Efficient constraint multi-objective optimization with applications in ship design

Winter, R. de

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.
    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.

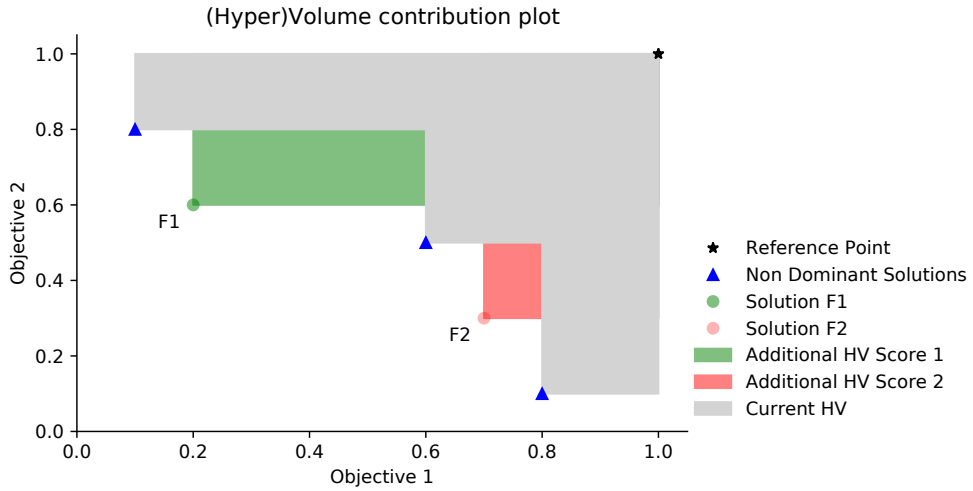


Figure 5.1: Visual representation of hypervolume contribution of two solutions. The hypervolume contribution of solution F1 is equal to $0.2 \cdot 0.4 = 0.08$, the hypervolume contribution of solution F2 is equal to $0.1 \cdot 0.1 = 0.01$. This makes solution F1 more desirable compared to solution F2.

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.

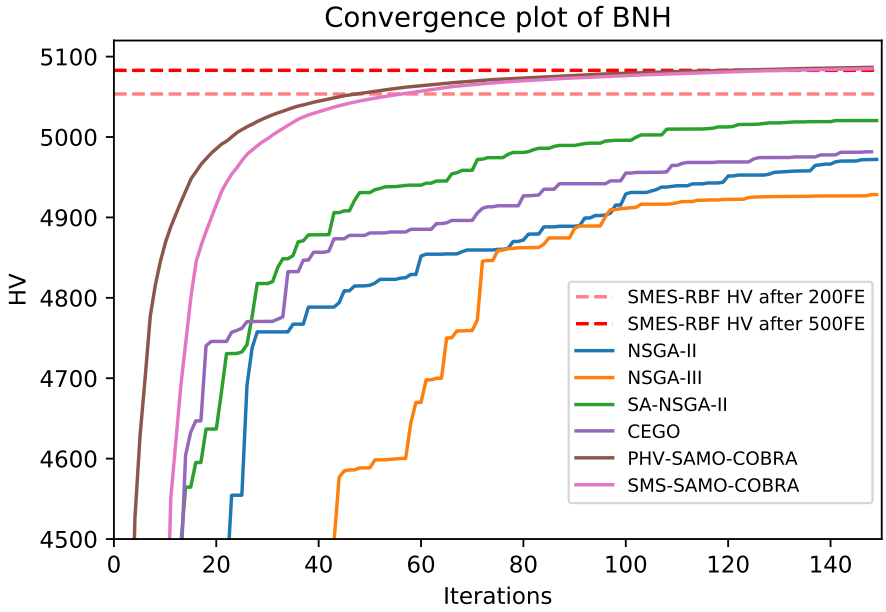


Figure 5.2: Convergence plot on BNH test problem for the NSGA-II, NSGA-III, SA-NSGA-II, CEGO, $\mathcal{P}_{hv}$-SAMO-COBRA, $\mathcal{S}$-metric-SAMO-COBRA algorithms. . The dashed lines represents the final obtained Hypervolume of SMES-RBF after 200 and 500 function evaluations.

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

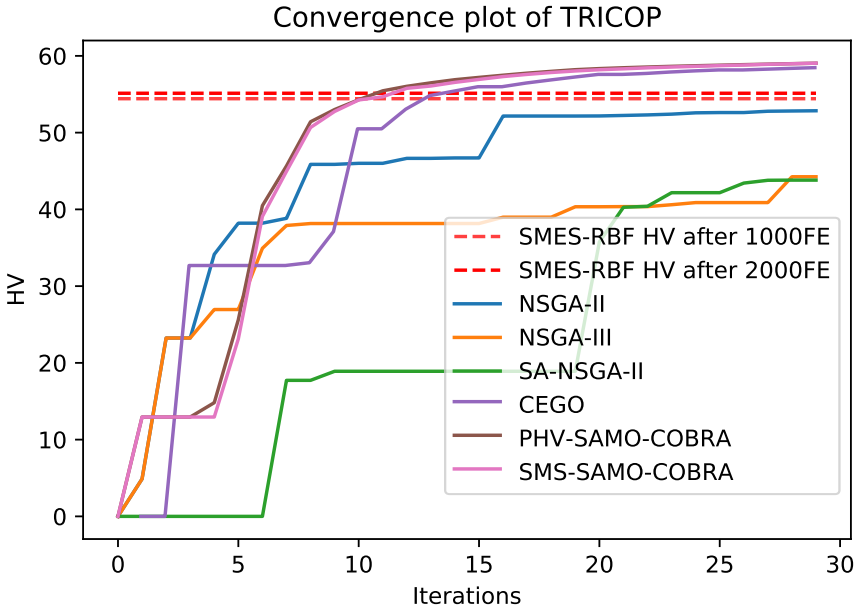


Figure 5.3: Convergence plot on TRICOP test problem for the NSGA-II, NSGA-III, SA-NSGA-II, CEGO, $\mathcal{P}_{hv}$-SAMO-COBRA, $\mathcal{S}$-metric-SAMO-COBRA algorithms. The dashed lines represent the final obtained Hypervolume of SMES-RBF after 1000 and 2000 function evaluations.

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

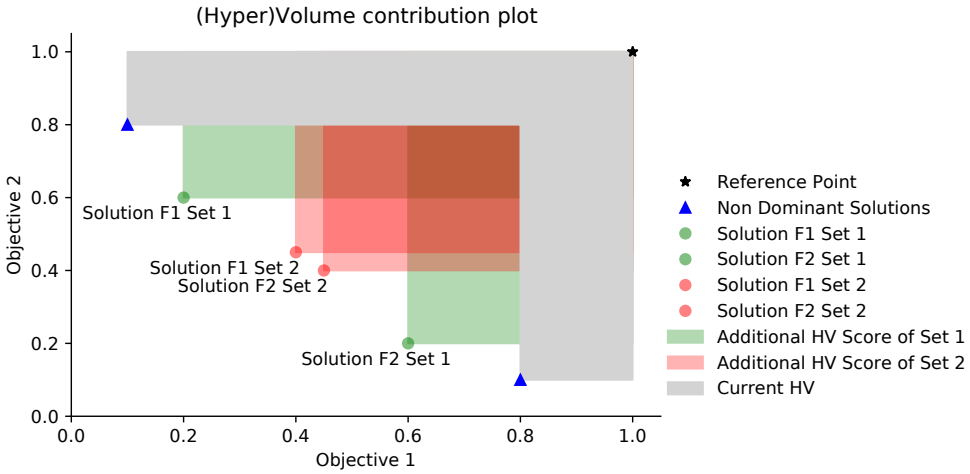


Figure 5.4: Visual representation of hypervolume contributions of two sets containing two solutions each. The hypervolume contribution of set 1 is equal to $2(0.6 \cdot 0.2) - (0.2 \cdot 0.2) = 0.2$. The hypervolume contribution of set 2 is equal to $2(0.35 \cdot 0.4) - (0.35 \cdot 0.35) = 0.1575$. So 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.

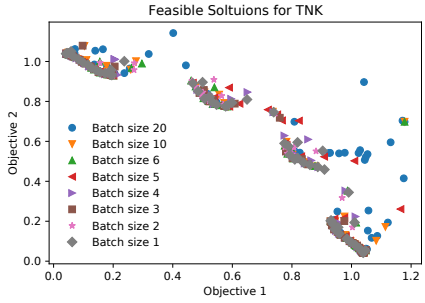
Table 5.5: Mean and standard deviation of hypervolume (hv) between Pareto frontier and Nadir point on set of test functions after $40 \cdot d$ function evaluations with different batch sizes (1,2,3,4,5,6,10,20) for the $\mathcal{MP}_{hv}$ infill criteria given 11 independent runs. HV Scores in **bold** 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	$3.677 \cdot 10^{18}$	$5 \cdot 10^{15}$	$3.662 \cdot 10^{18}$	$3 \cdot 10^{15}$	$3.653 \cdot 10^{18}$	$9 \cdot 10^{15}$	$3.631 \cdot 10^{18}$	$1.4 \cdot 10^{16}$

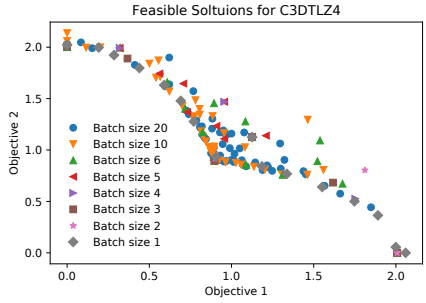
Function	Batch size 5		Batch size 6		Batch size 10		Batch size 20	
	hv	std	hv	std	hv	std	hv	std
BNH	4967.9	0.7	4967.6	1.2	4960.3	2.3	4949.8	5.7
CEXP	3.7964	0.0004	**3.7981***	0.0004	3.7925	0.0005	3.7794	0.0030
SRN	24729	53	24723	39	24583	97	24516	103
TNK	0.2953	0.0012	0.2985	0.0018	0.2957	0.0024	0.2676	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.

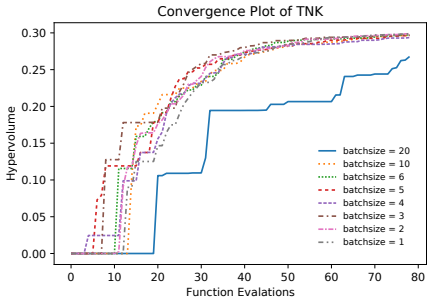
Table 5.6: Rounded mean number evaluations (Eval), mean number of algorithm iterations (Itr) given different batch sizes (1, 2, 3, 4, 5, 6, 10, 20) to achieve the hypervolume threshold. The threshold is 90% of the dominated area between the Nadir point and the Pareto frontier of all runs combined. A 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

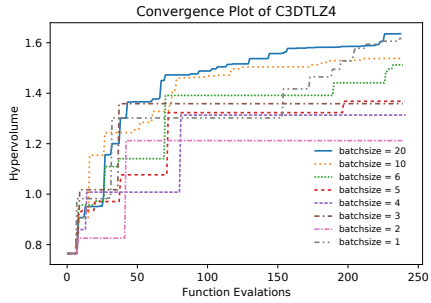
Function	Threshold	Batch size 5		Batch size 6		Batch size 10		Batch size 20	
		Eval	Itr	Eval	Itr	Eval	Itr	Eval	Itr
BNH	4496.2	8	2	9	2	13	2	21	2
CEXP	3.4380	13	3	11	2	17	2	31	2
SRN	22810	18	4	20	4	30	4	37	2
TNK	0.2775	46	10	44	8	49	5	68↓	4↓
CTP1	0.2717	19	4	16	3	19	2	34	2
C3DTLZ4	1.5788	-	-	-	232↓	24↓	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.

