Constrained multi-objective optimization with a limited budget of function evaluations

Winter, R. de; Bronkhorst, P.; Stein, B. van; Bäck, T.H.W.

Memetic Computing (2022) 14:151–164
https://doi.org/10.1007/s12293-022-00363-y

REGULAR RESEARCH PAPER

Constrained Multi-Objective Optimization with a Limited Budget of Function Evaluations

Roy de Winter[1,2] · Philip Bronkhorst[2] · Bas van Stein[1] · Thomas Bäck[1]

Received: 15 July 2021 / Accepted: 27 February 2022 / Published online: 8 April 2022

Abstract
This paper proposes the Self-Adaptive algorithm for Multi-Objective Constrained Optimization by using Radial Basis Function Approximations, SAMO-COBRA. This algorithm automatically determines the best Radial Basis Function-fit as surrogates for the objectives as well as the constraints, to find new feasible Pareto-optimal solutions. SAMO-COBRA is compared to a wide set of other state-of-the-art algorithms (IC-SA-NSGA-II, SA-NSGA-II, NSGA-II, NSGA-III, CEGO, SMES-RBF) on 18 constrained multi-objective problems. In the first experiment, SAMO-COBRA outperforms the other algorithms in terms of achieved Hypervolume (HV) after being given a fixed small evaluation budget on the majority of test functions. In the second experiment, SAMO-COBRA outperforms the majority of competitors in terms of required function evaluations to achieve 95% of the maximum achievable Hypervolume. In addition to academic test functions, SAMO-COBRA has been applied on a real-world ship design optimization problem with three objectives, two complex constraints, and five decision variables.

Keywords Constrained optimization · Multi-objective optimization · Optimization under limited budgets · Real-world application.

1 Introduction

Real-world optimization problems often have multiple conflicting objectives, several constraints, and decision variables in the continuous domain [4,16,43]. Without loss of generality, a constrained multi-objective optimization problem can be defined as follows [13]:

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

where k represents the number of objectives, m represents the number of constraints, and d represents the number of parameters in the optimization problem. Solving this problem type can be done by searching for *feasible Pareto-optimal solutions*. The definition of a solution that is feasible and Pareto-optimal can be found in Def. 1.

Definition 1 (**Feasible Pareto-optimal solution**): $\mathbf{x} \in \Omega$ is called *feasible Pareto-optimal* with respect to Ω and $g_i(\mathbf{x}) \leq 0 \;\; \forall i \in \{1, \ldots, m\}$, if and only if there is no solution $\mathbf{x}'$ for which $\mathbf{v} = f(\mathbf{x}') = (f_1(\mathbf{x}'), \ldots, f_k(\mathbf{x}'))^\top$ dominates $\mathbf{u} = f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))^\top$ where $g_i(\mathbf{x}) \leq 0$ and $g_i(\mathbf{x}') \leq 0 \; \forall i \in \{1, \ldots, m\}$.

On top of the already complex constrained multi-objective optimization problem characteristics, there is also a need to reduce the computational and licensing cost involved. There are many engineering examples which require expensive function evaluations [35], special hardware, or software licences [37]. Practical examples of such problems can be found in the automotive industry, aerospace engineering, and the maritime industry.

Handling constraints in optimization problems can be done in several ways: using penalty functions, separation of

✉ Roy de Winter
r.de.winter@liacs.leidenuniv.nl

Bas van Stein
b.van.stein@liacs.leidenuniv.nl

Thomas Bäck
t.h.w.baeck@liacs.leidenuniv.nl

[1] Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, 2333, CA, Leiden, The Netherlands

[2] Research and Development, C-Job Naval Architects, Regulusplein 1, 2132, JN Hoofddorp, The Netherlands

constraints and objectives, treating constraints as additional objectives, or hybrid methods [3,16]. In our approach we only consider separation of constraints and objectives 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.

An algorithm which uses the separation of constraints and objectives in combination with surrogates is SAMO-COBRA [41]. SAMO-COBRA, which is an abbreviation for Self-Adaptive Multi-Objective Constrained Optimization by using Radial Basis Function Approximations, owes its name to the very efficient constraint handling algorithms: COBRA [31] and SACOBRA [3]. Besides constraint handling, SAMO-COBRA shows to be efficient in finding Pareto-optimal solutions, thereby solving constrained multi-objective problems by using a limited number of function evaluations.

