optimized using the accelerated concept design methodology in combination with optimization algorithms. In the first optimization study, the most promising solution from a trailing suction hopper dredger study demonstrated a 19% reduction in resistance and a 14% decrease in steel weight compared to the original design. In the second optimization study, the design of a wind feeder is optimized, revealing a very complete Pareto frontier with improvements in all three objectives: operability, resistance, and lightship weight. The third optimization case focused on optimizing cargo volume and damage stability criteria for a single-hold cargo ship. Here, the power of the multi-point infill criteria (and therefore the possibility of parallel evaluations) and the exploitation of inexpensive functions directly in the algorithm demonstrated significant time savings with the IOC-SAMO-COBRA algorithm compared to traditional approaches. In the final two cases, resistance optimization was conducted for two real-world commercial ship design projects. In the first commercial project, a 26% reduction in resistance was achieved by optimizing the complete hull below the waterline. In the second commercial project, a 4.8% required power reduction was realized by exclusively refitting the bulb of a containership with a capacity of approximately 10,000 containers.

7.2 Conclusions

This study aimed to address the overarching research question: *How to identify the Pareto frontier of constraint multi-objective optimization problems with only a few function evaluations?* The development of innovative algorithms, particularly the *IOC-SAMO-COBRA* algorithm, is a significant scientific contribution that helps in answering this research question. This algorithm demonstrates its effectiveness in handling constraint multi-objective problems, considering both computationally expensive and inexpensive evaluation functions. It does so by iteratively learning and updating surrogates for computationally expensive functions and directly using inexpensive functions when searching for solutions that jointly contribute the most hypervolume.

The practical application of the developed algorithms in real-world ship design optimization problems showcased their impact and flexibility. From reducing the resistance of trailing suction hopper dredgers, ferries, and container ships, to optimizing cargo volume and damage stability in cargo ships, the algorithms consistently demonstrated improvements. Notably, the *IOC-SAMO-COBRA* algorithm's ability to handle paral-

lel simulations and exploit inexpensive functions showcased its efficiency in achieving significant time savings.

In conclusion, this work provided a constraint multi-objective optimization algorithm and the accelerated concept design methodology for ship design that offered valuable insights and practical solutions. The investigation into the research questions, the development of algorithms, and their practical applications have collectively made a substantial contribution to the naval and global multi-objective optimization research fields.

7.3 Future Work

There are many research directions possible to enhance the constraint multi-objective optimization algorithms for computationally demanding problems that are proposed in this thesis. One significant contribution would be to extend the algorithms with functionality that could also deal with discrete, integer, and categorical parameters. This way, computationally expensive mixed-inter constraint multi-objective problems could be solved. Other contributions would be to investigate and extend the limits of the SAMO-COBRA algorithm. For example, what is the limit on the number of objectives, constraints, and parameters that the SAMO-COBRA algorithm can deal with, and does increasing any of these significantly influence the performance? Other directions that require less effort but might improve the performance of the SAMO-COBRA algorithm and its extensions would be an advanced hyperparameter optimization study and validation of the algorithm on different benchmark problems.

From a ship design perspective, setting up the parameters (and their upper and lower limit), constraint functions, and objectives functions correctly before the first run remains challenging. This is problematic, especially when the evaluation functions are computationally demanding. Another open issue is that the parameterization defines the decision freedom and the outcome. Typically, only a small part of the vessel is parameterized and a lot is kept constant which drastically reduces the potential of the optimization process. Therefore, ship design could benefit from more research into generative models to generate feasible optimal solutions and domain-specific optimization algorithms that apply transfer learning to have a warm start in the optimization process.

A final interesting future research direction would be to investigate the applicability of the proposed algorithms in different application domains like aviation, automotive, or civil engineering disciplines.

Appendix A

Appendix

A.1 Empirical Attainment Difference Functions

To visually compare the IOC-SAMO-COBRA and the IC-SA-NSGA-II algorithms, Empirical Attainment Difference Function (EAF) plots are made. The EAF plots of the two-dimensional problems can be found in the 18 Figures. In the EAF difference plots the dark areas mark where the two algorithms obtained different results. The more frequently a certain area is dominated the darker the gray scale is. As can be seen in the majority of the figures (except for BICOP2, MW2 and WB), the IOC-SAMO-COBRA algorithm manages to find solutions that dominate the solutions of the IC-SA-NSGA-II algorithm.