Continuation of Table 5.9.

Function	p	SA-NSGA-II	IC-SA-NSGA-II	SAMO-COBRA	IOC-SAMO-COBRA
WB	1	$2.46 \cdot 10^{-1} \pm 5.47 \cdot 10^{-2}$ (−)	**$4.19 \cdot 10^{-1} \pm 2.11 \cdot 10^{-2}$** (+)	$3.77 \cdot 10^{-1} \pm 1.01 \cdot 10^{-2}$ (−)	$4.15 \cdot 10^{-1} \pm 1.43 \cdot 10^{-3}$
	2	$3.46 \cdot 10^{-1} \pm 4.48 \cdot 10^{-2}$ (−)	**$4.20 \cdot 10^{-1} \pm 3.02 \cdot 10^{-3}$** (≈)	$3.87 \cdot 10^{-1} \pm 1.70 \cdot 10^{-2}$ (−)	$4.15 \cdot 10^{-1} \pm 8.41 \cdot 10^{-3}$
	3	$3.73 \cdot 10^{-1} \pm 3.96 \cdot 10^{-2}$ (−)	**$4.23 \cdot 10^{-1} \pm 3.73 \cdot 10^{-3}$** (+)	$4.06 \cdot 10^{-1} \pm 1.32 \cdot 10^{-2}$ (≈)	$4.14 \cdot 10^{-1} \pm 5.39 \cdot 10^{-3}$
	4	$3.96 \cdot 10^{-1} \pm 1.95 \cdot 10^{-2}$ (−)	**$4.23 \cdot 10^{-1} \pm 1.86 \cdot 10^{-3}$** (+)	$3.86 \cdot 10^{-1} \pm 1.39 \cdot 10^{-2}$ (−)	$4.11 \cdot 10^{-1} \pm 7.66 \cdot 10^{-3}$
	5	$3.72 \cdot 10^{-1} \pm 6.19 \cdot 10^{-2}$ (−)	**$4.22 \cdot 10^{-1} \pm 2.55 \cdot 10^{-3}$** (+)	$3.84 \cdot 10^{-1} \pm 2.23 \cdot 10^{-2}$ (−)	$4.14 \cdot 10^{-1} \pm 1.11 \cdot 10^{-2}$
	6	$3.83 \cdot 10^{-1} \pm 3.70 \cdot 10^{-2}$ (≈)	**$4.24 \cdot 10^{-1} \pm 1.84 \cdot 10^{-3}$** (+)	$3.79 \cdot 10^{-1} \pm 1.73 \cdot 10^{-2}$ (−)	$4.02 \cdot 10^{-1} \pm 1.64 \cdot 10^{-2}$
	10	$3.92 \cdot 10^{-1} \pm 9.35 \cdot 10^{-3}$ (≈)	**$4.25 \cdot 10^{-1} \pm 2.64 \cdot 10^{-3}$** (+)	$3.76 \cdot 10^{-1} \pm 1.69 \cdot 10^{-2}$ (−)	$3.96 \cdot 10^{-1} \pm 5.10 \cdot 10^{-3}$
	20	$3.67 \cdot 10^{-1} \pm 7.21 \cdot 10^{-2}$ (≈)	**$4.24 \cdot 10^{-1} \pm 2.30 \cdot 10^{-3}$** (+)	$3.72 \cdot 10^{-1} \pm 1.20 \cdot 10^{-2}$ (≈)	$3.78 \cdot 10^{-1} \pm 1.24 \cdot 10^{-2}$
BICOP1	1	$6.38 \cdot 10^{-2} \pm 9.98 \cdot 10^{-2}$ (≈)	$9.60 \cdot 10^{-2} \pm 1.05 \cdot 10^{-1}$ (≈)	**$1.23 \cdot 10^{-1} \pm 1.62 \cdot 10^{-1}$** (≈)	$7.91 \cdot 10^{-2} \pm 1.16 \cdot 10^{-1}$
	2	$5.98 \cdot 10^{-1} \pm 1.92 \cdot 10^{-2}$ (+)	**$6.07 \cdot 10^{-1} \pm 1.34 \cdot 10^{-2}$** (+)	$3.17 \cdot 10^{-1} \pm 2.60 \cdot 10^{-1}$ (≈)	$4.16 \cdot 10^{-1} \pm 2.11 \cdot 10^{-1}$
	3	$6.29 \cdot 10^{-1} \pm 1.03 \cdot 10^{-2}$ (≈)	**$6.36 \cdot 10^{-1} \pm 4.84 \cdot 10^{-3}$** (≈)	$5.06 \cdot 10^{-1} \pm 2.54 \cdot 10^{-1}$ (≈)	$5.79 \cdot 10^{-1} \pm 8.34 \cdot 10^{-2}$
	4	$6.41 \cdot 10^{-1} \pm 6.41 \cdot 10^{-3}$ (≈)	**$6.43 \cdot 10^{-1} \pm 6.09 \cdot 10^{-3}$** (≈)	$6.34 \cdot 10^{-1} \pm 1.06 \cdot 10^{-2}$ (≈)	$6.09 \cdot 10^{-1} \pm 7.65 \cdot 10^{-2}$
	5	$6.49 \cdot 10^{-1} \pm 4.38 \cdot 10^{-3}$ (+)	**$6.50 \cdot 10^{-1} \pm 5.78 \cdot 10^{-3}$** (+)	$6.25 \cdot 10^{-1} \pm 1.39 \cdot 10^{-2}$ (≈)	$6.20 \cdot 10^{-1} \pm 1.31 \cdot 10^{-2}$
	6	$6.53 \cdot 10^{-1} \pm 4.50 \cdot 10^{-3}$ (+)	**$6.53 \cdot 10^{-1} \pm 3.46 \cdot 10^{-3}$** (+)	$5.89 \cdot 10^{-1} \pm 1.88 \cdot 10^{-2}$ (≈)	$5.99 \cdot 10^{-1} \pm 1.37 \cdot 10^{-2}$
	10	**$6.60 \cdot 10^{-1} \pm 1.08 \cdot 10^{-3}$** (+)	$6.59 \cdot 10^{-1} \pm 1.88 \cdot 10^{-3}$ (+)	$4.91 \cdot 10^{-1} \pm 4.87 \cdot 10^{-2}$ (≈)	$5.08 \cdot 10^{-1} \pm 3.28 \cdot 10^{-2}$
	20	$6.60 \cdot 10^{-1} \pm 8.35 \cdot 10^{-4}$ (+)	**$6.60 \cdot 10^{-1} \pm 7.77 \cdot 10^{-4}$** (+)	$2.98 \cdot 10^{-1} \pm 9.13 \cdot 10^{-2}$ (≈)	$2.68 \cdot 10^{-1} \pm 7.79 \cdot 10^{-2}$
BICOP2	1	$1.04 \cdot 10^{-1} \pm 2.31 \cdot 10^{-2}$ (−)	$1.17 \cdot 10^{-1} \pm 2.88 \cdot 10^{-2}$ (−)	$2.16 \cdot 10^{-1} \pm 4.01 \cdot 10^{-2}$ (−)	**$2.82 \cdot 10^{-1} \pm 1.79 \cdot 10^{-2}$**
	2	$1.06 \cdot 10^{-1} \pm 3.53 \cdot 10^{-2}$ (−)	$1.76 \cdot 10^{-1} \pm 3.44 \cdot 10^{-2}$ (−)	$2.15 \cdot 10^{-1} \pm 4.28 \cdot 10^{-2}$ (−)	**$3.11 \cdot 10^{-1} \pm 3.03 \cdot 10^{-2}$**
	3	$1.22 \cdot 10^{-1} \pm 3.01 \cdot 10^{-2}$ (−)	$1.53 \cdot 10^{-1} \pm 4.97 \cdot 10^{-2}$ (−)	$2.23 \cdot 10^{-1} \pm 4.93 \cdot 10^{-2}$ (−)	**$3.01 \cdot 10^{-1} \pm 5.25 \cdot 10^{-2}$**
	4	$1.21 \cdot 10^{-1} \pm 3.67 \cdot 10^{-2}$ (−)	$1.67 \cdot 10^{-1} \pm 5.22 \cdot 10^{-2}$ (≈)	$2.34 \cdot 10^{-1} \pm 5.56 \cdot 10^{-2}$ (≈)	**$2.50 \cdot 10^{-1} \pm 7.37 \cdot 10^{-2}$**
	5	$1.27 \cdot 10^{-1} \pm 4.19 \cdot 10^{-2}$ (−)	$1.77 \cdot 10^{-1} \pm 4.21 \cdot 10^{-2}$ (≈)	**$2.53 \cdot 10^{-1} \pm 3.15 \cdot 10^{-2}$** (≈)	$2.24 \cdot 10^{-1} \pm 6.76 \cdot 10^{-2}$
	6	$1.26 \cdot 10^{-1} \pm 3.79 \cdot 10^{-2}$ (−)	$1.55 \cdot 10^{-1} \pm 4.65 \cdot 10^{-2}$ (≈)	**$2.65 \cdot 10^{-1} \pm 1.68 \cdot 10^{-2}$** (≈)	$2.13 \cdot 10^{-1} \pm 6.50 \cdot 10^{-2}$
	10	$1.53 \cdot 10^{-1} \pm 3.98 \cdot 10^{-2}$ (−)	$1.45 \cdot 10^{-1} \pm 3.91 \cdot 10^{-2}$ (−)	$2.38 \cdot 10^{-1} \pm 2.77 \cdot 10^{-2}$ (≈)	**$2.44 \cdot 10^{-1} \pm 4.47 \cdot 10^{-2}$**
	20	$1.54 \cdot 10^{-1} \pm 4.41 \cdot 10^{-2}$ (−)	$1.50 \cdot 10^{-1} \pm 4.22 \cdot 10^{-2}$ (−)	$2.25 \cdot 10^{-1} \pm 2.07 \cdot 10^{-2}$ (−)	**$2.72 \cdot 10^{-1} \pm 1.60 \cdot 10^{-2}$**
MW1	1	$0.00 \cdot 10^{0} \pm 0.00 \cdot 10^{0}$ (−)	$2.73 \cdot 10^{-1} \pm 4.54 \cdot 10^{-2}$ (−)	$1.66 \cdot 10^{-2} \pm 3.27 \cdot 10^{-2}$ (−)	**$3.99 \cdot 10^{-1} \pm 5.85 \cdot 10^{-5}$**
	2	$2.40 \cdot 10^{-1} \pm 6.35 \cdot 10^{-2}$ (−)	$3.37 \cdot 10^{-1} \pm 5.49 \cdot 10^{-3}$ (−)	$1.92 \cdot 10^{-1} \pm 1.13 \cdot 10^{-1}$ (−)	**$3.98 \cdot 10^{-1} \pm 8.02 \cdot 10^{-5}$**
	3	$2.82 \cdot 10^{-1} \pm 4.11 \cdot 10^{-2}$ (−)	$3.40 \cdot 10^{-1} \pm 1.00 \cdot 10^{-2}$ (−)	$3.10 \cdot 10^{-1} \pm 7.11 \cdot 10^{-2}$ (−)	**$3.98 \cdot 10^{-1} \pm 1.55 \cdot 10^{-4}$**
	4	$3.24 \cdot 10^{-1} \pm 3.86 \cdot 10^{-2}$ (−)	$3.51 \cdot 10^{-1} \pm 1.15 \cdot 10^{-2}$ (−)	$2.20 \cdot 10^{-1} \pm 1.61 \cdot 10^{-1}$ (−)	**$3.98 \cdot 10^{-1} \pm 1.56 \cdot 10^{-4}$**
	5	$3.49 \cdot 10^{-1} \pm 1.34 \cdot 10^{-2}$ (−)	$3.66 \cdot 10^{-1} \pm 6.28 \cdot 10^{-3}$ (−)	$2.09 \cdot 10^{-1} \pm 1.73 \cdot 10^{-1}$ (−)	**$3.98 \cdot 10^{-1} \pm 1.70 \cdot 10^{-4}$**