Compared to our previous work describing SAMO-COBRA [41], in this paper the SAMO-COBRA algorithm is described in more detail, SAMO-COBRA is compared to two new state-of-the-art algorithms, additional experiments are conducted to evaluate its performance, and SAMO-COBRA has been used in practice to solve a real-world optimization problem. In the real-world application SAMO-COBRA is used to solve a real-world ship design optimization problem with five variables, two complex constraints, and three objectives.

1.1 Outline

The remainder of this paper is organized as follows. In Sect. 2 related work is discussed. In Sect. 3 the SAMO-COBRA algorithm is proposed and described in detail. In Sect. 4 the experimental setup is given on how SAMO-COBRA is compared to other state of the art algorithms. In Sect. 5 the results of the experiments are reported. In Sect. 6 an example is given of a real-world application of SAMO-COBRA. The final concluding remarks are given in Sect. 7.

2 Related work

Existing work on surrogate assisted optimization is typically limited to a subset of three relevant requirements: multi-objective, constrained, and speed. For example, methods exist for quickly solving constrained single-objective problems (e.g. SACOBRA [3]), for multi-objective optimization without efficient constraint handling techniques (e.g. SMS-EGO [29] and PAREGO [25]), or for constrained multi-objective optimization without using meta-models, leading to a large number of required function evaluations (e.g. NSGA-II [14], NSGA-III [24], SPEA2 [44], and SMS-EMOA [6]). Some recently proposed algorithms address all three requirements, however, their computational cost grows cubically in every iteration and exponentially for each additional decision parameter due to the use of Kriging surrogates (e.g. CEGO [42], ECMO [36]).

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 [11], Blank and Deb's SA-NSGA-II [8], and Blank's and Deb's IC-SA-NSGA-II [8]).

SMES-RBF 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. The most promising solution(s) according to a non-dominated sorting procedure are then evaluated on the real objective and constraint function until the evaluation budget limit has been reached.

SA-NSGA-II is the surrogate assisted NSGA-II algorithm that exploits the Cubic Radial Basis Functions with a linear tail as a surrogate to find the Pareto frontier. SA-NSGA-II starts with a Latin Hypercube Sample to train the surrogates. Afterwards, in every iteration the surrogates are updated and used by the NSGA-II algorithm to determine the solutions to be evaluated next.

IC-SA-NSGA-II assumes that the constraints are inexpensive and exploits this assumption by only evaluating the objective functions if the constraints are satisfied. IC-SA-NSGA-II starts by creating a Riesz s-Energy sample (Energy) [22], which results in a well-spaced feasible point set. After the initial sample is evaluated, cubic Radial Basis Functions with a linear tail are fitted as surrogates for the objective and constraint functions. The surrogates are then used by the NSGA-II algorithm to find novel solutions. The constraint functions are evaluated first, the objective functions are only evaluated if the constraints are not violated. 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.

3 Constrained multi-objective optimization algorithm

In this section, the new SAMO-COBRA algorithm is introduced. It is designed for dealing 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 is 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 (HV) which is computed between a reference point and the Pareto-frontier (PF). The pseudocode of SAMO-COBRA can be found in Algorithm 1. The Python implementation can be found on the Github page [39]. In the subsections below, the algorithm is explained in more detail.

3.1 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 [21] with an initial sample that is as small as possible [9]. A few experiments (See Appendix A.1) confirmed that a small initial sample size and Halton sampling, also leads to the best results when applied on six constrained multi-objective problems from Sect. 4.

An RBF model that models the relationship between the input space and the output space can already be trained with $d+1$ evaluated solutions. 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.

3.2 Radial basis function fitting and interpolation

RBF interpolation approximates a function by fitting a linear weighted combination of RBFs [3]. The challenge is to find correct weights ($\boldsymbol{\theta}$) and a good RBF kernel $\varphi(\|\mathbf{x}-\mathbf{c}\|)$. An RBF is only dependent on the distance between the input point $\mathbf{x}$ to the center $\mathbf{c}$. The RBFs used in this work take each evaluated point as the centroid of the function, and the weighted linear combination of RBFs always produces a perfect fit through the training points. Besides the perfect fit on the training points, the linear combination of the RBFs can also give a reasonable approximation of the unknown area.