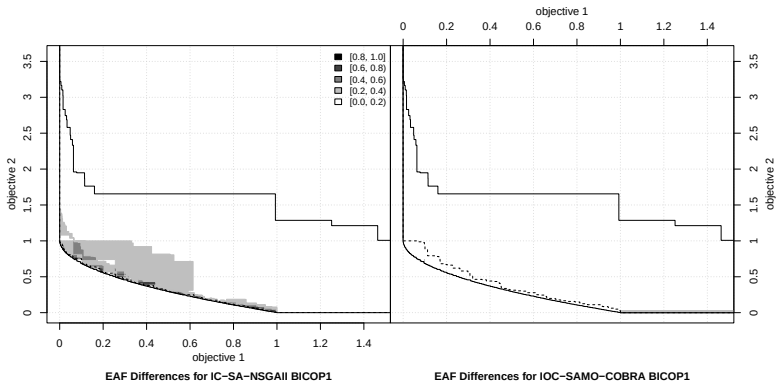


Figure A.1: EAF difference plot BIOCP1

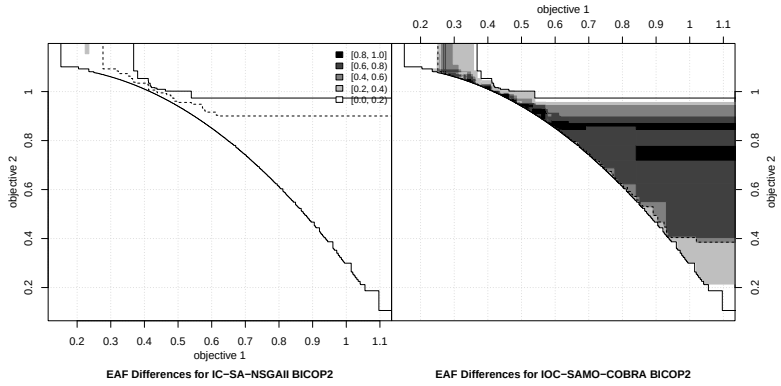


Figure A.2: EAF difference plot BIOCP2

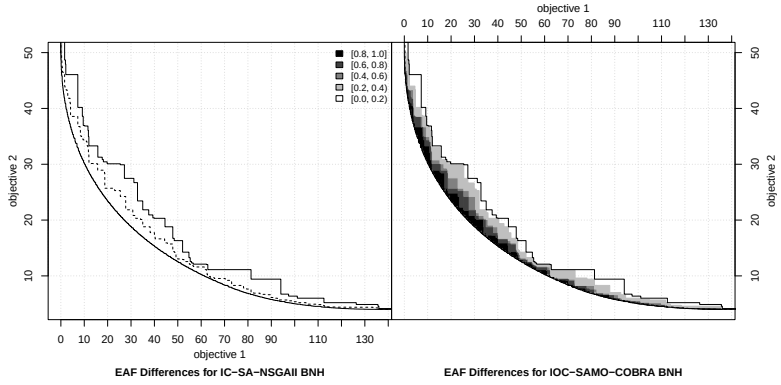


Figure A.3: EAF difference plot BNH

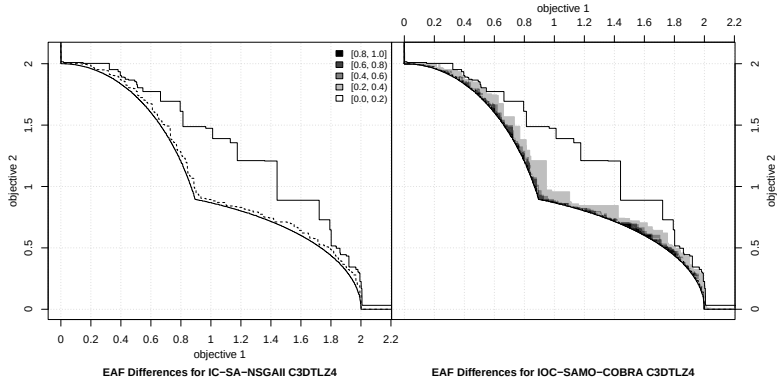


Figure A.4: EAF difference plot C3DTLZ4

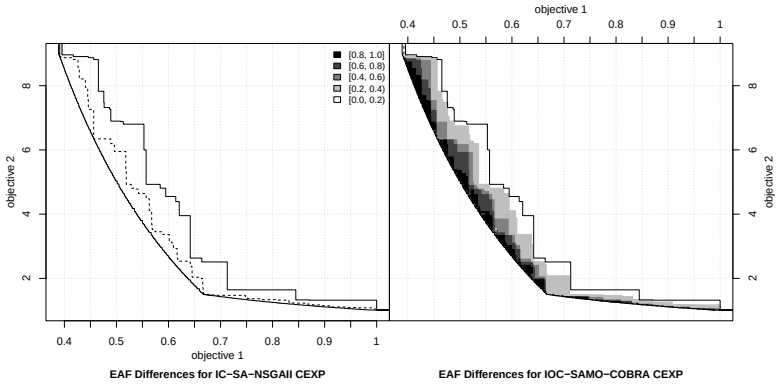


Figure A.5: EAF difference plot CEXP

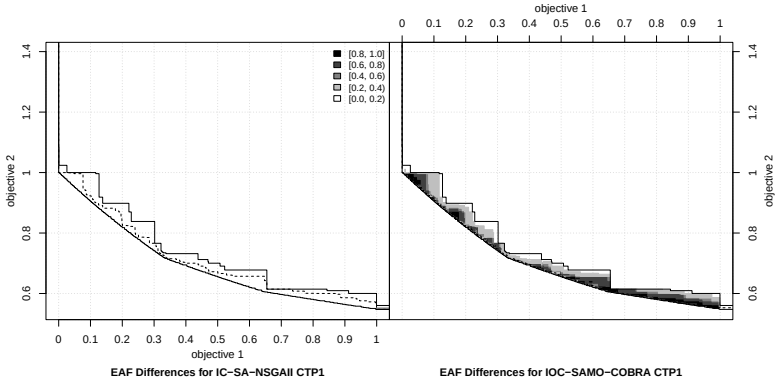


Figure A.6: EAF difference plot CTP1

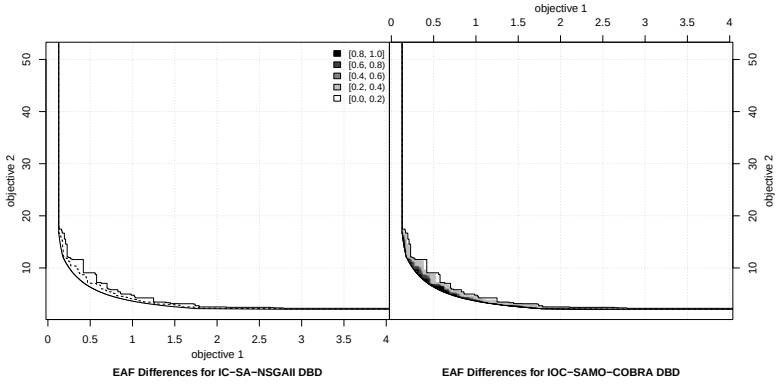


Figure A.7: EAF difference plot DBD

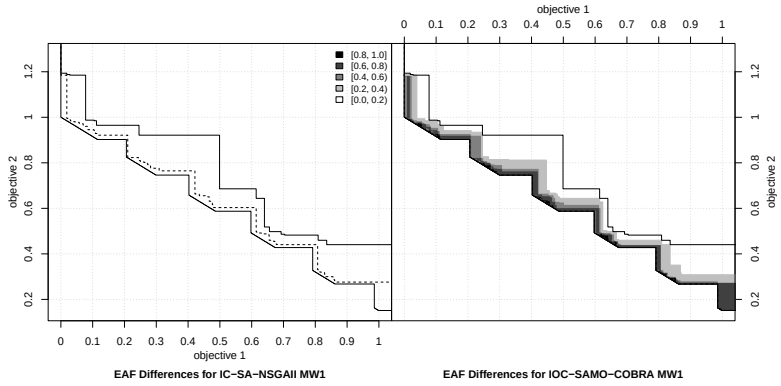


Figure A.8: EAF difference plot MW1

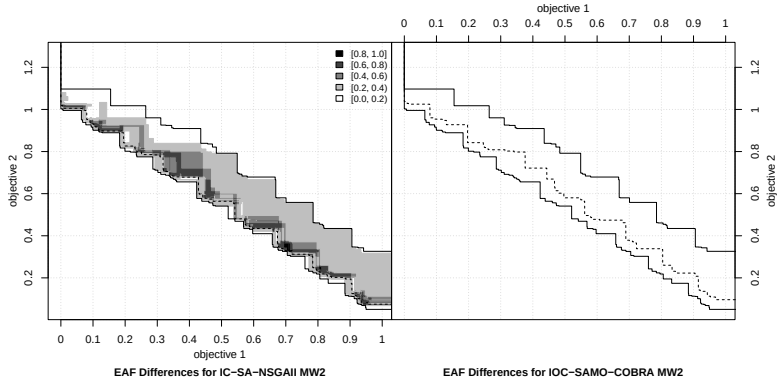


Figure A.9: EAF difference plot MW2

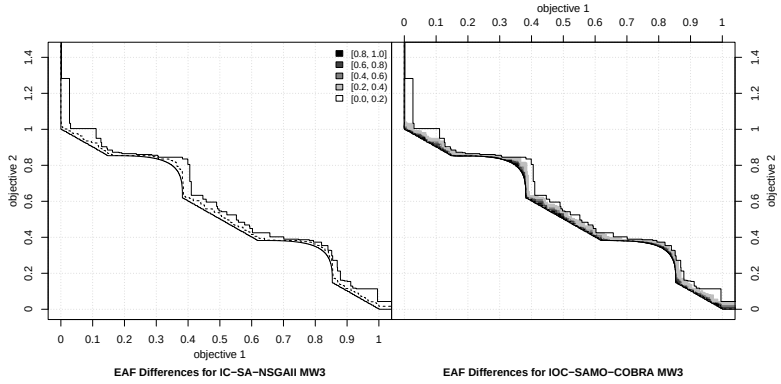


Figure A.10: EAF difference plot MW3

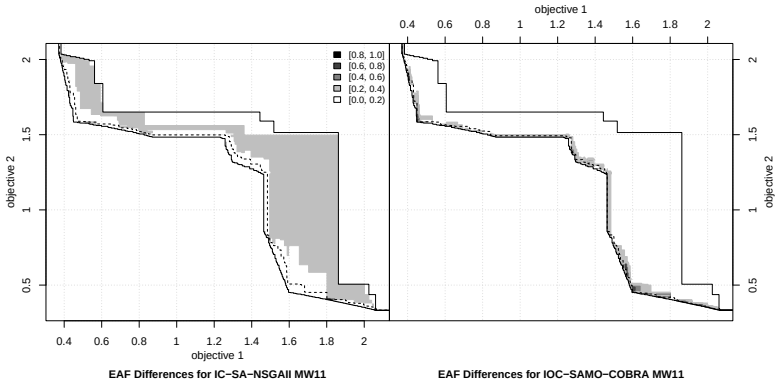


Figure A.11: EAF difference plot MW11

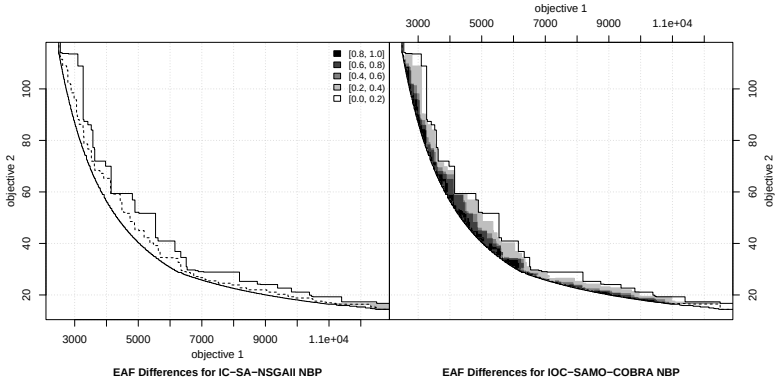


Figure A.12: EAF difference plot NBP

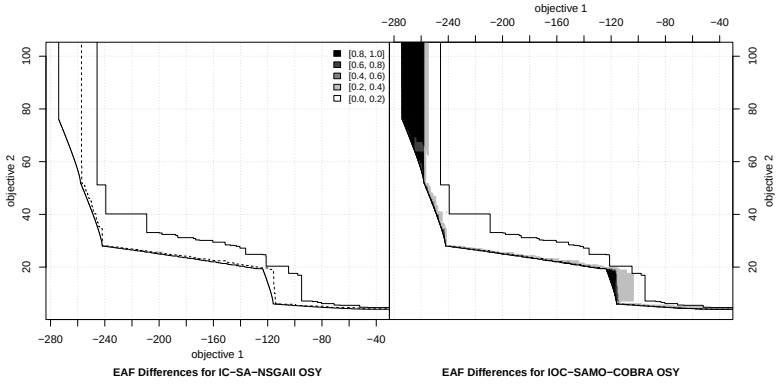


Figure A.13: EAF difference plot OSY

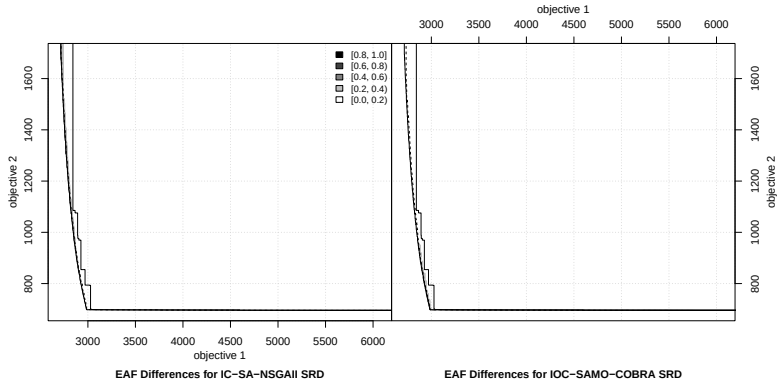


Figure A.14: EAF difference plot SRD

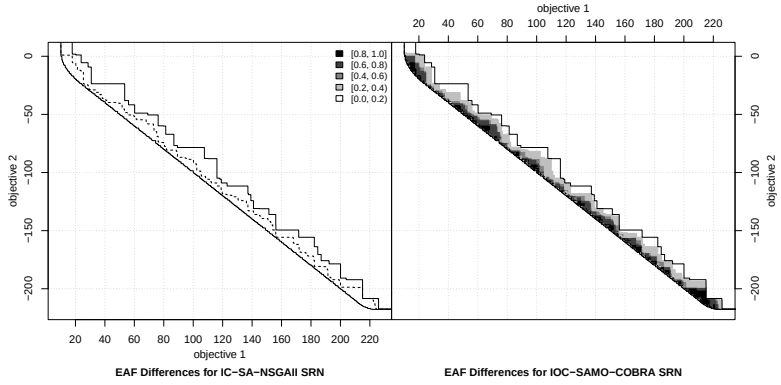


Figure A.15: EAF difference plot SRN

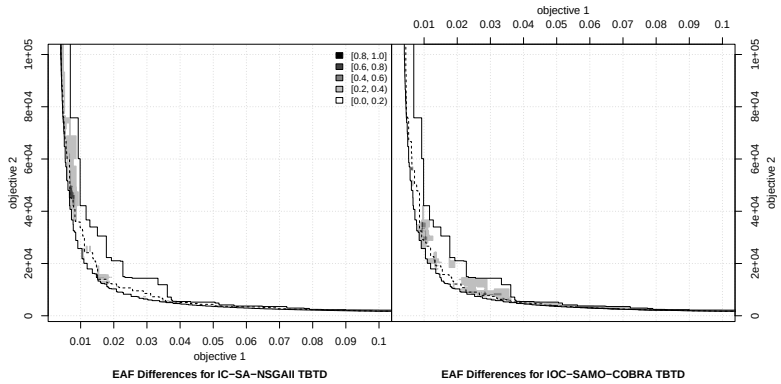


Figure A.16: EAF difference plot TBTD

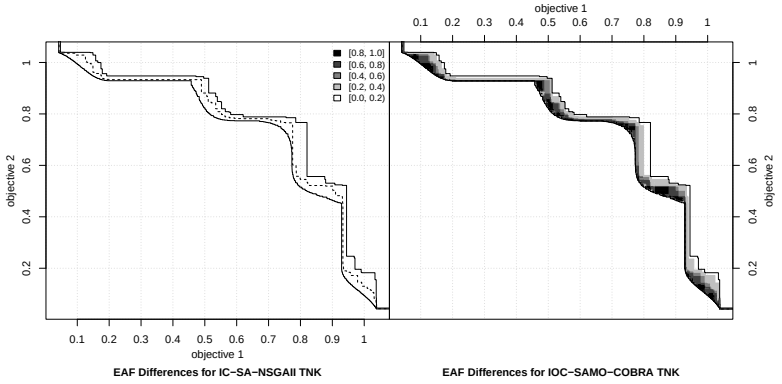


Figure A.17: EAF difference plot TNK

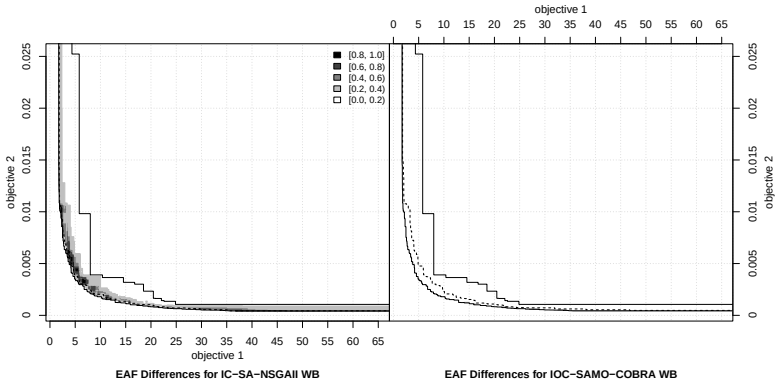


Figure A.18: EAF difference plot WB

Appendix B

Appendix

B.1 Expensive Single Objective Optimization

For the optimization of the two single objective ship design problems of Chapter 6 the Modular Adaptive Global Optimization Framework (MAGOF) is introduced. The objective from the two ship design problems were computationally expensive and the constraints are computationally inexpensive. To be ready for more different problem characteristics a modular adaptive framework is proposed. In this appendix a pseudocode and a detailed explanation is presented together with experiments and results on the well known constraint single objective G-Problem suite [93, 61].