	6	$3.56 \cdot 10^{-1} \pm 1.61 \cdot 10^{-2}$ (−)	$3.80 \cdot 10^{-1} \pm 5.43 \cdot 10^{-3}$ (−)	$2.55 \cdot 10^{-1} \pm 1.25 \cdot 10^{-1}$ (−)	**$3.98 \cdot 10^{-1} \pm 5.34 \cdot 10^{-4}$**
	10	$3.56 \cdot 10^{-1} \pm 2.92 \cdot 10^{-2}$ (−)	$3.90 \cdot 10^{-1} \pm 1.81 \cdot 10^{-3}$ (−)	$1.63 \cdot 10^{-1} \pm 1.23 \cdot 10^{-1}$ (−)	**$3.97 \cdot 10^{-1} \pm 1.10 \cdot 10^{-3}$**
	20	$3.72 \cdot 10^{-1} \pm 1.15 \cdot 10^{-2}$ (≈)	**$3.93 \cdot 10^{-1} \pm 1.16 \cdot 10^{-3}$** (+)	$2.09 \cdot 10^{-1} \pm 1.15 \cdot 10^{-1}$ (≈)	$2.73 \cdot 10^{-1} \pm 1.46 \cdot 10^{-1}$
MW2	1	$2.86 \cdot 10^{-2} \pm 5.73 \cdot 10^{-2}$ (−)	**$4.24 \cdot 10^{-1} \pm 1.51 \cdot 10^{-2}$** (+)	$1.60 \cdot 10^{-1} \pm 6.38 \cdot 10^{-2}$ (−)	$3.85 \cdot 10^{-1} \pm 2.32 \cdot 10^{-2}$
	2	$2.63 \cdot 10^{-1} \pm 6.16 \cdot 10^{-2}$ (−)	**$4.33 \cdot 10^{-1} \pm 6.25 \cdot 10^{-3}$** (≈)	$1.82 \cdot 10^{-1} \pm 1.22 \cdot 10^{-1}$ (−)	$4.19 \cdot 10^{-1} \pm 2.06 \cdot 10^{-2}$
	3	$2.93 \cdot 10^{-1} \pm 8.71 \cdot 10^{-2}$ (−)	**$4.41 \cdot 10^{-1} \pm 8.60 \cdot 10^{-3}$** (≈)	$1.98 \cdot 10^{-1} \pm 1.12 \cdot 10^{-1}$ (−)	$4.00 \cdot 10^{-1} \pm 6.68 \cdot 10^{-2}$
	4	$3.42 \cdot 10^{-1} \pm 8.05 \cdot 10^{-2}$ (≈)	**$4.40 \cdot 10^{-1} \pm 8.97 \cdot 10^{-3}$** (+)	$1.66 \cdot 10^{-1} \pm 1.03 \cdot 10^{-1}$ (−)	$3.47 \cdot 10^{-1} \pm 7.45 \cdot 10^{-2}$
	5	$3.38 \cdot 10^{-1} \pm 7.90 \cdot 10^{-2}$ (−)	**$4.42 \cdot 10^{-1} \pm 8.72 \cdot 10^{-3}$** (+)	$1.35 \cdot 10^{-1} \pm 7.24 \cdot 10^{-2}$ (−)	$3.96 \cdot 10^{-1} \pm 5.05 \cdot 10^{-2}$
	6	$3.40 \cdot 10^{-1} \pm 7.84 \cdot 10^{-2}$ (−)	**$4.42 \cdot 10^{-1} \pm 7.95 \cdot 10^{-3}$** (+)	$1.43 \cdot 10^{-1} \pm 9.95 \cdot 10^{-2}$ (−)	$4.11 \cdot 10^{-1} \pm 2.91 \cdot 10^{-2}$
	10	$3.20 \cdot 10^{-1} \pm 1.07 \cdot 10^{-1}$ (≈)	**$4.45 \cdot 10^{-1} \pm 1.08 \cdot 10^{-2}$** (+)	$1.04 \cdot 10^{-1} \pm 8.28 \cdot 10^{-2}$ (−)	$3.78 \cdot 10^{-1} \pm 3.47 \cdot 10^{-2}$
	20	$3.33 \cdot 10^{-1} \pm 9.66 \cdot 10^{-2}$ (≈)	**$4.49 \cdot 10^{-1} \pm 9.99 \cdot 10^{-3}$** (+)	$1.31 \cdot 10^{-1} \pm 1.09 \cdot 10^{-1}$ (−)	$3.10 \cdot 10^{-1} \pm 4.45 \cdot 10^{-2}$
MW3	1	$1.04 \cdot 10^{-1} \pm 1.48 \cdot 10^{-1}$ (−)	$4.10 \cdot 10^{-1} \pm 9.41 \cdot 10^{-3}$ (−)	$3.72 \cdot 10^{-1} \pm 2.46 \cdot 10^{-2}$ (−)	**$4.50 \cdot 10^{-1} \pm 9.80 \cdot 10^{-4}$**
	2	$4.06 \cdot 10^{-1} \pm 1.30 \cdot 10^{-2}$ (−)	$4.22 \cdot 10^{-1} \pm 3.38 \cdot 10^{-3}$ (−)	$4.07 \cdot 10^{-1} \pm 8.56 \cdot 10^{-3}$ (−)	**$4.51 \cdot 10^{-1} \pm 1.70 \cdot 10^{-3}$**
	3	$4.22 \cdot 10^{-1} \pm 6.32 \cdot 10^{-3}$ (−)	$4.29 \cdot 10^{-1} \pm 3.08 \cdot 10^{-3}$ (−)	$4.29 \cdot 10^{-1} \pm 1.08 \cdot 10^{-2}$ (−)	**$4.52 \cdot 10^{-1} \pm 4.92 \cdot 10^{-4}$**
	4	$4.22 \cdot 10^{-1} \pm 2.48 \cdot 10^{-3}$ (−)	$4.32 \cdot 10^{-1} \pm 3.15 \cdot 10^{-3}$ (−)	$4.43 \cdot 10^{-1} \pm 5.59 \cdot 10^{-3}$ (−)	**$4.52 \cdot 10^{-1} \pm 2.32 \cdot 10^{-4}$**
	5	$4.26 \cdot 10^{-1} \pm 4.97 \cdot 10^{-3}$ (−)	$4.36 \cdot 10^{-1} \pm 2.39 \cdot 10^{-3}$ (−)	$4.43 \cdot 10^{-1} \pm 3.76 \cdot 10^{-3}$ (−)	**$4.51 \cdot 10^{-1} \pm 1.01 \cdot 10^{-3}$**
	6	$4.25 \cdot 10^{-1} \pm 2.38 \cdot 10^{-3}$ (−)	$4.37 \cdot 10^{-1} \pm 3.88 \cdot 10^{-3}$ (−)	$4.42 \cdot 10^{-1} \pm 3.55 \cdot 10^{-3}$ (−)	**$4.50 \cdot 10^{-1} \pm 7.45 \cdot 10^{-4}$**
	10	$4.29 \cdot 10^{-1} \pm 3.28 \cdot 10^{-3}$ (−)	$4.41 \cdot 10^{-1} \pm 1.21 \cdot 10^{-3}$ (−)	$4.36 \cdot 10^{-1} \pm 1.97 \cdot 10^{-3}$ (−)	**$4.48 \cdot 10^{-1} \pm 6.46 \cdot 10^{-4}$**
	20	$4.28 \cdot 10^{-1} \pm 4.92 \cdot 10^{-3}$ (−)	$4.40 \cdot 10^{-1} \pm 1.41 \cdot 10^{-3}$ (−)	$4.29 \cdot 10^{-1} \pm 2.55 \cdot 10^{-3}$ (−)	**$4.44 \cdot 10^{-1} \pm 7.06 \cdot 10^{-4}$**
MW11	1	$6.65 \cdot 10^{-1} \pm 2.63 \cdot 10^{-1}$ (−)	**$1.36 \cdot 10^{0} \pm 4.41 \cdot 10^{-2}$** (+)	$9.80 \cdot 10^{-1} \pm 3.80 \cdot 10^{-1}$ (≈)	$1.10 \cdot 10^{0} \pm 1.99 \cdot 10^{-1}$
	2	$1.17 \cdot 10^{0} \pm 1.75 \cdot 10^{-1}$ (≈)	**$1.42 \cdot 10^{0} \pm 2.43 \cdot 10^{-2}$** (+)	$9.82 \cdot 10^{-1} \pm 1.74 \cdot 10^{-1}$ (−)	$1.17 \cdot 10^{0} \pm 1.55 \cdot 10^{-1}$
	3	$1.09 \cdot 10^{0} \pm 2.30 \cdot 10^{-1}$ (−)	$1.44 \cdot 10^{0} \pm 1.83 \cdot 10^{-2}$ (−)	$9.92 \cdot 10^{-1} \pm 1.97 \cdot 10^{-1}$ (−)	**$1.49 \cdot 10^{0} \pm 4.23 \cdot 10^{-2}$**
	4	$1.03 \cdot 10^{0} \pm 2.44 \cdot 10^{-1}$ (−)	$1.46 \cdot 10^{0} \pm 1.81 \cdot 10^{-2}$ (−)	$9.99 \cdot 10^{-1} \pm 1.23 \cdot 10^{-1}$ (−)	**$1.51 \cdot 10^{0} \pm 1.40 \cdot 10^{-2}$**
	5	$1.04 \cdot 10^{0} \pm 2.41 \cdot 10^{-1}$ (−)	$1.46 \cdot 10^{0} \pm 9.42 \cdot 10^{-3}$ (−)	$1.06 \cdot 10^{0} \pm 1.80 \cdot 10^{-1}$ (−)	**$1.52 \cdot 10^{0} \pm 8.07 \cdot 10^{-3}$**
	6	$9.08 \cdot 10^{-1} \pm 1.57 \cdot 10^{-1}$ (−)	$1.48 \cdot 10^{0} \pm 8.50 \cdot 10^{-3}$ (−)	$9.75 \cdot 10^{-1} \pm 2.81 \cdot 10^{-1}$ (−)	**$1.52 \cdot 10^{0} \pm 1.26 \cdot 10^{-2}$**
	10	$9.52 \cdot 10^{-1} \pm 2.21 \cdot 10^{-1}$ (−)	$1.49 \cdot 10^{0} \pm 8.09 \cdot 10^{-3}$ (−)	$8.27 \cdot 10^{-1} \pm 1.73 \cdot 10^{-1}$ (−)	**$1.52 \cdot 10^{0} \pm 5.60 \cdot 10^{-3}$**
	20	$8.02 \cdot 10^{-1} \pm 1.80 \cdot 10^{-1}$ (−)	$1.49 \cdot 10^{0} \pm 1.53 \cdot 10^{-2}$ (−)	$8.78 \cdot 10^{-1} \pm 5.96 \cdot 10^{-2}$ (−)	**$1.50 \cdot 10^{0} \pm 6.86 \cdot 10^{-3}$**
TRICOP	1	$4.47 \cdot 10^{1} \pm 2.03 \cdot 10^{0}$ (−)	$4.57 \cdot 10^{1} \pm 1.19 \cdot 10^{0}$ (−)	**$4.97 \cdot 10^{1} \pm 6.30 \cdot 10^{-3}$** (≈)	$4.97 \cdot 10^{1} \pm 3.81 \cdot 10^{-2}$
	2	$4.19 \cdot 10^{1} \pm 1.56 \cdot 10^{0}$ (−)	$4.55 \cdot 10^{1} \pm 7.37 \cdot 10^{-1}$ (−)	$4.96 \cdot 10^{1} \pm 2.76 \cdot 10^{-2}$ (≈)	**$4.97 \cdot 10^{1} \pm 3.93 \cdot 10^{-2}$**