B.2 Modular Optimization Framework

The pseudocode of the Modular Adaptive Global Optimization Framework (MAGOF) is presented in Algorithm 4. The evaluation method and strategy are described in more detail in Algorithm 5 and Section B.2.3. The input, the overall explanation of the pseudocode, and the working of the framework are described in more detail in the following subsections.

B.2.1 Input parameters

The input arguments for the modular framework are:

1. **Objective function** $f(\mathbf{x})$ that is to be minimized. The objective function is defined by the user as expensive $f_e(\mathbf{x})$, or inexpensive $f_c(\mathbf{x})$.

2. **Constraint function(s)** $\mathbf{g}(\mathbf{x})$ that consist out of m separate constraint functions, where $m \geq 0$. The constraint function(s) are either defined by the user as expensive $\mathbf{g}_e(\mathbf{x})$, or inexpensive $\mathbf{g}_c(\mathbf{x})$. The constraint functions return the constraint violation, meaning that constraint values $g(\mathbf{x}) \leq 0$ are defined as feasible.

3. **Input space** $x \in \Omega \subset \mathbb{R}^d$ that is limited by the lower and the upper boundary $[\mathbf{x}_{lb}, \mathbf{x}_{ub}]$.

4. **Initial sample strategy and sample size** N_{init} and DoE define how many samples are evaluated in the design of experiments. This should at least be larger than $d + 1$.

5. **Evaluation budget** N_{max} defines how many expensive function evaluations are allowed to be evaluated.

6. **RBF strategy domain**, $\Phi = \{Cubic, Gaussian, Multiquadric, InverseQuadratic, InverseMultiquadric, ThinPlateSpline\} \times \{\text{PLOG}, standardized\}$. The RBF strategy domain defines the different surrogates that are used in every iteration to model the computationally expensive functions.

7. **Parallelism** p, is the number of solutions that can be evaluated in parallel. Note that parallelism is not a requirement as p can also be 1.

8. **Acquisition function** α that used to find promising solutions. The acquisition function uses the surrogates or the inexpensive functions directly to find p promising solutions for evaluation.

9. **Constraints first indicator** that defines if the constraints should all be satisfied before the objective function can be evaluated.

B.2.2 Design of Experiments

The framework in Algorithm 4 starts in line 2 by creating a Design of Experiments (DoE). The size and the strategy for the DoE can be chosen by the user and can be random, a latin hypercube sample, solutions on the boundaries, or a Halton sample. Each of these sample strategies has its strengths, however, an empirical comparison by Bossek et al. [23] showed that an as small as possible initial Halton sample [73] is in most cases the most efficient strategy. It is also possible to start with an initial sample that is already evaluated.

Algorithm 4: MAGOF.
Input: Objective function $f(\mathbf{x})$, that can be computationally expensive $f_e(\mathbf{x})$ or computationally inexpensive $f_c(\mathbf{x})$, constraint function(s) $\mathbf{g}(\mathbf{x})$, split where required into expensive constraint function(s) $\mathbf{g}_e(\mathbf{x})$, computationally inexpensive constraint function(s) $\mathbf{g}_c(\mathbf{x})$, decision parameters' lower and upper bounds $[\mathbf{x_{lb}}, \mathbf{x_{ub}}] \subset \mathbb{R}^d$, sampling strategy DoE, number of initial samples N_{init}, maximum evaluation budget N_{max}, RBF strategy domain consisting of 12 RBF strategies $\Phi = \{Cubic, Gaussian, Multiquadric, InverseQuadratic, InverseMultiquadric, ThinPlateSpline\} \times \{\text{PLOG}, standardized\}$, number of solutions that can be evaluated in parallel p, acquisition function α, constraint first indicator c_{first}.
Output: Evaluated solutions.

1 **Function** MAGOF(f, $\mathbf{g}$, $\mathbf{x}_{lb}, \mathbf{x}_{ub}$,DoE, N_{init}, N_{max}, Φ, p, α, c_{first}):
2 $\quad \mathbf{x}^* \leftarrow \{\mathbf{x}_1, \cdots, \mathbf{x}_{N_{init}}\} \leftarrow \text{DoE}(\mathbf{x}_{lb}, \mathbf{x}_{ub}, N_{init})$ ▷ Generate DoE, $\mathbf{X} \in \mathbb{R}^{d \times N_{init}}$
3 $\quad \mathbf{F}, \mathbf{G}, \mathbf{X} \leftarrow \text{EVALUATE}(\mathbf{x}^*, f, g, p, c_{first}, N_{init}, \mathbf{F} = [], \mathbf{G} = [], \mathbf{X} = [])$ ▷ Evaluate initial sample and initialize archives $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{X}$
4 $\quad \mathbf{h} \leftarrow \{f_e \cup \mathbf{g}_e\}$ ▷ Union of expensive objective and constraint functions
5 $\quad \varphi^* \leftarrow (\varphi_1, \ldots, \varphi_{|\mathbf{h}|}) \leftarrow (Cubic, standardized)^{|\mathbf{h}|}$ ▷ Initialize RBF strategy for all expensive functions, $\varphi^* \in \Phi$
6 $\quad \mathbf{E}_{i,j} \leftarrow 0 \; \forall (i,j) \in \mathbf{h} \times \Phi$ ▷ Initialize RBF approximation errors for each possible RBF configuration(Φ) for all expensive functions($\mathbf{h}$)
7 $\quad j \leftarrow N_{init}$ ▷ Initialize expensive evaluation counter
8 $\quad$ **while** $j < N_{max}$ **do**
9 $\quad\quad \mathbf{S}^{\Phi} \leftarrow \left(S^{\Phi_1}_{h_1}, \ldots, S^{\Phi_{12}}_{h_{|\mathbf{h}|}}\right) \leftarrow \{\text{FITRBF}(\mathbf{X}, h, \Phi, \mathbf{x_{lb}}, \mathbf{x_{ub}}) \mid \forall h \in \mathbf{h}\}$ ▷ Fit RBFs using all strategies(Φ) for all expensive functions($\mathbf{h}$)
10 $\quad\quad \mathbf{S}^{\varphi^*} \leftarrow \left(S^*_1, \ldots, S^*_{|\mathbf{h}|}\right)$ ▷ Select best RBF strategy based on line 5 or 15
11 $\quad\quad \mathbf{x}^*_1, \ldots, \mathbf{x}^*_p \leftarrow \text{MAX}(\alpha, p, \mathbf{S}^{\varphi^*}, f_c, \mathbf{g}_c)$ ▷ Get p new solutions based on acquisition function α, use cheap functions f_c and $\mathbf{g}_c$ directly
12 $\quad\quad j \leftarrow j + p$ ▷ Increase iteration counter to new matrix sizes
13 $\quad\quad \mathbf{X} \leftarrow [\mathbf{X}, \mathbf{x}^*_1, \ldots, \mathbf{x}^*_p]$ ▷ Add p new solution vectors, $\mathbf{X} \in \mathbb{R}^{d \times j}$
14 $\quad\quad \mathbf{F}, \mathbf{G}, \mathbf{X} \leftarrow \text{EVALUATE}(\mathbf{x}^*, f, g, p, c_{first}, j, \mathbf{F}, \mathbf{G}, \mathbf{X})$ ▷ Evaluate new solutions
15 $\quad\quad \varphi^*, \mathbf{E} \leftarrow \text{SELECTBESTRBFSTRATEGY}(\mathbf{E}, S^{\Phi}, \mathbf{F}, \mathbf{G}, \mathbf{X})$ ▷ Update RBF approximation errors $\mathbf{E}$, and new best RBF configuraiton φ^*
16 $\quad$ **end**
17 **return** ($\mathbf{F}$, $\mathbf{G}$, $\mathbf{X}$)

B.2.3 Evaluation of the solutions

On lines 3 and 14 of Algorithm 4 the solutions that are proposed by the DoE, or after optimizing the acquisition function, are evaluated as described in the evaluate Algorithm 5. The approach is dependent on the inexpensiveness of the constraint and objective functions. There are 3 levels of expensiveness. (1) the function can be evaluated almost instantly and can therefore be evaluated millions of times (it must at least be faster than fitting and interpolating an RBF surrogate model). (2) the function

requires a little bit of evaluation time and can therefore not be evaluated numerous times and evaluating them millions of times is too costly. Function evaluations of level 2 for example require a few seconds up to a few minutes and are significantly less costly compared to the most expensive evaluations. (3) the function is computationally expensive and the evaluation budget is very limited e.g. computational fluid dynamic simulations or finite element analysis that can take up to hours on a cluster to evaluate.

The inexpensive functions level 1 are used directly in the optimization algorithm, see Section B.2.5. If the constraints are inexpensive level 2, they are evaluated first before the computationally expensive functions. Only if the constraints are satisfied, the expensive functions (level 3) are evaluated. If the constraints are violated, a null is stored instead of the expensive outcome. This way, in the next iteration of MAGOF the RBF surrogates for the expensive functions remain the same as in the previous iteration while for the inexpensive functions level 2 the RBF surrogates are updated.