	3	$4.31 \cdot 10^{1} \pm 1.75 \cdot 10^{0}$ (−)	$4.63 \cdot 10^{1} \pm 5.46 \cdot 10^{-1}$ (−)	**$4.97 \cdot 10^{1} \pm 1.93 \cdot 10^{-2}$** (+)	$4.96 \cdot 10^{1} \pm 3.41 \cdot 10^{-2}$
	4	$4.31 \cdot 10^{1} \pm 1.50 \cdot 10^{0}$ (−)	$4.63 \cdot 10^{1} \pm 6.92 \cdot 10^{-1}$ (−)	**$4.96 \cdot 10^{1} \pm 4.30 \cdot 10^{-2}$** (≈)	$4.96 \cdot 10^{1} \pm 3.22 \cdot 10^{-2}$
	5	$4.26 \cdot 10^{1} \pm 1.45 \cdot 10^{0}$ (−)	$4.66 \cdot 10^{1} \pm 3.57 \cdot 10^{-1}$ (−)	**$4.97 \cdot 10^{1} \pm 2.46 \cdot 10^{-2}$** (≈)	$4.97 \cdot 10^{1} \pm 2.45 \cdot 10^{-2}$
	6	$4.36 \cdot 10^{1} \pm 1.48 \cdot 10^{0}$ (−)	$4.63 \cdot 10^{1} \pm 5.04 \cdot 10^{-1}$ (−)	**$4.97 \cdot 10^{1} \pm 3.34 \cdot 10^{-2}$** (≈)	$4.97 \cdot 10^{1} \pm 5.05 \cdot 10^{-2}$
	10	$4.40 \cdot 10^{1} \pm 1.43 \cdot 10^{0}$ (−)	$4.71 \cdot 10^{1} \pm 3.92 \cdot 10^{-1}$ (−)	**$4.97 \cdot 10^{1} \pm 3.00 \cdot 10^{-2}$** (≈)	$4.97 \cdot 10^{1} \pm 1.95 \cdot 10^{-2}$
	20	$4.55 \cdot 10^{1} \pm 8.60 \cdot 10^{-1}$ (−)	$4.76 \cdot 10^{1} \pm 3.49 \cdot 10^{-1}$ (−)	$4.95 \cdot 10^{1} \pm 4.39 \cdot 10^{-2}$ (≈)	**$4.95 \cdot 10^{1} \pm 2.77 \cdot 10^{-2}$**
SPD	1	$5.04 \cdot 10^{9} \pm 8.11 \cdot 10^{7}$ (−)	$4.95 \cdot 10^{9} \pm 1.07 \cdot 10^{8}$ (−)	$5.87 \cdot 10^{9} \pm 1.68 \cdot 10^{7}$ (−)	**$6.01 \cdot 10^{9} \pm 1.95 \cdot 10^{6}$**
	2	$4.88 \cdot 10^{9} \pm 1.58 \cdot 10^{8}$ (−)	$5.05 \cdot 10^{9} \pm 6.73 \cdot 10^{7}$ (−)	$5.91 \cdot 10^{9} \pm 2.36 \cdot 10^{7}$ (−)	**$6.02 \cdot 10^{9} \pm 3.06 \cdot 10^{6}$**
	3	$5.02 \cdot 10^{9} \pm 7.48 \cdot 10^{7}$ (−)	$5.08 \cdot 10^{9} \pm 8.10 \cdot 10^{7}$ (−)	$5.93 \cdot 10^{9} \pm 9.35 \cdot 10^{6}$ (−)	**$6.01 \cdot 10^{9} \pm 2.47 \cdot 10^{6}$**
	4	$4.97 \cdot 10^{9} \pm 1.19 \cdot 10^{8}$ (−)	$5.02 \cdot 10^{9} \pm 7.63 \cdot 10^{7}$ (−)	$5.93 \cdot 10^{9} \pm 6.67 \cdot 10^{6}$ (−)	**$6.01 \cdot 10^{9} \pm 3.71 \cdot 10^{6}$**
	5	$5.06 \cdot 10^{9} \pm 8.50 \cdot 10^{7}$ (−)	$5.03 \cdot 10^{9} \pm 1.44 \cdot 10^{8}$ (−)	$5.91 \cdot 10^{9} \pm 1.68 \cdot 10^{7}$ (−)	**$6.01 \cdot 10^{9} \pm 3.20 \cdot 10^{6}$**
	6	$5.07 \cdot 10^{9} \pm 5.48 \cdot 10^{7}$ (−)	$5.05 \cdot 10^{9} \pm 5.88 \cdot 10^{7}$ (−)	$5.91 \cdot 10^{9} \pm 1.17 \cdot 10^{7}$ (−)	**$6.00 \cdot 10^{9} \pm 5.75 \cdot 10^{6}$**
	10	$5.08 \cdot 10^{9} \pm 6.05 \cdot 10^{7}$ (−)	$5.06 \cdot 10^{9} \pm 9.73 \cdot 10^{7}$ (−)	$5.86 \cdot 10^{9} \pm 2.25 \cdot 10^{7}$ (−)	**$5.99 \cdot 10^{9} \pm 3.85 \cdot 10^{6}$**
	20	$5.08 \cdot 10^{9} \pm 1.15 \cdot 10^{8}$ (−)	$5.04 \cdot 10^{9} \pm 8.77 \cdot 10^{7}$ (−)	$5.80 \cdot 10^{9} \pm 2.77 \cdot 10^{7}$ (−)	**$5.93 \cdot 10^{9} \pm 1.02 \cdot 10^{7}$**
CSI	1	$7.31 \cdot 10^{0} \pm 9.10 \cdot 10^{-2}$ (−)	$6.06 \cdot 10^{0} \pm 3.59 \cdot 10^{-1}$ (−)	$8.33 \cdot 10^{0} \pm 6.84 \cdot 10^{-2}$ (≈)	**$8.35 \cdot 10^{0} \pm 9.34 \cdot 10^{-2}$**
	2	$7.11 \cdot 10^{0} \pm 1.34 \cdot 10^{-1}$ (−)	$7.04 \cdot 10^{0} \pm 7.90 \cdot 10^{-2}$ (−)	$8.45 \cdot 10^{0} \pm 1.70 \cdot 10^{-2}$ (≈)	**$8.46 \cdot 10^{0} \pm 2.16 \cdot 10^{-2}$**
	3	$7.20 \cdot 10^{0} \pm 1.01 \cdot 10^{-1}$ (−)	$7.12 \cdot 10^{0} \pm 1.45 \cdot 10^{-1}$ (−)	$8.47 \cdot 10^{0} \pm 1.32 \cdot 10^{-2}$ (−)	**$8.49 \cdot 10^{0} \pm 5.44 \cdot 10^{-3}$**
	4	$7.13 \cdot 10^{0} \pm 6.49 \cdot 10^{-2}$ (−)	$7.14 \cdot 10^{0} \pm 7.68 \cdot 10^{-2}$ (−)	$8.46 \cdot 10^{0} \pm 1.36 \cdot 10^{-2}$ (−)	**$8.48 \cdot 10^{0} \pm 9.85 \cdot 10^{-3}$**
	5	$7.09 \cdot 10^{0} \pm 6.13 \cdot 10^{-2}$ (−)	$7.10 \cdot 10^{0} \pm 1.75 \cdot 10^{-1}$ (−)	$8.46 \cdot 10^{0} \pm 1.11 \cdot 10^{-2}$ (−)	**$8.48 \cdot 10^{0} \pm 7.05 \cdot 10^{-3}$**
	6	$7.14 \cdot 10^{0} \pm 1.04 \cdot 10^{-1}$ (−)	$7.17 \cdot 10^{0} \pm 9.63 \cdot 10^{-2}$ (−)	$8.45 \cdot 10^{0} \pm 1.00 \cdot 10^{-2}$ (≈)	**$8.46 \cdot 10^{0} \pm 1.21 \cdot 10^{-2}$**
	10	$7.21 \cdot 10^{0} \pm 1.05 \cdot 10^{-1}$ (−)	$7.19 \cdot 10^{0} \pm 6.00 \cdot 10^{-2}$ (−)	$8.33 \cdot 10^{0} \pm 1.89 \cdot 10^{-2}$ (−)	**$8.35 \cdot 10^{0} \pm 1.28 \cdot 10^{-2}$**
	20	$7.10 \cdot 10^{0} \pm 1.68 \cdot 10^{-1}$ (−)	$7.22 \cdot 10^{0} \pm 1.10 \cdot 10^{-1}$ (−)	**$7.90 \cdot 10^{0} \pm 8.04 \cdot 10^{-2}$** (≈)	$7.86 \cdot 10^{0} \pm 4.02 \cdot 10^{-2}$
WP	1	$1.03 \cdot 10^{18} \pm 3.31 \cdot 10^{17}$ (−)	$1.00 \cdot 10^{18} \pm 2.70 \cdot 10^{17}$ (−)	$3.42 \cdot 10^{18} \pm 8.57 \cdot 10^{15}$ (≈)	**$3.42 \cdot 10^{18} \pm 7.12 \cdot 10^{15}$**
	2	$1.47 \cdot 10^{18} \pm 2.17 \cdot 10^{17}$ (−)	$1.36 \cdot 10^{18} \pm 4.05 \cdot 10^{17}$ (−)	**$3.44 \cdot 10^{18} \pm 1.62 \cdot 10^{15}$** (+)	$3.42 \cdot 10^{18} \pm 5.17 \cdot 10^{15}$
	3	$1.61 \cdot 10^{18} \pm 2.10 \cdot 10^{17}$ (−)	$1.47 \cdot 10^{18} \pm 2.37 \cdot 10^{17}$ (−)	$3.42 \cdot 10^{18} \pm 1.04 \cdot 10^{16}$ (≈)	**$3.42 \cdot 10^{18} \pm 8.90 \cdot 10^{15}$**
	4	$1.70 \cdot 10^{18} \pm 1.89 \cdot 10^{17}$ (−)	$1.53 \cdot 10^{18} \pm 2.55 \cdot 10^{17}$ (−)	**$3.42 \cdot 10^{18} \pm 7.99 \cdot 10^{15}$** (≈)	$3.42 \cdot 10^{18} \pm 5.15 \cdot 10^{15}$
	5	$1.69 \cdot 10^{18} \pm 2.27 \cdot 10^{17}$ (−)	$1.57 \cdot 10^{18} \pm 3.08 \cdot 10^{17}$ (−)	**$3.42 \cdot 10^{18} \pm 6.03 \cdot 10^{15}$** (+)	$3.41 \cdot 10^{18} \pm 1.46 \cdot 10^{16}$
	6	$1.86 \cdot 10^{18} \pm 1.69 \cdot 10^{17}$ (−)	$1.58 \cdot 10^{18} \pm 3.50 \cdot 10^{17}$ (−)	**$3.42 \cdot 10^{18} \pm 5.69 \cdot 10^{15}$** (+)	$3.40 \cdot 10^{18} \pm 1.47 \cdot 10^{16}$
	10	$1.97 \cdot 10^{18} \pm 2.78 \cdot 10^{17}$ (−)	$1.70 \cdot 10^{18} \pm 2.86 \cdot 10^{17}$ (−)	$3.37 \cdot 10^{18} \pm 1.51 \cdot 10^{16}$ (−)	**$3.40 \cdot 10^{18} \pm 6.14 \cdot 10^{15}$**
	20	$1.90 \cdot 10^{18} \pm 2.51 \cdot 10^{17}$ (−)	$1.79 \cdot 10^{18} \pm 2.84 \cdot 10^{17}$ (−)	**$3.37 \cdot 10^{18} \pm 7.01 \cdot 10^{15}$** (≈)	$3.37 \cdot 10^{18} \pm 1.09 \cdot 10^{16}$