Algorithm 5: Evaluate.
Input: Solutions $\mathbf{x}^*$ to be evaluated, Objective function $f(\mathbf{x})$, that can be computationally expensive $f_e(\mathbf{x})$ or computationally inexpensive $f_c(\mathbf{x})$, constraint function(s) $\mathbf{g}(\mathbf{x})$, split where required into expensive constraint function(s) $\mathbf{g}_e(\mathbf{x})$, computationally inexpensive constraint function(s) $\mathbf{g}_c(\mathbf{x})$, number of solutions that can be evaluated in parallel p, constraint first indicator c_{first}, objective values of evaluated solutions $\mathbf{F}$, constraint values of evaluated solutions $\mathbf{G}$, evaluated solutions $\mathbf{X}$.
Output: Evaluated solutions.

```
Function Evaluate(x*, f, g, p, c_first, j, F, G, X):
    G ← [G, g_c(x*_1), ..., g_c(x*_p)]      ▷ Add vectors of cheap constraints, G ∈ ℝ^{m×j}
    if not c_first then
        ▷ If constraints do not need to be satisfied first then add
        F ← [F, f_e(x*_1), ..., f_e(x*_p)]        ▷ Vector of evaluated objectives, F ∈ ℝ^j
        G ← [G, g_e(x*_1), ..., g_e(x*_p)]        ▷ Vectors of evaluated constr G ∈ ℝ^{m×j}
    else
        for x_i ∈ {x*_1, ..., x*_p} do
            if g_c(x_i) ≤ 0 then
                ▷ If constraints need to be satisfied first
                F ← [F, f_e(x_i)]     ▷ Only add objective value of feasible solutions
                G ← [G, g_e(x_i)]   ▷ Only add constraint value of feasible solutions
            else
                ▷ If constraints are violated
                F ← [F, null]                          ▷ Don't evaluate and add null
                G ← [G, null]                          ▷ Don't evaluate and add null
        end
    return (F, G, X)
```

B.2.4 Radial Basis Functions

In every iteration of MAGOF, surrogates are fitted to approximate the constraint and objective functions (line 9 of Algorithm 4). However, there are many kernel options and scaling techniques available when fitting RBF surrogates and each option can be good for different scenarios. Therefore, RBF surrogates are fitted with the following kernels: *Cubic*, *Gaussian*, *Multiquadric*, *InverseQuadratic*, *InverseMultiquadric*, *ThinPlateSpline* and two different scaling strategies are used to scale the constraint and objective values. The standardization method is used so that the uncertainty quantification method can be used for the RBFs. The Plog transformation from Equation B.1 is selected so that the RBFs can better model steep slopes. For each combination of these kernels and transformation methods, a surrogate is fitted which results in a total 12 RBF surrogate models per expensive function. In every iteration, the RBF strategy with the smallest approximation error is selected (line 5 and 15 of Algorithm 4) and the RBF approximation errors are stored.

$$\text{Plog}(y) = \begin{cases} +\ln(1+y), & \text{if} \quad y \geq 0 \\ -\ln(1-y), & \text{if} \quad y < 0 \end{cases} \tag{B.1}$$

B.2.5 Acquisition Function Optimization

The acquisition functions integrated into the framework are: the expected improvement acquisition function [85], the generalized expected improvement acquisition function [119] for parallel evaluations when $p > 1$, and the purely exploitative acquisition function that predicts the objective value with the RBF surrogate without uncertainty. This acquisition function is optimized with the COBYLA algorithm [120]. COBYLA is a single objective optimization algorithm that optimizes an optimization problem with constraints by linearly approximating the acquisition function and the most violated constraint in a small trust region. COBYLA finds the most promising solution in this trust region, then checks the constraint and objective values, and iteratively adjusts the trust region until the trust region is so small and the local optimum is found.

For the functions with expensiveness levels 2 and 3, COBYLA is instructed to use the surrogate models when optimizing the acquisition function. The inexpensive functions (level 1) that can be calculated instantly, are directly used by COBYLA when optimizing the acquisition function. The usage of the inexpensive functions is beneficial because they don't make approximation errors that surrogate models make.

Note that during the optimization of the acquisition function, the inexpensive functions (or surrogate for expensive functions) are evaluated many times.

Because COBYLA is a local optimizer, the COBYLA algorithm starts searching from multiple random locations. This makes it more likely that the global optimum is found.

B.3 Experiments

For this algorithm, four types of experiments are conducted. All experiments are conducted on the G-problem test suite [93, 61].

B.3.1 G-Problem experimental setup

The G-Problems (G1 to G11 from [93, 61]) are selected as an artificially created benchmark suite to validate the performance of MAGOF. A Python implementation of the G-Problems is taken from the CEC 2006 Special Session on Constrained Real-Parameter Optimization [63]. The G-problems considered have between 1 and 9 constraints and between 2 and 20 decision parameters, and all are to be minimized. The optimal solutions are known for all G-problems, some problems have active constraints at the optimum, while other optima are somewhere in the feasible region. The feasibility ratio of the G-Problems varies between less than 1% feasible and 99% feasible per test problem. More details regarding the G-Problems can be found in e.g. [13, 93, 61]. Evaluation of the constraint and the objective functions of the G-problems are computationally inexpensive. However, for the experiments, it is assumed that the objectives are computationally expensive to evaluate.

To test the functionality of the mixed expensiveness, four different configurations of MAGOF are tested with different infill criteria and different inexpensive function handling techniques.

1 Traditional The "traditional" configuration uses MAGOF without any special treatment and/or separation of expensive versus inexpensive functions. The objective and constraint methods are considered equally expensive which in MAGOF means that in every iteration the RBFs are fitted for the constraints and objective, the best RBF strategy is selected, and then the default acquisition function is optimized. The resulting solution is computed and evaluated with the constraints and the objectives.

2 Constraints First: The "constraints first" configuration of MAGOF utilizes the adjustment in the evaluation strategy as presented in Algorithm 5. In this con-

figuration it is assumed that the constraints evaluations are computationally way less expensive compared to the objective evaluation. In every iteration, the RBFs are fitted, the best RBF strategy is selected, and the default acquisition function is optimized using the surrogates. The solution that is proposed is now evaluated first on the constraints. In case any of the constraints are violated, the objective function is not evaluated and the next iteration starts. In case the constraints apply the objective function is evaluated.

3 Constraints Integrated: The "constraint integrated" configuration in MAGOF is not conventional as instead of fitting RBFs for the constraints, the constraints are directly used when optimizing the acquisition function because it is assumed that the constraint function evaluations are computationally cheaper than fitting an RBF and making predictions with RBFs. The RBFs are now only used to model the objective function since the objective function evaluation is assumed to remain computationally expensive.

4 Parallel: The "parallel" configuration of MAGOF does not assume inexpensive constraints or objectives and therefore uses RBF models to model the assumed expensive constraint and objective functions. After the RBF models are fitted, the best RBF approximation is selected, and the generalized expected improvement acquisition function is optimized. The hyperparameter (g_{EI}) of the acquisition function is set in such a way that one solution proposed by the algorithm is purely exploitative ($g_{EI} = 0$), one is explorative and would be most similar to solutions proposed by the expected improvement acquisition function ($g_{EI} = 6$), and one solution is a balance between exploitative and explorative ($g_{EI} = 3$). This way, the generalized expected improvement acquisition function can be used to propose 3 different solutions. After the solutions are proposed, they are evaluated in parallel with both the objective and constraint functions.

All configurations start with an as small as possible initial Halton sample as a DoE. After the DoE the first 3 configurations are allowed to do a total of $300 - |\text{DoE}|$ iterations for the non-parallel configurations. The configuration that proposes 3 solutions per iteration was allowed to do an additional 100 iterations. This way each algorithm configuration has in theory the possibility to do 300 objective function evaluations. Note that the configuration with constraint first, does not necessarily use all these 300 objective evaluations since in the 300 iterations, this configuration also sometimes proposes infeasible solutions. The constraints integrated configuration uses a lot more constraint function evaluations since when optimizing the acquisition function, the constraints are evaluated many more times.

B.4 Results

In Table B.1 the results are presented for the G-problem test suite. In this table, the mean smallest objective scores of the feasible solutions are presented after 10 independent runs of MAGOF with the four different options. Besides the mean objective score, the number of required function evaluations is reported that was required to reach the minimum. Please refer to [63] for the complete set of minima for all functions. A red cross (x) indicates that the optimum was not found within 300 evaluations.

Function		Traditional	Constraint First	Constraint Integrated	Parallel P=3
G01	fv	−15.00	−15.00	**−15.00**	−13.54
	fe	28	24	**22**	x
G02	fv	−0.304	−0.304	**−0.383**	−0.304
	fe	x	x	x	x
G03	fv	−0.000	**−0.089**	−0.006	−0.000
	fe	x	x	x	x
G04	fv	−30665	−30665	**−30665**	−30665
	fe	26	19	**11**	111
G05	fv	5126.5	5126.5	**5126.5**	5126.5
	fe	41	12	**10**	x
G06	fv	−6957	**−6959**	**−6959**	−6956
	fe	x	x	x	x
G07	fv	24.306	24.306	**24.306**	35.479
	fe	34	22	**21**	x
G08	fv	**−0.096**	−0.096	−0.096	−0.096
	fe	**13**	30	28	100
G09	fv	680.63	1186.8	**680.64**	1597.8
	fe	240	x	**89**	x
G10	fv	7114.3	7088.6	**7049.3**	9233.1
	fe	x	x	**65**	x
G11	fv	0.7500	**0.7500**	**0.7500**	0.7500
	fe	7	**5**	**5**	12

Table B.1: The mean minimum encountered objective score of feasible solutions (fv), and the mean objective function evaluations (fe) required to find the optimal value (a x indicates the known optimal value was not reached in 300 iterations). Four different approaches are compared, the traditional optimization technique, the constraint first approach, the constraints integrated into the acquisition function optimization process, and the approach with the generalized expected improvement acquisition function that proposes 3 solutions in parallel. The best combination of fv and fe are marked in **bold** per G-problem. All G-problems are optimized in 10 independent optimization runs.

Inspection of the results shows that in the majority of the problems using the cheap

constraints directly in the optimization algorithm when optimizing the acquisition function is beneficial in terms of execution time and convergence. The other option where the constraints are first evaluated to check for feasibility before the objective function is evaluated also shows better results compared to the conventional approach where the objective is evaluated together with the constraints. The option to propose 3 solutions in parallel with the generalized expected improvement acquisition function does not show good results. It was expected upfront that when proposing multiple solutions for parallel evaluation (and this way save computation time), the number of iterations of the algorithm could be reduced. However, the number of required iterations and therefore also the number of function evaluations is higher compared to the other approaches.

On the G08 test problem, the traditional approach finds the optimum in less required objective evaluations compared to the other approaches. It is assumed that the reason for this quick convergence is that the information gathered from the evaluated infeasible solutions is of great value for this optimization problem. The information from the infeasible evaluated solutions is missing when the constraint first configuration or constraints integrated configuration is used in MAGOF.

B.5 Conclusion and Future Work

Specifically for the optimization of the two single objective ship design problems from Chapter 6, the Modular Adaptive Global Optimization Framework (MAGOF) is introduced. MAGOF can solve constraint single objective problems with a mix of computationally expensive and computationally inexpensive constraint and objective functions. MAGOF uses RBF surrogates for expensive functions, the inexpensive functions can directly be used when searching for promising solutions with an acquisition function. Besides this, a strategy is added to MAGOF that enforces the feasibility of the inexpensive constraints before computationally expensive objective and/or computationally expensive constraints are evaluated. MAGOF with the inexpensive constraints used directly when optimizing the acquisition function showed to be the most promising option when optimizing the G-Problem test suite.

In the future, more research is required on how to effectively propose multiple solutions in parallel with other batch acquisition functions described in e.g. [67, 171, 8]. Secondly, more research is required on setting up the parameterization of optimization problems.

Bibliography

[1] Hamid Afshari, Warren Hare, and Solomon Tesfamariam. Constrained multi-objective optimization algorithms: Review and comparison with application in reinforced concrete structures. *Applied Soft Computing*, 83:105631, 2019.

[2] AISHub. Ais data sharing and vessel tracking by aishub, 2024. accessed: 10-01-2023, `https://www.aishub.net/`.

[3] Taimoor Akhtar and Christine A Shoemaker. Efficient multi-objective optimization through population-based parallel surrogate search, 2019.

[4] Richard Allmendinger and Joshua Knowles. Heterogeneous objectives: state-of-the-art and future research. *arXiv preprint arXiv:2103.15546*, 2021.

[5] Juan J Alonso, Patrick LeGresley, and Víctor Pereyra. Aircraft design optimization. *Mathematics and Computers in Simulation*, 79(6):1948–1958, 2009.

[6] Hanan Alsouly, Michael Kirley, and Mario Andrés Muñoz. An instance space analysis of constrained multi-objective optimization problems. *IEEE Transactions on Evolutionary Computation*, 2022.

[7] DJ Andrews. A comprehensive methodology for the design of ships (and other complex systems). *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1968):187–211, 1998.

[8] Javad Azimi, Alan Fern, and Xiaoli Fern. Batch bayesian optimization via simulation matching. *Advances in neural information processing systems*, 23:109–117, 2010.

[9] Thomas Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[10] Thomas Bäck, Anna V Kononova, Bas van Stein, Hao Wang, Kirill Antonov, Roman Kalkreuth, Jacob de Nobel, Diederick Vermetten, **Roy de Winter**, and Furong Ye. Evolutionary algorithms for parameter optimization—thirty years later. *Evolutionary Computation*, 31(2):81–122, 2023.

[11] Samineh Bagheri, Wolfgang Konen, and Thomas Bäck. Online selection of surrogate models for constrained black-box optimization. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

[12] Samineh Bagheri, Wolfgang Konen, and Thomas Bäck. Comparing kriging and radial basis function surrogates. In F. Hoffmann, E. Hüllermeier, and R. Mikut, editors, *Proc. 27. Workshop Computational Intelligence*, pages 243–259. Universitätsverlag Karlsruhe, 2017.

[13] Samineh Bagheri, Wolfgang Konen, Michael Emmerich, and Thomas Bäck. Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. *Applied Soft Computing*, 61:377–393, 2017.

[14] Sunith Bandaru, Amos HC Ng, and Kalyanmoy Deb. Data mining methods for knowledge discovery in multi-objective optimization: Part A-Survey. *Expert Systems with Applications*, 70:139–159, 2017.

[15] Slim Bechikh, Lamjed Ben Said, and Khaled Ghedira. Estimating nadir point in multi-objective optimization using mobile reference points. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–9. IEEE, 2010.

[16] Robert Beck, Arthur Reed, Paul Sclavounos, and Bruce L Hutchison. Modern computational methods for ships in a seaway. *Transactions-Society of Naval Architects and Marine Engineers*, 109:1–51, 2001.

[17] David Berengut. Statistics for experimenters: Design, innovation, and discovery. *The American Statistician*, 60:341–342, 2006.

[18] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[19] Julian Blank and Kalyanmoy Deb. Constrained bi-objective surrogate-assisted optimization of problems with heterogeneous evaluation times: Expensive objectives and inexpensive constraints. In Hisao Ishibuchi, Qingfu Zhang, Ran Cheng, Ke Li, Hui Li, Handing Wang, and Aimin Zhou, editors, *International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, pages 257–269. Springer, 2021.

[20] Julian Blank and Kalyanmoy Deb. pysamoo: Surrogate-assisted multi-objective optimization in python, 2022.

[21] Julian Blank, Kalyanmoy Deb, Yashesh Dhebar, Sunith Bandaru, and Haitham Seada. Generating well-spaced points on a unit simplex for evolutionary many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 25(1):48–60, 2021.

[22] Torsten Bosse, Nicolas R. Gauger, Andreas Griewank, Stefanie Günther, and Volker Schulz. One-shot approaches to design optimzation. In Günter Leugering, Peter Benner, Sebastian Engell, Andreas Griewank, Helmut Harbrecht, Michael Hinze, Rolf Rannacher, and Stefan Ulbrich, editors, *Trends in PDE Constrained Optimization*, pages 43–66, Cham, 2014. Springer International Publishing.

[23] Jakob Bossek, Carola Doerr, and Pascal Kerschke. Initial design strategies and their effects on sequential model-based optimization: an exploratory case study based on BBOB. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 778–786, 2020.

[24] Jakob Bossek, Carola Doerr, Pascal Kerschke, Aneta Neumann, and Frank Neumann. Evolving sampling strategies for one-shot optimization tasks. In Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann, editors, *Parallel Problem Solving from Nature – PPSN XVI*, volume 12269, pages 111–124, Cham, 2020. Springer International Publishing.

[25] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[26] BRL Shipping Consultants. Brl active fleet vessels, 2024. accessed: 20-12-2023, `http://www.brldata.net/Pages/AFV.aspx`.

[27] Philip Bronkhorst, **Roy de Winter**, Thijs Velner, and Austin A Kana. Enhancing offshore service vessel concept design by involving seakeeping-developing a framework to efficiently design high-performance offshore service vessel concepts. In Volker Bertram, editor, *22nd Conference on Computer and IT Applications in the Maritime Industries (COMPIT)*, pages 273–287. Hamburg University of Technology, Schriftenreihe Schiffbau, 2022.

[28] Philip D.H. Bronkhorst. Enhancing offshore service vessel concept design by involving seakeeping. Master's thesis, Delft University of Technology, 2021. `http://resolver.tudelft.nl/uuid:f674236b-c71c-4335-8b37-22f1a53d5a3d`.

[29] Martin D Buhmann. *Radial basis functions: theory and implementations*. Cambridge University Press, 2003.

[30] C-Job Naval Architects. Wind feeder vessel – solution for us offshore wind, 2022. Accessed 20-12-2023, `https://c-job.com/wind-feeder-vessel-solution-for-us-offshore-wind/`.

[31] C-Job Naval Architects. Saronic ferries, 2023. Accessed: 20-12-2023, `https://c-job.com/projects/saronic-ferries/`.

[32] C-Job Naval Architects. Saronic ferries partners with c-job naval architects for the design of the first fully-electric ro-pax ferry in greece, 2023. Accessed 20-12-2023, https://c-job.com/press/saronic-ferries-partners-with-c-job-naval-architects-for-the-design-of-the-first-fully-electric-ro-pax-ferry-in-greece/.

[33] Yu Cai, Dushhyanth Rajaram, and Dimitri N Mavris. Multi-mission multi-objective optimization in commercial aircraft conceptual design. In *AIAA Aviation 2019 Forum*, page 3577, 2019.

[34] Yu Cai, Dushhyanth Rajaram, and Dimitri N Mavris. Simultaneous aircraft sizing and multi-objective optimization considering off-design mission performance during early design. *Aerospace Science and Technology*, 126:107662, 2022.

[35] CN Calvano, O Jons, and RG Keane. Systems engineering in naval ship design. *Naval engineers journal*, 112(4):45–57, 2000.

[36] Tian-You Chai. Challenges of optimal control for plant-wide production processes in terms of control and optimization theories. *Acta automatica sinica*, 35(6):641–649, 2009.

[37] Nikoleta Dimitra Charisi, Hans Hopman, Austin Kana, Nikos Papapanagiotou, and Thijs Muller. Parametric modelling method based on knowledge based engineering: The lng bunkering vessel case. In *Proceedings of the 12th Symposium on High-Performance Marine Vehicles, HIPER'20*. Technische Universität Hamburg-Harburg, 2020.

[38] Tinkle Chugh, Karthik Sindhya, Jussi Hakanen, and Kaisa Miettinen. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing*, 23(9):3137–3166, 2019.

[39] Tinkle Chugh, Karthik Sindhya, Kaisa Miettinen, Jussi Hakanen, and Yaochu Jin. On constraint handling in surrogate-assisted evolutionary many-objective optimization. In *Parallel Problem Solving from Nature–PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings 14*, pages 214–224. Springer, 2016.

[40] Clarksons Research. World fleet register, 2024. accessed: 20-12-2023, `https://www.clarksons.net/wfr/`.

[41] Carlos A Coello Coello, Gary B Lamont, David A van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[42] Carlos A Coello Coello. Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering Systems*, 17(4):319–346, 2000.

[43] Sabine Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196, 1990.

[44] Rituparna Datta and Rommel G Regis. A surrogate-assisted evolution strategy for constrained multi-objective optimization. *Expert Systems with Applications*, 57:270–284, 2016.

[45] GM Watson David. *Practical ship design*. Elsevier Ocean Engineering Series Editor, 1998.

[46] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[47] Kalyanmoy Deb and Samir Agrawal. A niched-penalty approach for constraint handling in genetic algorithms. In Andrej Dobnikar, Nigel C. Steele, David W. Pearson, and Rudolf F. Albrecht, editors, *Artificial neural nets and genetic algorithms*, pages 235–243. Springer Vienna, 1999.

[48] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Trans. Evolutionary Computation*, 18(4):577–601, 2014.

[49] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and Thirunavukarasu Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[50] Kalyanmoy Deb, Amrit Pratap, and T Meyarivan. Constrained test problems for multi-objective evolutionary optimization. In Eckart Zitzler, Lothar Thiele, Kalyanmoy Deb, Carlos Artemio Coello Coello, and David Corne, editors, *International conference on evolutionary multi-criterion optimization (EMO)*, pages 284–298. Springer, 2001.

[51] Audrey Delévacq, Pierre Delisle, Marc Gravel, and Michaël Krajecki. Parallel ant colony optimization on graphics processing units. *Journal of Parallel and Distributed Computing*, 73(1):52–61, 2013.

[52] Det Norske Veritas. *Recommended Practice. Environmental Conditions and Environmental Loads*. DNV RP C205, 2010.

[53] Mike Diessner, Joseph O'Connor, Andrew Wynn, Sylvain Laizet, Yu Guan, Kevin Wilson, and Richard D Whalley. Investigating bayesian optimization for expensive-to-evaluate black box functions: Application in fluid dynamics. *Frontiers in Applied Mathematics and Statistics*, 8:1076296, 2022.

[54] EAE Duchateau. *Interactive evolutionary concept exploration in preliminary ship design*. PhD thesis, Delft University of Technology, 2016.