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	$1.77 \cdot 10^{-2} \pm 2.89 \cdot 10^{-3}$ (–)	$2.15 \cdot 10^{-2} \pm 3.60 \cdot 10^{-3}$ (–)	$2.06 \cdot 10^{-3} \pm 1.44 \cdot 10^{-5}$ (≈)	**$2.06 \cdot 10^{-3} \pm 1.08 \cdot 10^{-5}$**
	2	$1.95 \cdot 10^{-2} \pm 1.39 \cdot 10^{-3}$ (–)	$2.21 \cdot 10^{-2} \pm 2.89 \cdot 10^{-3}$ (–)	**$2.12 \cdot 10^{-3} \pm 1.14 \cdot 10^{-5}$** (≈)	$2.12 \cdot 10^{-3} \pm 1.52 \cdot 10^{-5}$
	3	$1.81 \cdot 10^{-2} \pm 2.50 \cdot 10^{-3}$ (–)	$2.03 \cdot 10^{-2} \pm 2.92 \cdot 10^{-3}$ (–)	$2.13 \cdot 10^{-3} \pm 3.93 \cdot 10^{-5}$ (≈)	**$2.12 \cdot 10^{-3} \pm 2.83 \cdot 10^{-5}$**
	4	$1.75 \cdot 10^{-2} \pm 1.61 \cdot 10^{-3}$ (–)	$2.11 \cdot 10^{-2} \pm 1.95 \cdot 10^{-3}$ (–)	**$2.12 \cdot 10^{-3} \pm 4.18 \cdot 10^{-5}$** (≈)	$2.15 \cdot 10^{-3} \pm 4.39 \cdot 10^{-5}$
	5	$1.94 \cdot 10^{-2} \pm 2.28 \cdot 10^{-3}$ (–)	$2.05 \cdot 10^{-2} \pm 2.19 \cdot 10^{-3}$ (–)	$2.14 \cdot 10^{-3} \pm 2.59 \cdot 10^{-5}$ (≈)	**$2.13 \cdot 10^{-3} \pm 3.13 \cdot 10^{-5}$**
	6	$1.84 \cdot 10^{-2} \pm 1.03 \cdot 10^{-3}$ (–)	$1.79 \cdot 10^{-2} \pm 2.52 \cdot 10^{-3}$ (–)	$2.09 \cdot 10^{-3} \pm 5.02 \cdot 10^{-5}$ (≈)	**$2.09 \cdot 10^{-3} \pm 3.86 \cdot 10^{-5}$**
	10	$1.85 \cdot 10^{-2} \pm 1.72 \cdot 10^{-3}$ (–)	$1.97 \cdot 10^{-2} \pm 2.28 \cdot 10^{-3}$ (–)	**$2.40 \cdot 10^{-3} \pm 5.77 \cdot 10^{-5}$** (≈)	$2.42 \cdot 10^{-3} \pm 5.79 \cdot 10^{-5}$
	20	$1.67 \cdot 10^{-2} \pm 2.40 \cdot 10^{-3}$ (–)	$1.89 \cdot 10^{-2} \pm 2.32 \cdot 10^{-3}$ (–)	**$3.03 \cdot 10^{-3} \pm 5.58 \cdot 10^{-5}$** (≈)	$3.06 \cdot 10^{-3} \pm 6.72 \cdot 10^{-5}$
CEXP	1	$1.79 \cdot 10^{-2} \pm 2.35 \cdot 10^{-3}$ (–)	$1.83 \cdot 10^{-2} \pm 6.60 \cdot 10^{-3}$ (–)	$2.54 \cdot 10^{-3} \pm 4.97 \cdot 10^{-5}$ (–)	**$2.17 \cdot 10^{-3} \pm 1.17 \cdot 10^{-5}$**
	2	$2.50 \cdot 10^{-2} \pm 4.41 \cdot 10^{-3}$ (–)	$2.60 \cdot 10^{-2} \pm 6.07 \cdot 10^{-3}$ (–)	$2.43 \cdot 10^{-3} \pm 1.16 \cdot 10^{-4}$ (–)	**$2.35 \cdot 10^{-3} \pm 4.84 \cdot 10^{-5}$**
	3	$2.50 \cdot 10^{-2} \pm 3.26 \cdot 10^{-3}$ (–)	$2.66 \cdot 10^{-2} \pm 3.94 \cdot 10^{-3}$ (–)	$2.17 \cdot 10^{-3} \pm 5.72 \cdot 10^{-5}$ (≈)	**$2.15 \cdot 10^{-3} \pm 8.76 \cdot 10^{-6}$**
	4	$2.57 \cdot 10^{-2} \pm 3.90 \cdot 10^{-3}$ (–)	$2.69 \cdot 10^{-2} \pm 3.73 \cdot 10^{-3}$ (–)	**$2.36 \cdot 10^{-3} \pm 1.49 \cdot 10^{-5}$** (≈)	$2.38 \cdot 10^{-3} \pm 4.91 \cdot 10^{-5}$
	5	$2.51 \cdot 10^{-2} \pm 3.15 \cdot 10^{-3}$ (–)	$2.73 \cdot 10^{-2} \pm 5.23 \cdot 10^{-3}$ (–)	**$2.45 \cdot 10^{-3} \pm 3.29 \cdot 10^{-5}$** (≈)	$2.46 \cdot 10^{-3} \pm 3.09 \cdot 10^{-5}$
	6	$2.52 \cdot 10^{-2} \pm 2.47 \cdot 10^{-3}$ (–)	$2.46 \cdot 10^{-2} \pm 3.33 \cdot 10^{-3}$ (–)	$2.34 \cdot 10^{-3} \pm 5.45 \cdot 10^{-5}$ (≈)	**$2.33 \cdot 10^{-3} \pm 4.39 \cdot 10^{-5}$**
	10	$2.67 \cdot 10^{-2} \pm 5.05 \cdot 10^{-3}$ (–)	$2.31 \cdot 10^{-2} \pm 2.66 \cdot 10^{-3}$ (–)	$2.88 \cdot 10^{-3} \pm 8.27 \cdot 10^{-5}$ (≈)	**$2.84 \cdot 10^{-3} \pm 6.72 \cdot 10^{-5}$**
	20	$2.81 \cdot 10^{-2} \pm 3.01 \cdot 10^{-3}$ (–)	$2.39 \cdot 10^{-2} \pm 3.53 \cdot 10^{-3}$ (–)	$5.00 \cdot 10^{-3} \pm 3.61 \cdot 10^{-4}$ (≈)	**$4.99 \cdot 10^{-3} \pm 4.43 \cdot 10^{-4}$**
SRN	1	$1.89 \cdot 10^{-2} \pm 1.70 \cdot 10^{-3}$ (–)	$1.54 \cdot 10^{-2} \pm 1.73 \cdot 10^{-3}$ (–)	$3.47 \cdot 10^{-3} \pm 4.37 \cdot 10^{-5}$ (–)	**$3.39 \cdot 10^{-3} \pm 3.39 \cdot 10^{-5}$**
	2	$3.06 \cdot 10^{-2} \pm 6.16 \cdot 10^{-3}$ (–)	$2.23 \cdot 10^{-2} \pm 4.60 \cdot 10^{-3}$ (–)	$3.66 \cdot 10^{-3} \pm 1.52 \cdot 10^{-4}$ (–)	**$3.32 \cdot 10^{-3} \pm 5.62 \cdot 10^{-5}$**
	3	$2.09 \cdot 10^{-2} \pm 2.54 \cdot 10^{-3}$ (–)	$2.02 \cdot 10^{-2} \pm 2.50 \cdot 10^{-3}$ (–)	$3.56 \cdot 10^{-3} \pm 5.90 \cdot 10^{-5}$ (–)	**$3.31 \cdot 10^{-3} \pm 3.23 \cdot 10^{-5}$**
	4	$2.45 \cdot 10^{-2} \pm 3.00 \cdot 10^{-3}$ (–)	$2.25 \cdot 10^{-2} \pm 2.36 \cdot 10^{-3}$ (–)	**$3.82 \cdot 10^{-3} \pm 5.09 \cdot 10^{-5}$** (+)	$4.02 \cdot 10^{-3} \pm 3.30 \cdot 10^{-5}$
	5	$2.24 \cdot 10^{-2} \pm 2.40 \cdot 10^{-3}$ (–)	$1.98 \cdot 10^{-2} \pm 3.02 \cdot 10^{-3}$ (–)	$3.25 \cdot 10^{-3} \pm 3.31 \cdot 10^{-5}$ (≈)	**$3.23 \cdot 10^{-3} \pm 4.57 \cdot 10^{-5}$**
	6	$2.17 \cdot 10^{-2} \pm 1.66 \cdot 10^{-3}$ (–)	$1.89 \cdot 10^{-2} \pm 1.60 \cdot 10^{-3}$ (–)	$3.25 \cdot 10^{-3} \pm 9.15 \cdot 10^{-5}$ (≈)	**$3.20 \cdot 10^{-3} \pm 7.84 \cdot 10^{-5}$**
	10	$2.20 \cdot 10^{-2} \pm 2.10 \cdot 10^{-3}$ (–)	$1.82 \cdot 10^{-2} \pm 2.36 \cdot 10^{-3}$ (–)	**$4.97 \cdot 10^{-3} \pm 3.33 \cdot 10^{-4}$** (≈)	$5.01 \cdot 10^{-3} \pm 2.88 \cdot 10^{-4}$
	20	$2.37 \cdot 10^{-2} \pm 4.11 \cdot 10^{-3}$ (–)	$1.75 \cdot 10^{-2} \pm 1.39 \cdot 10^{-3}$ (–)	**$5.58 \cdot 10^{-3} \pm 1.63 \cdot 10^{-4}$** (≈)	$5.68 \cdot 10^{-3} \pm 2.12 \cdot 10^{-4}$
TNK	1	$1.01 \cdot 10^{-1} \pm 2.12 \cdot 10^{-2}$ (–)	$1.42 \cdot 10^{-2} \pm 2.16 \cdot 10^{-3}$ (–)	$9.36 \cdot 10^{-3} \pm 1.10 \cdot 10^{-3}$ (–)	**$3.81 \cdot 10^{-3} \pm 3.15 \cdot 10^{-4}$**
	2	$6.63 \cdot 10^{-2} \pm 2.19 \cdot 10^{-2}$ (–)	$2.13 \cdot 10^{-2} \pm 3.89 \cdot 10^{-3}$ (–)	$9.14 \cdot 10^{-3} \pm 1.52 \cdot 10^{-3}$ (–)	**$2.68 \cdot 10^{-3} \pm 2.64 \cdot 10^{-4}$**
	3	$4.75 \cdot 10^{-2} \pm 1.65 \cdot 10^{-2}$ (–)	$1.97 \cdot 10^{-2} \pm 4.01 \cdot 10^{-3}$ (–)	$1.03 \cdot 10^{-2} \pm 2.01 \cdot 10^{-3}$ (–)	**$2.34 \cdot 10^{-3} \pm 1.24 \cdot 10^{-4}$**
	4	$4.29 \cdot 10^{-2} \pm 8.54 \cdot 10^{-3}$ (–)	$2.12 \cdot 10^{-2} \pm 3.24 \cdot 10^{-3}$ (–)	$8.88 \cdot 10^{-3} \pm 1.46 \cdot 10^{-3}$ (–)	**$2.26 \cdot 10^{-3} \pm 1.64 \cdot 10^{-4}$**
	5	$3.56 \cdot 10^{-2} \pm 6.84 \cdot 10^{-3}$ (–)	$1.99 \cdot 10^{-2} \pm 4.01 \cdot 10^{-3}$ (–)	$1.04 \cdot 10^{-2} \pm 1.77 \cdot 10^{-3}$ (–)	**$2.31 \cdot 10^{-3} \pm 7.63 \cdot 10^{-5}$**