[55] J. d'Almeida. *Arquitectura Naval – O Dimensionamento do Navio*. Prime Books, 2009.

[56] Agoston E. Eiben and James E. Smith. *Introduction to evolutionary computing*. Natural Computing Series. Springer, Berlin, 2003.

[57] Khairy Elsayed, Dean Vucinic, Roberto Dippolito, and Christian Lacor. Comparison between rbf and kriging surrogates in design optimization of high dimensional problems. In *3rd International Conference on Engineering Optimization*, 2012.

[58] J Harvey Evans. Basic design concepts. *Journal of the American Society for Naval Engineers*, 71(4):671–678, 1959.

[59] Zhun Fan, Yi Fang, Wenji Li, Jiewei Lu, Xinye Cai, and Caimin Wei. A comparative study of constrained multi-objective evolutionary algorithms on constrained multi-objective optimization problems. In *2017 IEEE congress on evolutionary computation (CEC)*, pages 209–216. IEEE, 2017.

[60] Vinicius Sousa Fazio and Mauro Roisenberg. Spatial interpolation: an analytical comparison between kriging and rbf networks. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 2–7, 2013.

[61] Christodoulos A Floudas and Panos M Pardalos. *A collection of test problems for constrained global optimization algorithms*. Springer, 1990.

[62] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[63] Manolis Georgioudakis. PyDE, 2017.

[64] Gerard Petersen. *Powerful Ship Hull Design in Rhino with Rapid Hull Modeling Methodology*. Rhino Centre, 2015. `http://rhinocentre.blogspot.com/2009/11/rhino-rapid-hull-modeling-methodology.html`.

[65] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to parallelize optimization. In *Computational intelligence in expensive optimization problems*, pages 131–162. Springer, 2010.

[66] Wenyin Gong, Zhihua Cai, and Li Zhu. An efficient multiobjective differential evolution algorithm for engineering design. *Structural and Multidisciplinary Optimization*, 38(2):137–157, 2009.

[67] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In *Artificial intelligence and statistics*, pages 648–657. PMLR, 2016.

[68] David Goodfriend and Alan J Brown. Exploration of system vulnerability in naval ship concept design. *Journal of Ship Production and Design*, 34(01):42–58, 2018.

[69] Tim Gourlay, Alexander von Graefe, Vladimir Shigunov, and Evert Lataire. Comparison of aqwa, gl rankine, moses, octopus, pdstrip and wamit with model test results for cargo ship wave-induced motions in shallow water. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 56598, page V011T12A006. American Society of Mechanical Engineers, 2015.

[70] Steven Gustafson and Edmund K Burke. The speciating island model: An alternative parallel evolutionary algorithm. *Journal of Parallel and Distributed Computing*, 66(8):1025–1036, 2006.

[71] M Gutsch, S Steen, and F Sprenger. Operability robustness index as seakeeping performance criterion for offshore vessels. *Ocean Engineering*, 217:107931, 2020.

[72] Raphael T. Haftka, Diane Villanueva, and Anirban Chaudhuri. Parallel surrogate-assisted global optimization with expensive functions - a survey. *Structural and Multidisciplinary Optimization*, 54(1):3–13, 2016.

[73] John H Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2(1):84–90, 1960.

[74] Zhonghua Han, Fei Liu, Chenzhou Xu, Keshi Zhang, and Qingfu Zhang. Efficient multi-objective evolutionary algorithm for constrained global optimization of expensive functions. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2026–2033. IEEE, 2019.

[75] Gideon Hanse, **Roy de Winter**, Bas van Stein, and Thomas Bäck. Optimally weighted ensembles for efficient multi-objective optimization. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 144–156. Springer, 2021.

[76] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

[77] DP Hardin and EB Saff. Minimal riesz energy point configurations for rectifiable d-dimensional manifolds. *Advances in Mathematics*, 193(1):174–204, 2005.

[78] Julian Heinrich and Daniel Weiskopf. State of the art of parallel coordinates. In *Eurographics (STARs)*, pages 95–116, 2013.

[79] J Holtrop, GGJ Mennen, et al. An approximate power prediction method. *International Shipbuilding Progress*, 29(335):166–170, 1982.

[80] Qi Huang, **Roy de Winter**, Bas van Stein, Thomas Bäck, and Anna V Kononova. Multi-surrogate assisted efficient global optimization for discrete problems. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1650–1658. IEEE, 2022.

[81] IMO. Chapter II-1 - Construction - Structure, subdivision and stability, machinery and electrical installations, Part B - Subdivision and stability, 2020.

[82] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified distance calculation in generational distance and inverted generational distance. In *Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29–April 1, 2015. Proceedings, Part II 8*, pages 110–125. Springer, 2015.

[83] Himanshu Jain and Kalyanmoy 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*, 18(4):602–622, 2014.

[84] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[85] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[86] Shahroz Khan, Kosa Goucher-Lambert, Konstantinos Kostas, and Panagiotis Kaklis. Shiphullgan: A generic parametric modeller for ship hull design using deep convolutional generative model. *Computer Methods in Applied Mechanics and Engineering*, 411:116051, 2023.

[87] Bhuvan Khoshoo, Julian Blank, Thang Q. Pham, Kalyanmoy Deb, and Shanelle N. Foster. Optimal design of electric machine with efficient handling of constraints and surrogate assistance. *Engineering Optimization*, pages 1–19, 2023.

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

[89] Alexander Kossiakoff, William N Sweet, et al. *Systems engineering: Principles and practices*. Wiley Online Library, 2003.

[90] Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in bayesian optimization with applications in aerospace engineering. In *2018 AIAA Non-Deterministic Approaches Conference*, page 1656, 2018.

[91] Sangmin Lee and Seoung Bum Kim. Parallel simulated annealing with a greedy algorithm for bayesian network structure learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1157–1166, 2019.

[92] Jian-Yu Li, Zhi-Hui Zhan, and Jun Zhang. Evolutionary computation for expensive optimization: A survey. *International Journal of Automation and Computing*, 18:1–21, October 2021.

[93] Jing J Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, Ponnuthurai Nagaratnam Suganthan, CA Coello Coello, and Kalyanmoy Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics*, 41(8):8–31, 2006.

[94] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[95] Nasrulloh Loka, Ivo Couckuyt, Federico Garbuglia, Domenico Spina, Inneke van Nieuwenhuyse, and Tom Dhaene. Bi-objective bayesian optimization of engineering problems with cheap and expensive cost functions. *Engineering with Computers*, 39(3):1923–1933, 2023.

[96] Manuel López-Ibáñez, Luís Paquete, and Thomas Stützle. Exploratory analysis of stochastic local search algorithms in biobjective optimization. In homas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, editors, *Experimental methods for the analysis of optimization algorithms*, pages 209–222. Springer, 2010.

[97] Zhongwei Ma and Yong Wang. Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons. *IEEE Transactions on Evolutionary Computation*, 23(6):972–986, 2019.

[98] Marine Traffic. Vessels, 2024. accessed: 20-12-2023, `https://www.marinetraffic.com/en/data/?asset\_type=vessels`.

[99] R Timothy Marler and Jasbir S Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization*, 41:853–862, 2010.

[100] J Marzi, A Papanikolaou, P Corrignan, G Zaraphonitis, and S Harries. Holistic ship design for future waterborne transport. *Proceedings of the 7th Transport Research Arena TRA*, 2018.

[101] Charles A Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive approximation*, 2(1):11–22, 1986.

[102] Bas Milatz. Multi-level optimisation and global sensitivity analysis of the probabilistic damage stability method for single hold ships. Master's thesis, Delft University of Technology, 2022. `http://resolver.tudelft.nl/uuid:24b94160-8804-4d9a-887f-24e3c68ae89c`.

[103] Bas Milatz, **Roy de Winter**, Jelle DJ van de Ridder, Martijn van Engeland, Francesco Mauro, and Austin A Kana. Parameter space exploration for the probabilistic damage stability method for dry cargo ships. *International Journal of Naval Architecture and Ocean Engineering*, page 100549, 2023.

[104] Jeremy Miles. R squared, adjusted r squared. *Wiley StatsRef: Statistics Reference Online*, 2014.

[105] Seyedali Mirjalili, Pradeep Jangir, and Shahrzad Saremi. Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. *Applied Intelligence*, 46(1):79–95, 2017.

[106] Farrokh Mistree, WF Smith, B Bras, JK Allen, D Muster, et al. Decision-based design: a contemporary paradigm for ship design. *Transactions, Society of Naval Architects and Marine Engineers*, 98(1990):565–597, 1990.

[107] Jonas Močkus. On Bayesian methods for seeking the extremum. In Josef Stoer, editor, *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

[108] Anthony F Molland. *The maritime engineering reference book: a guide to ship design, construction and operation*. Elsevier, 2011.

[109] Yew Soon Ong, PB Nair, AJ Keane, and KW Wong. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In Yoachu Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, pages 307–331. Springer, 2005.

[110] IMO International Maritime Organization. International convention for the safety of life at sea (solas), 1974, 2023.

[111] Emre Özkaya and Nicolas R Gauger. Single-step one-shot aerodynamic shape optimization. In *Optimal control of coupled systems of partial differential equations*, pages 191–204. Springer, 2009.

[112] A Papanikolaou, S Harries, M Wilken, and G Zaraphonitis. Integrated design and multiobjective optimization approach to ship design. In *Proceedings of the 15th International Conference on Computer Applications in Shipbuilding, ICCASAt: Trieste*, 2011.

[113] Apostolos Papanikolaou. Holistic ship design optimization. *Computer-Aided Design*, 42(11):1028–1044, 2010.

[114] Apostolos Papanikolaou. *A Holistic Approach to Ship Design*. Springer, 2019.

[115] Apostolos Papanikolaou, George Zaraphonitis, Markus Jokinen, Adrien Aubert, Stephan Harries, Jochen Marzi, George Mermiris, and Rachmat Gunawan. Holistic ship design for green shipping. In *SNAME Maritime Convention*. OnePetro, 2022.

[116] Sung-Wook Park, Jae-Hoon Choi, and Byung-Chai Lee. Multi-objective optimization of an automotive body component with fiber-reinforced composites. *Structural and Multidisciplinary Optimization*, 58:2203–2217, 2018.

[117] Michael G Parsons and Randall L Scott. Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. *Journal of Ship Research*, 48(1):61–76, 2004.

[118] Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection. In Günter Rudolph, Thomas Jansen, Nicola Beume, Simon Lucas, and Carlo Poloni, editors, *International Conference on Parallel Problem Solving from Nature (PPSN)*, pages 784–794. Springer, 2008.

[119] Wolfgang Ponweiser, Tobias Wagner, and Markus Vincze. Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models. In *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)*, pages 3515–3522. IEEE, IEEE, 2008.

[120] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In Susana Gomez and Jean-Pierre Hennart, editors, *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer Netherlands, 1994.