	6	$3.27 \cdot 10^{-2} \pm 7.32 \cdot 10^{-3}$ (–)	$1.72 \cdot 10^{-2} \pm 2.30 \cdot 10^{-3}$ (–)	$1.14 \cdot 10^{-2} \pm 1.07 \cdot 10^{-3}$ (–)	**$2.33 \cdot 10^{-3} \pm 1.28 \cdot 10^{-4}$**
	10	$3.84 \cdot 10^{-2} \pm 5.80 \cdot 10^{-3}$ (–)	$2.14 \cdot 10^{-2} \pm 3.90 \cdot 10^{-3}$ (–)	$1.11 \cdot 10^{-2} \pm 1.54 \cdot 10^{-3}$ (–)	**$2.84 \cdot 10^{-3} \pm 1.19 \cdot 10^{-4}$**
	20	$4.53 \cdot 10^{-2} \pm 6.25 \cdot 10^{-3}$ (–)	$2.08 \cdot 10^{-2} \pm 3.36 \cdot 10^{-3}$ (–)	$1.85 \cdot 10^{-2} \pm 1.75 \cdot 10^{-3}$ (–)	**$5.97 \cdot 10^{-3} \pm 5.56 \cdot 10^{-4}$**
CTP1	1	$2.29 \cdot 10^{-2} \pm 4.86 \cdot 10^{-3}$ (–)	$1.87 \cdot 10^{-2} \pm 2.91 \cdot 10^{-3}$ (–)	**$4.39 \cdot 10^{-3} \pm 1.56 \cdot 10^{-4}$** (≈)	$4.48 \cdot 10^{-3} \pm 2.87 \cdot 10^{-4}$
	2	$3.43 \cdot 10^{-2} \pm 3.52 \cdot 10^{-3}$ (–)	$3.62 \cdot 10^{-2} \pm 8.82 \cdot 10^{-3}$ (–)	$6.82 \cdot 10^{-3} \pm 1.72 \cdot 10^{-3}$ (≈)	**$6.38 \cdot 10^{-3} \pm 1.41 \cdot 10^{-3}$**
	3	$3.13 \cdot 10^{-2} \pm 6.48 \cdot 10^{-3}$ (–)	$2.75 \cdot 10^{-2} \pm 6.81 \cdot 10^{-3}$ (–)	**$4.93 \cdot 10^{-3} \pm 4.39 \cdot 10^{-4}$** (≈)	$5.00 \cdot 10^{-3} \pm 8.95 \cdot 10^{-4}$
	4	$2.91 \cdot 10^{-2} \pm 2.52 \cdot 10^{-3}$ (–)	$3.26 \cdot 10^{-2} \pm 4.12 \cdot 10^{-3}$ (–)	$5.12 \cdot 10^{-3} \pm 4.82 \cdot 10^{-4}$ (≈)	**$5.06 \cdot 10^{-3} \pm 4.51 \cdot 10^{-4}$**
	5	$3.56 \cdot 10^{-2} \pm 6.89 \cdot 10^{-3}$ (–)	$3.38 \cdot 10^{-2} \pm 5.08 \cdot 10^{-3}$ (–)	**$5.24 \cdot 10^{-3} \pm 4.87 \cdot 10^{-4}$** (≈)	$5.24 \cdot 10^{-3} \pm 4.87 \cdot 10^{-4}$
	6	$3.17 \cdot 10^{-2} \pm 5.38 \cdot 10^{-3}$ (–)	$2.98 \cdot 10^{-2} \pm 2.93 \cdot 10^{-3}$ (–)	**$4.64 \cdot 10^{-3} \pm 2.99 \cdot 10^{-4}$** (≈)	$4.64 \cdot 10^{-3} \pm 2.99 \cdot 10^{-4}$
	10	$3.43 \cdot 10^{-2} \pm 6.21 \cdot 10^{-3}$ (–)	$2.85 \cdot 10^{-2} \pm 3.84 \cdot 10^{-3}$ (–)	**$5.59 \cdot 10^{-3} \pm 3.13 \cdot 10^{-4}$** (≈)	$5.71 \cdot 10^{-3} \pm 3.34 \cdot 10^{-4}$
	20	$3.47 \cdot 10^{-2} \pm 4.60 \cdot 10^{-3}$ (–)	$2.91 \cdot 10^{-2} \pm 5.41 \cdot 10^{-3}$ (–)	$8.78 \cdot 10^{-3} \pm 1.04 \cdot 10^{-3}$ (≈)	**$8.26 \cdot 10^{-3} \pm 1.26 \cdot 10^{-3}$**
C3DTLZ4	1	$3.69 \cdot 10^{-2} \pm 1.18 \cdot 10^{-2}$ (–)	$7.80 \cdot 10^{-2} \pm 3.13 \cdot 10^{-2}$ (–)	$4.38 \cdot 10^{-2} \pm 6.68 \cdot 10^{-3}$ (–)	**$5.71 \cdot 10^{-3} \pm 6.34 \cdot 10^{-4}$**
	2	$4.23 \cdot 10^{-2} \pm 1.83 \cdot 10^{-2}$ (–)	$3.22 \cdot 10^{-2} \pm 1.46 \cdot 10^{-2}$ (–)	$6.59 \cdot 10^{-2} \pm 7.44 \cdot 10^{-3}$ (–)	**$4.63 \cdot 10^{-3} \pm 1.08 \cdot 10^{-3}$**
	3	$2.12 \cdot 10^{-2} \pm 3.48 \cdot 10^{-3}$ (–)	$1.77 \cdot 10^{-2} \pm 4.18 \cdot 10^{-3}$ (–)	$4.79 \cdot 10^{-2} \pm 6.48 \cdot 10^{-3}$ (–)	**$2.48 \cdot 10^{-3} \pm 1.45 \cdot 10^{-4}$**
	4	$1.98 \cdot 10^{-2} \pm 1.49 \cdot 10^{-3}$ (–)	$1.33 \cdot 10^{-2} \pm 1.59 \cdot 10^{-3}$ (–)	$5.18 \cdot 10^{-2} \pm 4.42 \cdot 10^{-3}$ (–)	**$2.42 \cdot 10^{-3} \pm 1.64 \cdot 10^{-4}$**
	5	$1.86 \cdot 10^{-2} \pm 1.81 \cdot 10^{-3}$ (–)	$1.20 \cdot 10^{-2} \pm 1.58 \cdot 10^{-3}$ (–)	$4.75 \cdot 10^{-2} \pm 5.26 \cdot 10^{-3}$ (–)	**$2.23 \cdot 10^{-3} \pm 1.31 \cdot 10^{-4}$**
	6	$1.68 \cdot 10^{-2} \pm 1.94 \cdot 10^{-3}$ (–)	$1.00 \cdot 10^{-2} \pm 1.02 \cdot 10^{-3}$ (–)	$4.51 \cdot 10^{-2} \pm 8.05 \cdot 10^{-3}$ (–)	**$2.15 \cdot 10^{-3} \pm 7.18 \cdot 10^{-5}$**
	10	$1.66 \cdot 10^{-2} \pm 2.44 \cdot 10^{-3}$ (–)	$8.14 \cdot 10^{-3} \pm 7.08 \cdot 10^{-4}$ (–)	$5.22 \cdot 10^{-2} \pm 1.80 \cdot 10^{-2}$ (–)	**$2.33 \cdot 10^{-3} \pm 1.67 \cdot 10^{-4}$**
	20	$1.91 \cdot 10^{-2} \pm 2.98 \cdot 10^{-3}$ (–)	$8.04 \cdot 10^{-3} \pm 5.96 \cdot 10^{-4}$ (–)	$6.22 \cdot 10^{-2} \pm 1.54 \cdot 10^{-2}$ (–)	**$2.71 \cdot 10^{-3} \pm 1.66 \cdot 10^{-4}$**
OSY	1	$1.08 \cdot 10^{-1} \pm 7.19 \cdot 10^{-2}$ (–)	$4.87 \cdot 10^{-2} \pm 1.38 \cdot 10^{-2}$ (–)	**$9.78 \cdot 10^{-4} \pm 1.23 \cdot 10^{-4}$** (+)	$1.07 \cdot 10^{-3} \pm 4.00 \cdot 10^{-5}$
	2	$3.11 \cdot 10^{-2} \pm 1.13 \cdot 10^{-2}$ (–)	$3.23 \cdot 10^{-2} \pm 9.08 \cdot 10^{-3}$ (–)	$9.60 \cdot 10^{-4} \pm 4.80 \cdot 10^{-5}$ (–)	**$8.35 \cdot 10^{-4} \pm 8.30 \cdot 10^{-5}$**
	3	$2.05 \cdot 10^{-2} \pm 6.24 \cdot 10^{-3}$ (–)	$1.61 \cdot 10^{-2} \pm 3.05 \cdot 10^{-3}$ (–)	$9.91 \cdot 10^{-4} \pm 6.88 \cdot 10^{-5}$ (–)	**$9.38 \cdot 10^{-4} \pm 4.70 \cdot 10^{-5}$**
	4	$1.69 \cdot 10^{-2} \pm 4.00 \cdot 10^{-3}$ (–)	$1.54 \cdot 10^{-2} \pm 4.47 \cdot 10^{-3}$ (–)	**$1.24 \cdot 10^{-3} \pm 7.74 \cdot 10^{-5}$** (≈)	$1.26 \cdot 10^{-3} \pm 1.30 \cdot 10^{-4}$
	5	$1.35 \cdot 10^{-2} \pm 3.60 \cdot 10^{-3}$ (–)	$1.23 \cdot 10^{-2} \pm 2.71 \cdot 10^{-3}$ (–)	$1.54 \cdot 10^{-3} \pm 8.39 \cdot 10^{-5}$ (≈)	**$1.54 \cdot 10^{-3} \pm 1.02 \cdot 10^{-4}$**
	6	$1.21 \cdot 10^{-2} \pm 2.18 \cdot 10^{-3}$ (–)	$1.17 \cdot 10^{-2} \pm 2.25 \cdot 10^{-3}$ (–)	$2.14 \cdot 10^{-3} \pm 1.76 \cdot 10^{-4}$ (≈)	**$2.01 \cdot 10^{-3} \pm 1.61 \cdot 10^{-4}$**
	10	$1.14 \cdot 10^{-2} \pm 3.75 \cdot 10^{-3}$ (–)	$1.20 \cdot 10^{-2} \pm 3.15 \cdot 10^{-3}$ (–)	$7.60 \cdot 10^{-3} \pm 1.10 \cdot 10^{-3}$ (≈)	**$7.26 \cdot 10^{-3} \pm 8.61 \cdot 10^{-4}$**
	20	**$1.31 \cdot 10^{-2} \pm 7.56 \cdot 10^{-3}$** (+)	$1.31 \cdot 10^{-2} \pm 5.45 \cdot 10^{-3}$ (+)	$4.22 \cdot 10^{-2} \pm 1.06 \cdot 10^{-2}$ (≈)	$3.67 \cdot 10^{-2} \pm 5.01 \cdot 10^{-3}$
TBTD	1	$4.43 \cdot 10^{-2} \pm 3.30 \cdot 10^{-2}$ (–)	$2.20 \cdot 10^{-2} \pm 8.82 \cdot 10^{-3}$ (–)	$6.43 \cdot 10^{-3} \pm 9.72 \cdot 10^{-4}$ (–)	**$4.27 \cdot 10^{-3} \pm 2.41 \cdot 10^{-3}$**
	2	$2.87 \cdot 10^{-2} \pm 6.48 \cdot 10^{-3}$ (–)	$1.46 \cdot 10^{-2} \pm 5.57 \cdot 10^{-3}$ (–)	$1.10 \cdot 10^{-2} \pm 2.68 \cdot 10^{-3}$ (–)	**$5.94 \cdot 10^{-3} \pm 2.08 \cdot 10^{-3}$**
	3	$2.17 \cdot 10^{-2} \pm 3.41 \cdot 10^{-3}$ (–)	$1.40 \cdot 10^{-2} \pm 4.45 \cdot 10^{-3}$ (–)	$1.37 \cdot 10^{-2} \pm 5.00 \cdot 10^{-3}$ (–)	**$5.24 \cdot 10^{-3} \pm 1.22 \cdot 10^{-3}$**
	4	$1.56 \cdot 10^{-2} \pm 3.89 \cdot 10^{-3}$ (–)	$1.14 \cdot 10^{-2} \pm 2.67 \cdot 10^{-3}$ (–)	$1.46 \cdot 10^{-2} \pm 3.13 \cdot 10^{-3}$ (–)	**$6.47 \cdot 10^{-3} \pm 1.29 \cdot 10^{-3}$**
	5	$1.72 \cdot 10^{-2} \pm 3.94 \cdot 10^{-3}$ (–)	$1.20 \cdot 10^{-2} \pm 2.80 \cdot 10^{-3}$ (–)	$1.19 \cdot 10^{-2} \pm 2.65 \cdot 10^{-3}$ (–)	**$6.60 \cdot 10^{-3} \pm 1.12 \cdot 10^{-3}$**
	6	$1.38 \cdot 10^{-2} \pm 4.03 \cdot 10^{-3}$ (–)	$1.12 \cdot 10^{-2} \pm 2.78 \cdot 10^{-3}$ (≈)	$1.53 \cdot 10^{-2} \pm 4.81 \cdot 10^{-3}$ (–)	**$8.85 \cdot 10^{-3} \pm 2.62 \cdot 10^{-3}$**
	10	$1.20 \cdot 10^{-2} \pm 3.09 \cdot 10^{-3}$ (≈)	$1.07 \cdot 10^{-2} \pm 3.27 \cdot 10^{-3}$ (≈)	$1.56 \cdot 10^{-2} \pm 6.24 \cdot 10^{-3}$ (–)	**$1.02 \cdot 10^{-2} \pm 1.53 \cdot 10^{-3}$**
	20	$1.14 \cdot 10^{-2} \pm 1.54 \cdot 10^{-3}$ (≈)	**$9.82 \cdot 10^{-3} \pm 2.00 \cdot 10^{-3}$** (+)	$1.55 \cdot 10^{-2} \pm 3.37 \cdot 10^{-3}$ (≈)	$1.36 \cdot 10^{-2} \pm 3.73 \cdot 10^{-3}$
NBP	1	$1.83 \cdot 10^{-2} \pm 2.50 \cdot 10^{-3}$ (–)	$1.82 \cdot 10^{-2} \pm 4.90 \cdot 10^{-3}$ (–)	$3.76 \cdot 10^{-3} \pm 1.78 \cdot 10^{-4}$ (–)	**$2.33 \cdot 10^{-3} \pm 3.94 \cdot 10^{-5}$**
	2	$2.32 \cdot 10^{-2} \pm 4.35 \cdot 10^{-3}$ (–)	$2.27 \cdot 10^{-2} \pm 2.88 \cdot 10^{-3}$ (–)	$3.64 \cdot 10^{-3} \pm 4.00 \cdot 10^{-4}$ (–)	**$2.32 \cdot 10^{-3} \pm 5.64 \cdot 10^{-5}$**
	3	$2.00 \cdot 10^{-2} \pm 3.63 \cdot 10^{-3}$ (–)	$1.91 \cdot 10^{-2} \pm 1.80 \cdot 10^{-3}$ (–)	$3.64 \cdot 10^{-3} \pm 1.72 \cdot 10^{-4}$ (–)	**$2.46 \cdot 10^{-3} \pm 1.46 \cdot 10^{-5}$**
	4	$2.60 \cdot 10^{-2} \pm 4.15 \cdot 10^{-3}$ (–)	$2.13 \cdot 10^{-2} \pm 3.90 \cdot 10^{-3}$ (–)	$3.87 \cdot 10^{-3} \pm 2.37 \cdot 10^{-4}$ (–)	**$2.45 \cdot 10^{-3} \pm 2.87 \cdot 10^{-5}$**
	5	$2.23 \cdot 10^{-2} \pm 2.68 \cdot 10^{-3}$ (–)	$1.88 \cdot 10^{-2} \pm 2.01 \cdot 10^{-3}$ (–)	$4.41 \cdot 10^{-3} \pm 4.01 \cdot 10^{-4}$ (–)	**$2.88 \cdot 10^{-3} \pm 9.64 \cdot 10^{-5}$**
	6	$2.09 \cdot 10^{-2} \pm 2.49 \cdot 10^{-3}$ (–)	$1.98 \cdot 10^{-2} \pm 3.48 \cdot 10^{-3}$ (–)	$4.03 \cdot 10^{-3} \pm 3.02 \cdot 10^{-4}$ (–)	**$3.03 \cdot 10^{-3} \pm 7.37 \cdot 10^{-5}$**
	10	$2.11 \cdot 10^{-2} \pm 2.92 \cdot 10^{-3}$ (–)	$1.77 \cdot 10^{-2} \pm 2.28 \cdot 10^{-3}$ (–)	$5.38 \cdot 10^{-3} \pm 4.40 \cdot 10^{-4}$ (–)	**$3.58 \cdot 10^{-3} \pm 2.04 \cdot 10^{-4}$**
	20	$2.29 \cdot 10^{-2} \pm 3.26 \cdot 10^{-3}$ (–)	$1.94 \cdot 10^{-2} \pm 2.35 \cdot 10^{-3}$ (–)	$1.44 \cdot 10^{-2} \pm 3.86 \cdot 10^{-3}$ (–)	**$6.09 \cdot 10^{-3} \pm 3.19 \cdot 10^{-4}$**
DBD	1	$1.17 \cdot 10^{-2} \pm 3.63 \cdot 10^{-3}$ (–)	$2.43 \cdot 10^{-2} \pm 1.51 \cdot 10^{-2}$ (–)	**$1.35 \cdot 10^{-3} \pm 3.36 \cdot 10^{-4}$** (+)	$2.26 \cdot 10^{-3} \pm 1.40 \cdot 10^{-3}$
	2	$1.41 \cdot 10^{-2} \pm 2.84 \cdot 10^{-3}$ (–)	$2.06 \cdot 10^{-2} \pm 4.21 \cdot 10^{-3}$ (–)	**$1.24 \cdot 10^{-3} \pm 2.49 \cdot 10^{-4}$** (+)	$4.29 \cdot 10^{-3} \pm 1.55 \cdot 10^{-3}$
	3	$1.01 \cdot 10^{-2} \pm 1.49 \cdot 10^{-3}$ (–)	$2.01 \cdot 10^{-2} \pm 6.65 \cdot 10^{-3}$ (–)	$1.32 \cdot 10^{-3} \pm 3.12 \cdot 10^{-4}$ (–)	**$9.60 \cdot 10^{-4} \pm 5.64 \cdot 10^{-5}$**
	4	$1.39 \cdot 10^{-2} \pm 2.75 \cdot 10^{-3}$ (–)	$1.69 \cdot 10^{-2} \pm 2.64 \cdot 10^{-3}$ (–)	$2.89 \cdot 10^{-3} \pm 1.54 \cdot 10^{-3}$ (–)	**$1.18 \cdot 10^{-3} \pm 4.70 \cdot 10^{-5}$**
	5	$1.42 \cdot 10^{-2} \pm 4.50 \cdot 10^{-3}$ (–)	$1.75 \cdot 10^{-2} \pm 3.70 \cdot 10^{-3}$ (–)	$1.84 \cdot 10^{-3} \pm 1.04 \cdot 10^{-3}$ (≈)	**$1.31 \cdot 10^{-3} \pm 8.45 \cdot 10^{-5}$**
	6	$1.07 \cdot 10^{-2} \pm 1.31 \cdot 10^{-3}$ (–)	$2.10 \cdot 10^{-2} \pm 8.34 \cdot 10^{-3}$ (–)	**$2.08 \cdot 10^{-3} \pm 7.46 \cdot 10^{-4}$** (≈)	$2.21 \cdot 10^{-3} \pm 7.33 \cdot 10^{-4}$
	10	$1.22 \cdot 10^{-2} \pm 2.49 \cdot 10^{-3}$ (–)	$1.72 \cdot 10^{-2} \pm 4.16 \cdot 10^{-3}$ (–)	$2.62 \cdot 10^{-3} \pm 5.78 \cdot 10^{-4}$ (≈)	**$2.33 \cdot 10^{-3} \pm 2.25 \cdot 10^{-4}$**
	20	$1.30 \cdot 10^{-2} \pm 3.30 \cdot 10^{-3}$ (–)	$2.06 \cdot 10^{-2} \pm 5.11 \cdot 10^{-3}$ (–)	**$3.74 \cdot 10^{-3} \pm 2.29 \cdot 10^{-4}$** (+)	$4.09 \cdot 10^{-3} \pm 3.45 \cdot 10^{-4}$
SRD	1	$4.42 \cdot 10^{-3} \pm 3.15 \cdot 10^{-3}$ (≈)	$1.86 \cdot 10^{-2} \pm 9.63 \cdot 10^{-3}$ (–)	**$6.51 \cdot 10^{-4} \pm 9.70 \cdot 10^{-5}$** (+)	$2.45 \cdot 10^{-3} \pm 9.97 \cdot 10^{-4}$
	2	$2.26 \cdot 10^{-3} \pm 1.37 \cdot 10^{-3}$ (≈)	$2.09 \cdot 10^{-3} \pm 7.84 \cdot 10^{-4}$ (–)	**$1.24 \cdot 10^{-3} \pm 1.31 \cdot 10^{-4}$** (+)	$1.40 \cdot 10^{-3} \pm 1.77 \cdot 10^{-4}$
	3	$1.22 \cdot 10^{-3} \pm 4.50 \cdot 10^{-4}$ (≈)	**$1.03 \cdot 10^{-3} \pm 2.35 \cdot 10^{-4}$** (+)	$1.91 \cdot 10^{-3} \pm 1.38 \cdot 10^{-4}$ (–)	$1.50 \cdot 10^{-3} \pm 1.16 \cdot 10^{-4}$
	4	$1.15 \cdot 10^{-3} \pm 1.29 \cdot 10^{-4}$ (+)	**$1.09 \cdot 10^{-3} \pm 1.64 \cdot 10^{-4}$** (+)	$2.11 \cdot 10^{-3} \pm 3.10 \cdot 10^{-4}$ (–)	$1.58 \cdot 10^{-3} \pm 1.41 \cdot 10^{-4}$
	5	$1.09 \cdot 10^{-3} \pm 1.41 \cdot 10^{-4}$ (+)	**$9.05 \cdot 10^{-4} \pm 8.43 \cdot 10^{-5}$** (+)	$2.20 \cdot 10^{-3} \pm 3.38 \cdot 10^{-4}$ (–)	$1.56 \cdot 10^{-3} \pm 9.89 \cdot 10^{-5}$
	6	$1.13 \cdot 10^{-3} \pm 1.61 \cdot 10^{-4}$ (+)	**$9.45 \cdot 10^{-4} \pm 1.19 \cdot 10^{-4}$** (+)	$2.27 \cdot 10^{-3} \pm 3.09 \cdot 10^{-4}$ (–)	$1.70 \cdot 10^{-3} \pm 1.29 \cdot 10^{-4}$
	10	$1.19 \cdot 10^{-3} \pm 1.44 \cdot 10^{-4}$ (+)	**$9.06 \cdot 10^{-4} \pm 9.72 \cdot 10^{-5}$** (+)	$2.95 \cdot 10^{-3} \pm 2.89 \cdot 10^{-4}$ (–)	$2.26 \cdot 10^{-3} \pm 2.06 \cdot 10^{-4}$
	20	$1.25 \cdot 10^{-3} \pm 1.61 \cdot 10^{-4}$ (+)	**$1.09 \cdot 10^{-3} \pm 1.24 \cdot 10^{-4}$** (+)	$4.92 \cdot 10^{-3} \pm 4.88 \cdot 10^{-4}$ (–)	$3.20 \cdot 10^{-3} \pm 3.17 \cdot 10^{-4}$