[121] Alexandros Priftis, Apostolos Papanikolaou, and Timoleon Plessas. Parametric design and multiobjective optimization of containerships. *Journal of Ship Production and Design*, 32(3):1–14, 2016.

[122] Hoyte Christiaan Raven. *A solution method for the nonlinear ship wave resistance problem*. Phd thesis, Delft University of Technology, 1998.

[123] Rommel G Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243, 2014.

[124] Rommel G Regis. A survey of surrogate approaches for expensive constrained black-box optimization. In Hoai An Le Thi, Hoai Minh Le, and Tao Pham Dinh, editors, *World Congress on Global Optimization*, pages 37–47. Springer, 2019.

[125] Rommel G Regis and Christine A Shoemaker. A quasi-multistart framework for global optimization of expensive functions using response surface models. *Journal of Global Optimization*, 56(4):1719–1753, 2013.

[126] Frederik Rehbach, Martin Zaefferer, Boris Naujoks, and Thomas Bartz-Beielstein. Expected improvement versus predicted value in surrogate-based optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, GECCO '20, page 868–876, New York, NY, USA, 2020. Association for Computing Machinery.

[127] Carl Fredrik Rehn. *Ship design under uncertainty*. PhD thesis, Norwegian University of Science and Technology, 2018.

[128] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *2015 Latin American computing conference (CLEI)*, pages 1–11. IEEE, 2015.

[129] Herbert Schneekluth and Volker Bertram. *Ship design for efficiency and economy*, volume 218. Butterworth-Heinemann Oxford, 1998.

[130] Thomas P Scholcz, Tomasz Gornicz, and Christian Veldhuis. Multi-objective hull-form optimization using kriging on noisy computer experiments. In *MARINE VI: proceedings of the VI International Conference on Computational Methods in Marine Engineering*, pages 1064–1077. CIMNE, CIMNE, 2015.

[131] Lei Shi, RJ Yang, and Ping Zhu. A method for selecting surrogate models in crashworthiness optimization. *Structural and Multidisciplinary Optimization*, 46(2):159–170, 2012.

[132] Daniel Sieger, Stefan Menzel, Mario Botsch, et al. A comprehensive comparison of shape deformation methods in evolutionary design optimization. In *Proceedings of the International Conference on Engineering Optimization*, pages 1–5. Citeseer, 2012.

[133] Prashant Singh, Ivo Couckuyt, Francesco Ferranti, and Tom Dhaene. A constrained multi-objective surrogate-based optimization algorithm. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014.

[134] I.M Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[135] Siemens Digital Industries Software. Simcenter STAR-CCM+ User Guide, version 2021.1. In *Adaptive Mesh Refinement for Overset Meshes*, pages 3067–3070. Siemens, 2021.

[136] S&P Market Intelligence. Sea-web ships, 2024. accessed: 20-12-2023, `https://www.spglobal.com/marketintelligence/en/mi/products/sea-web-vessel-search.html`.

[137] Bas van Stein, Hao Wang, and Thomas Bäck. Automatic configuration of deep neural networks with parallel efficient global optimization. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2019.

[138] Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[139] Robert Taggart. *Ship design and construction*. Society of Naval Architects & Marine Engineers, 1980.

[140] Yusuke Tahara, F Stern, and Y Himeno. Computational fluid dynamics–based optimization of a surface combatant. *Journal of ship Research*, 48(04):273–287, 2004.

[141] Ryoji Tanabe and Akira Oyama. A note on constrained multi-objective optimization benchmark problems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1127–1134. IEEE, 2017.

[142] **Roy de Winter**. Designing ships using constrained multi-objective efficient global optimization, 2018. Available at `https://theses.liacs.nl/pdf/2017-2018-WinterRoyde.pdf`.

[143] **Roy de Winter**. RoydeZomer/SAMO-COBRA: Release with new experiments, November 2020. `https://doi.org/10.5281/zenodo.5105636`.

[144] **Roy de Winter**. Roy de winter/multi-point-samo-cobra: Release1, February 2022. https://doi.org/10.5281/zenodo.6461614.

[145] **Roy de Winter**. IOC-SAMO-COBRA: Release 1.1.1 for Swarm and Evolutionary Computation, July 2023. https://doi.org/10.5281/zenodo.8112883.

[146] **Roy de Winter**. Parallel constrained multi-objective optimization for ship design damage stability problem with (in)expensive function evaluations. In *Marine 2023*, page 1. Marine 2023, Scipedia, 2023.

[147] **Roy de Winter**, Thomas Bäck, and Niki van Stein. Modular optimization framework for mixed expensive and inexpensive real-world problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2024.

[148] **Roy de Winter**, Philip Bronkhorst, Bas van Stein, and Thomas Bäck. Constrained multi-objective optimization with a limited budget of function evaluations. *Memetic Computing*, 14:151–164, 2022.

[149] **Roy de Winter**, Jan Furustam, Thomas Bäck, and Thijs Muller. Optimizing ships using the holistic accelerated concept design methodology. In Tetsuo Okada, Katsuyuki Suzuki, and Yasumi Kawamura, editors, *Practical Design of Ships and Other Floating Structures (PRADS)*, pages 38–50, Singapore, 2021. Springer.

[150] **Roy de Winter**, Fu Xing Long, Andre Thomaser, Niki van Stein, de, Thomas Bäck, and Anna V Kononova. Landscape analysis based vs. domain-specific optimization algorithm selection for engineering design applications: A clear case. In *2024 IEEE Conference on Artificial Intelligence (CAI)*. IEEE, 2023.

[151] **Roy de Winter**, Bas Milatz, Julian Blank, Niki van Stein, Thomas Bäck, and Kalyanmoy Deb. Hot off the press: Parallel multi-objective optimization for expensive and inexpensive objectives and constraints. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2024.

[152] **Roy de Winter**, Bas Milatz, Julian Blank, Niki van Stein, Thomas Bäck, and Kalyanmoy Deb. Parallel multi-objective optimization for expensive and inexpensive objectives and constraints. *Swarm and Evolutionary Computation*, 86:101508, 2024.

[153] **Roy de Winter**, Bas van Stein, and Thomas Bäck. SAMO-COBRA: A fast surrogate assisted constrained multi-objective optimization algorithm. In Hisao Ishibuchi, Qingfu Zhang, Ran Cheng, Ke Li, Hui Li, Handing Wang, and Aimin Zhou, editors, *International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, pages 270–282. Springer, 2021.

[154] **Roy de Winter**, Bas van Stein, Matthys Dijkman, and Thomas Bäck. Designing ships using constrained multi-objective efficient global optimization. In

Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, Renato Umeton, and Vincenzo Sciacca, editors, *International Conference on Machine Learning, Optimization, and Data Science*, pages 191–203. Springer, 2018.

[155] **Roy de Winter**, Bas van Stein, and Thomas Bäck. Ship design performance and cost optimization with machine learning. In *20st Conference on Computer and IT Applications in the Maritime Industries (COMPIT)*, pages 185–196. Hamburg University of Technology, 2020.

[156] **Roy de Winter**, Bas van Stein, and Thomas Bäck. Multi-point acquisition function for constraint parallel efficient multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 511–519, 2022.

[157] André Thomaser, Anna V Kononova, Marc-Eric Vogt, and Thomas Bäck. One-shot optimization for vehicle dynamics control systems: towards benchmarking and exploratory landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 2036–2045, 2022.

[158] Ye Tian, Ran Cheng, Xingyi Zhang, Miqing Li, and Yaochu Jin. Diversity assessment of multi-objective evolutionary algorithms: Performance metric and benchmark problems. *IEEE Computational Intelligence Magazine*, 14(3):61–74, 2019.

[159] Magnus Urquhart, Emil Ljungskog, and Simone Sebben. Surrogate-based optimisation using adaptively scaled radial basis functions. *Applied Soft Computing*, 88:106050, 2020.

[160] Magnus Urquhart, Emil Ljungskog, and Simone Sebben. Surrogate-based optimisation using adaptively scaled radial basis functions. *Applied Soft Computing*, 88:106050, 2020.

[161] Koen van der Blom, Timo M Deist, Vanessa Volz, Mariapia Marchi, Yusuke Nojima, Boris Naujoks, Akira Oyama, and Tea Tušar. Identifying properties of real-world optimisation problems through a questionnaire. In *Many-Criteria Optimization and Decision Analysis: State-of-the-Art, Present Challenges, and Future Perspectives*, pages 59–80. Springer, 2023.

[162] Lucas Van Rooij, **Roy de Winter**, Anna V Kononova, and Bas van Stein. Explainable AI for ship design analysis with AIS and static ship data. In *15th International Symposium on Practical Design of Ships and Other Floating Structures (PRADS)*, pages 1521–1535, 2022.

[163] Niki van Stein, de **Roy de Winter**, Thomas Bäck, and Anna V Kononova. AI for Expensive Optimization Problems in Industry. In *2023 IEEE Conference on Artificial Intelligence (CAI)*, pages 251–254. IEEE, 2023.

[164] Takashi Wada and Hideitsu Hino. Bayesian optimization for multi-objective optimization and multi-point search, 2019.

[165] Hao Wang, Bas van Stein, Michael Emmerich, and Thomas Bäck. A new acquisition function for bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 507–512. IEEE, IEEE, 2017.

[166] Hao Wang, Diederick Vermetten, Furong Ye, Carola Doerr, and Thomas Bäck. IOHanalyzer: Detailed performance analyses for iterative optimization heuristics. *ACM Transactions on Evolutionary Learning and Optimization*, 2(1), apr 2022.

[167] Jolan Wauters, Andy Keane, and Joris Degroote. Development of an adaptive infill criterion for constrained multi-objective asynchronous surrogate-based optimization. *Journal of Global Optimization*, 78(1):137–160, 2020.

[168] Henry G Weller, Gavin Tabor, Hrvoje Jasak, and Christer Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, 12(6):620–631, 1998.

[169] Thomas Wortmann, Christoph Waibel, Giacomo Nannicini, Ralph Evins, Thomas Schroepfer, and Jan Carmeliet. Are genetic algorithms really the best choice for building energy optimization? In Michela Turrin, Brady Peters, William O'Brien, Rudi Stouffs, and Timur Dogan, editors, *Proceedings of the Symposium on Simulation for Architecture and Urban Design*, pages 51–58, 2017.

[170] Haofeng Wu, Jinliang Ding, and Qingda Chen. Gaussian process-assisted evolutionary algorithm for constrained expensive multi-objective optimization. In *Asian Control Conference (ASCC)*, pages 1027–1032. IEEE, 2022.

[171] Jian Wu and Peter Frazier. The parallel knowledge gradient method for batch bayesian optimization. *Advances in neural information processing systems*, 29:3126–3134, 2016.

[172] Yongkuan Yang, Jianchang Liu, and Shubin Tan. A multi-objective evolutionary algorithm for steady-state constrained multi-objective optimization problems. *Applied Soft Computing*, 101(107042), 2021.

[173] Yiming Yao and Xudong Yang. Efficient global multi-objective aerodynamic optimization using combined multi-point infilling strategy and surrogate models. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1537–1542. IEEE, 2021.

[174] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[175] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *International conference on parallel problem solving from nature (PPSN)*, pages 292–301. Springer, 1998.

Summary

Constraint multi-objective optimization with a restriction to the number of allowed function evaluations is a challenging topic. This thesis proposes a solution for constraint multi-objective optimization problems, with a special emphasis on its application in the ship design industry. It distinguishes itself by abandoning traditional design methodologies by introducing a more holistic, computational framework that uses efficient global optimization for complex design challenges. These algorithms are intended to help naval architects deal with these design challenges, where competing objectives such as cost, efficiency, environmental impact, and safety must be balanced while the constraints imposed by physics and regulations should be satisfied. This shift is facilitated by advancements in computational power and simulation technologies, enabling designers to explore more of the design space efficiently.

Central to this research is the development and application of efficient constraint multi-objective optimization algorithms that are capable of identifying the feasible Pareto frontier of computationally expensive problems. The proposed algorithms do so by exploiting surrogates for the computationally expensive functions, while the computationally inexpensive functions are used directly when searching for promising solutions that jointly have a large hypervolume contribution. Because of the use of the introduced multi-point acquisition function, the expensive evaluation of the solutions can be done in parallel. This research provides a comprehensive examination of these algorithms on a diverse set of benchmark problems and compares them with other state-of-the-art algorithms. After empirically proving the success, the algorithms are deployed and used to solve ship design optimization problems.

This thesis also highlights the importance of the early design phase, arguing that decisions made during this stage have a significant impact on the ship's lifecycle costs and performance. By integrating multi-objective optimization techniques early in the design process, the designers can make more informed decisions that lead to more

cost-effective, efficient, and environmentally friendly vessels. This approach not only enhances the sustainability of ship designs but also aligns with the broader industry trend towards greener and more sustainable maritime operations.

The practical applications of these optimization techniques are demonstrated with a series of case studies. These case studies not only validate the effectiveness of the algorithms in real-world scenarios but also provide valuable insights into the challenges and opportunities associated with their implementation. Through these practical examples, the research bridges the gap between theory and practice, offering a compelling argument for the adoption of optimization techniques in ship design.

Looking toward the future, several areas for further research are identified, including the need for more efficient algorithms that can handle mixed integer optimization problems. It also calls for a broader application of these techniques beyond ship design, the use of the proposed algorithms can be interesting to other engineering disciplines faced with similar multi-objective optimization problems.

In conclusion, the research presented in this dissertation represents a significant contribution to the field of multi-objective optimization and naval architecture. By advancing the state of the art in multi-objective optimization and demonstrating its practical applications in ship design, the research paves the way for more innovative, sustainable, and cost-effective design solutions. This work not only enhances our understanding of the complexities involved in ship design but also offers a blueprint for the future of maritime engineering, where computational optimization techniques play a central role in addressing the industry's most pressing challenges.

Samenvatting

Meerdere-doelstellingen optimalisatie met functionele restricties en met een limiet op het aantal toegestane functie-evaluaties is een uitdagend onderzoeksonderwerp. Deze thesis geeft een oplossing voor constraint multi-objective optimalisatieproblemen, met speciale nadruk op de toepassing in scheepsontwerp. Het onderscheidt zich door traditionele ontwerpmethodologieën met een meer holistisch, computationeel intensievere methodologie die efficiënte globale optimalisatie gebruikt voor complexe ontwerpuitdagingen. Deze algoritmes die voorgesteld worden zijn bedoeld om maritiem ingenieurs te helpen bij het optimaliseren van scheepsontwerpen, waarbij concurrerende doelstellingen zoals kosten, efficiëntie, milieueffect en veiligheid in evenwicht moeten worden gebracht binnen de door fysica en regelgeving opgelegde beperkingen. Deze verschuiving wordt vergemakkelijkt door vooruitgang in computationele kracht en simulatietechnologieën, waardoor ontwerpers efficiënter meer van de ontwerpruimte kunnen verkennen.

Centraal in dit onderzoek staat de ontwikkeling en toepassing van efficiënte constraint multi-objective optimalisatie algoritmes die in staat zijn de Pareto-front van computationeel intensieve problemen te identificeren. De voorgestelde algoritmen werken door gebruik te maken van surrogaten voor de computationeel intensieve functies terwijl de goedkopere functies direct gebruikt worden tijdens het zoeken naar veelbelovende oplossingen die gezamelijk een groot aandeel hebben in toegevoegd hypervolume. Door het gebruik maken van de multi-point acquisitie functie kunnen meerdere voorgestelde oplossingen tegelijkertijd geevalueerd worden. Dit onderzoek biedt een uitgebreide evaluatie van de deze algoritmes door ze te testen op diverse benchmark problemen en ze te vergelijken met andere algorithmes. Na het empirisch bewijzen dat ze goed werken zijn de algoritmen ingezet in hedendaags scheeps optimalisatie problemen op te lossen.

Deze thesis benadrukt ook het belang van de vroege ontwerpfase, door te stellen

dat beslissingen die in deze fase worden genomen een diepgaande invloed hebben op de levenscycluskosten en prestaties van het schip. Door multi-objective optimalisatietechnieken vroeg in het ontwerpproces te integreren, kunnen ontwerpers meer geïnformeerde beslissingen nemen die leiden tot kosteneffectievere, efficiëntere en milieuvriendelijkere schepen. Deze aanpak verbetert niet alleen de duurzaamheid van scheepsontwerpen, maar sluit ook aan bij de bredere industrietrend naar groenere en meer duurzame maritieme operaties.

De praktische toepassingen van deze optimalisatietechnieken worden gedemonstreerd door een reeks casestudy's. Deze casestudy's valideren niet alleen de effectiviteit van de algoritmes in real-world scenario's, maar bieden ook waardevolle inzichten in de uitdagingen en kansen die gepaard gaan met hun implementatie. Via deze praktische voorbeelden overbrugt het onderzoek de kloof tussen theorie en praktijk, en biedt een overtuigend argument voor de adoptie van optimalisatietechnieken in scheepsontwerp.

Kijkend naar de toekomst, worden verschillende gebieden voor verder onderzoek geïdentificeerd, inclusief de behoefte aan efficiëntere algoritmes die mixed-integer optimalisatieproblemen kunnen aanpakken. Er wordt ook opgeroepen tot een bredere toepassing van deze technieken buiten scheepsontwerp, omdat ze potentieel nuttig kunnen zijn in andere technische disciplines die geconfronteerd worden met vergelijkbare multi-objective optimalisatieproblemen.

Samenvattend vertegenwoordigt het onderzoek gepresenteerd in deze dissertatie een significante bijdrage aan het veld van multi-objective optimalisatie en scheepsarchitectuur. Door de stand van zaken op het gebied van multi-objective optimalisatie vooruit te helpen en de praktische toepassingen ervan in scheepsontwerp te demonstreren, maakt het onderzoek de weg vrij voor innovatievere, duurzamere en kosteneffectievere ontwerpoplossingen. Dit werk verbetert niet alleen ons begrip van de complexiteit betrokken bij scheepsontwerp, maar biedt ook een blauwdruk voor de toekomst van maritieme engineering, waar computationele optimalisatietechnieken een centrale rol spelen bij het aanpakken van de meest urgente uitdagingen van de industrie.

Acknowledgements

Many wise people I know say that the time they were students was one of the best times of their life. I agree with them, saying that my time as a PhD student was a lot of fun, very rewarding, and above all, a great time. This time was so nice because of the never-ending support, guidance, and freedom I received from my supervisors and managers: Prof. Dr. Thomas Bäck, Dr. Niki van Stein, Thijs Muller, Tim Vlaar, Job Volwater, and Basjan Faber. They all gave me the feeling that they believed in me and allowed me to pursue research into the topics I was most interested in. They did so by challenging me, allocating the right resources, sending me to conferences, making me feel like a valuable team member of both C-Job Naval Architects and the Natural Computing group, and arranging an exchange with the COIN lab of Prof. Dr. Kalyanmoy Deb (a big thank you for inviting me).

Secondly, I would like to thank all my collaborators who inspired me with their ideas, helped me develop academic skills, and assisted me in writing impactful papers. More specifically, I would like to thank Philip Bronkhorst, Matthys Dijkman, Jan Furustam, Jan Willem Krijger, Bas Milatz, Thijs Muller, Jelle van de Ridder, and Bob van Veen, who all contributed to the practical applications of the algorithms presented in this work. In addition to my work, I would like to thank Philip Bronkhorst, Prof. Dr. Thomas Bäck, Dr. Kalyanmoy Deb, Gideon Hanse, Qi Huang, Dr. Austin Kana, Dr. Anna Kononova, Fu Xing Long, Bas Milatz, Jacob de Nobel, Lucas van Rooij, Dr. Niki van Stein, Andre Thomaser, Diederick Vermetten, Dr. Furong Ye for involving me in their work.

Thirdly, I would like to thank my colleagues with whom I have worked. I really enjoyed the lectures from Michael Emmerich, who introduced me to multi-objective optimization and surrogate-assisted optimization. I felt very fortunate that I could always turn to Andre Deutz and Hao Wang to discuss how to find optimal solutions for various optimization problems and to work out the mathematics for them. Gyula

Sarsanszky, I will not forget our daily commutes to and from the office in the beginning of my research. For everything I needed to know about sustainable fuels or anything related to naval architecture, I could knock on Niels de Vries's door (thank you so much). During paper writing sessions, I had great assistance for the smallest practical things from Diederick Vermetten. Santosh Kumar and Ritam Guha showed me that working internationally can also be a lot of fun. One day, I hope to visit the COIN lab again. In short, I hope I didn't forget anybody in particular, but I have been working among great colleagues who made me feel welcome, allowed me to be my true self, and made work, lunches, dinners, coffee breaks, and social outings a lot of fun.

My last thank you goes out to my girlfriend, parents, family, friends, kite buddies, ski friends, my bouldering mates, and my long-time friends from high school. They kept me grounded, shared in my happiness, and listened to my troubles when needed. My dear friends and family, you make my life joyful, and you make sure that I make time for my hobbies, friends, or a well-deserved holiday.

This PhD journey has been one of the best periods of my life thus far. To all of the people above, thank you for being part of it.

About the Author

Roy de Winter was born on the 2nd of June 1994 in Nisse. After finishing his Bachelor's degree in computer science and economics at Leiden University in 2016, Roy continued his studies and obtained a master's degree in computer science with a data science specialization in 2018 also at Leiden University. After gaining some working experience at C-Job Naval Architects as a research and development engineer, in January 2020 Roy arranged that besides his commercial work at C-Job, he could continue his research in efficient optimization algorithms as a guest PhD candidate at the Leiden Institute of Advanced Computer Science under the supervision of Prof. dr. Thomas Bäck and Dr. Niki van Stein.