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.

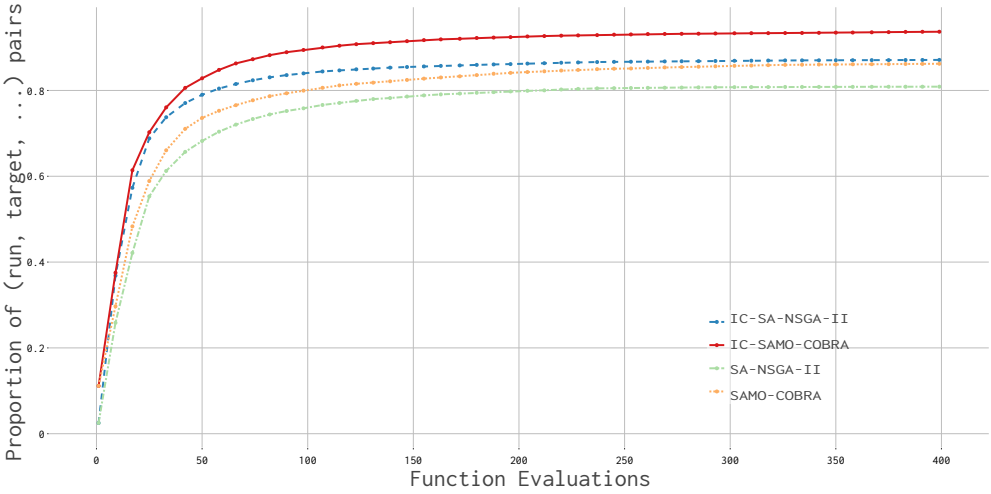


Figure 5.7: Empirical Cumulative Distribution Functions of hypervolume performance metric for SA-NSGA-II, IC-SA-NSGA-II, SAMO-COBRA and IC-SAMO-COBRA. All experiments with different numbers of candidate solutions per iteration and on different test functions are aggregated.

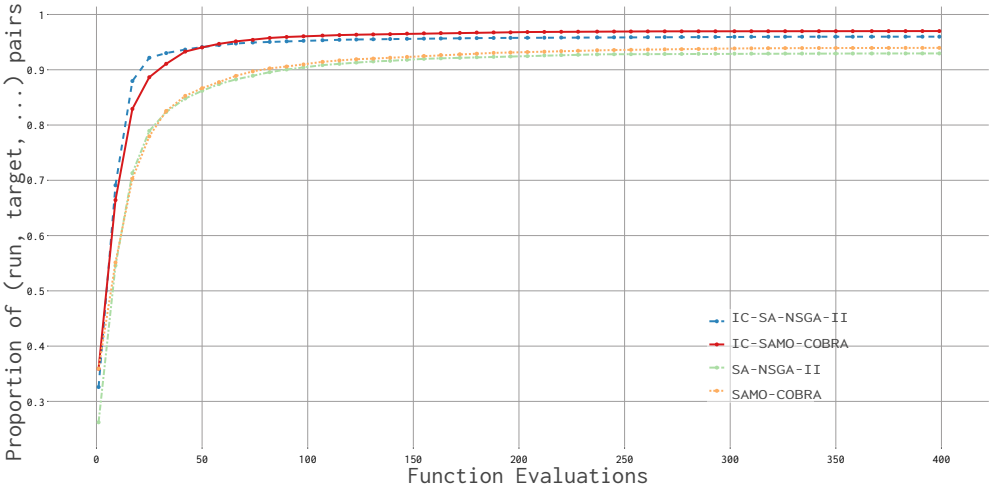


Figure 5.8: Empirical Cumulative Distribution Functions of IGD+ performance metric for SA-NSGA-II, IC-SA-NSGA-II, SAMO-COBRA and IC-SAMO-COBRA. All experiments with different numbers of candidate solutions per iteration and on different test functions are aggregated.

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.

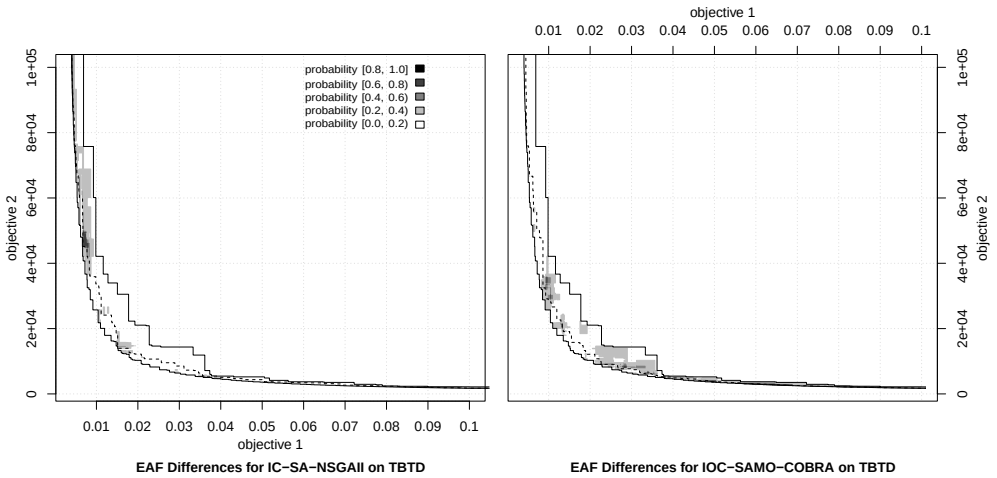


Figure 5.9: Visualization of the Empirical Attainment Function differences between IC-SA-NSGA-II and the IOC-SAMO-COBRA algorithm on the TBTD problem. The solid, dashed and solid lines from left to right represent the best, median and worst found Pareto frontier of both algorithms combined. The grey 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.