Any function which is only dependent on the distance from a specific point to another point belongs to the group of RBFs. The RBF kernels (φ) considered in this work are the *cubic* with $\varphi(r) = r^3$, *Gaussian* with $\varphi(r) = \exp(-(\epsilon \cdot r)^2)$, *multiquadric* with $\varphi(r) = \sqrt{1+(\epsilon \cdot r)^2}$, *inverse quadratic* with $\varphi(r) = (1+(\epsilon \cdot r)^2)^{-1}$, *inverse multiquadric* with $\varphi(r) = (\sqrt{1+(\epsilon \cdot r)^2})^{-1}$, and *thin plate spline* with $\varphi(r) = r^2 \log r$. Note that the shape/width parameter ϵ for every individual RBF is kept constant such as proposed by Urquhart et al. [38]. Moreover, all shape parameters are fixed to $\epsilon = 1$.

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

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

Here $\mathbf{f}$ is a vector of length n with the function values belonging to one of the objectives or constraints. Because $\boldsymbol{\Phi}$ is not always invertible, Micchelli introduced RBFs with a polynomial tail, better known as augmented RBFs [26]. In this work augmented RBFs are used with a second order polynomial tail. The polynomial tail is created by extending the original matrix $\boldsymbol{\Phi}$ with $\mathbf{P} = (1, x_{i1}, \ldots, x_{id}, x_{i,1}^2, \ldots, x_{id}^2)$, in its ith row, where x_{ij} is the jth component of vector $\mathbf{x}_i$, for $i = 1, \ldots, n$ and $j = 1, \ldots, d$, $\mathbf{P}^\top$, and zeros $\mathbf{0}_{(2d+1)\times(2d+1)}$, leading to $1+2d$ more weights $\boldsymbol{\mu}$ to learn.

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

Now that the weights $\boldsymbol{\theta}$ can be computed and $\boldsymbol{\mu}$ with Eq. 1 (Lines 16, 17, 20, 21 of Algorithm 1), for each unseen input $\mathbf{x}'$ the function value (f') can be interpolated/predicted by using Eq. (3).

$$\begin{aligned} f' &= \boldsymbol{\Phi}' \cdot [\boldsymbol{\theta}\boldsymbol{\mu}] \\ f' &= \sum_{i=1}^{n} \theta_i \varphi(\|\mathbf{x}' - \mathbf{x}_i\|) + \mu_0 + \sum_{l=1}^{d} \mu_l \mathbf{x}'_l + \sum_{l=1}^{d} \mu_l \mathbf{x}'^2_l, \\ \mathbf{x} &\in \mathbb{R}^d \end{aligned} \tag{3}$$

3.3 Scaling

In SAMO- COBRA, various scaling and transformation functions are used in lines 9-13 of the algorithm. This is done to improve the predictive accuracy of the RBF surrogate models. The four functions SCALE, PLOG, STANDARDIZE and the SCALE CONSTRAINT are described below.

Scale: The input space/decision variables are scaled into the range $[-1, 1]$ with $x = 2 \cdot (x - x_{lb})/(x_{ub} - x_{lb}) - 1$. By scaling large values in the input space, computationally singular (ill-conditioned) coefficient matrices in Eq. (1) can be prevented. In case the large values in the input space are kept, the linear equation solver will terminate with an error, or it will result in a large root mean square error [3]. Additionally, when fitting the RBFs, a small change in one of the variables, is relatively the same small change in all the other variables, making each variable in the basis equally important and equally sensitive.

Standardize: The relationship between the input space and the objective function values is modelled with RBF surrogates. Besides this relationship, Bagheri et al. also exploited

Algorithm 1: SAMO-COBRA. **Input:** Objective functions $f(\mathbf{x})$, constraint function(s) $g(\mathbf{x})$, decision parameters' lower and upper bounds $[\mathbf{lb}, \mathbf{ub}] \subset \mathbb{R}^d$, 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.

1 **Function** SAMO-COBRA(f, g, [**lb**, **ub**], **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 $\quad \mathbf{F} \leftarrow f(\mathbf{X})$ ▷ Obtain objective scores, $\mathbf{F} \in \mathbb{R}^{k \times N}$
4 $\quad \mathbf{G} \leftarrow g(\mathbf{X})$ ▷ Obtain constraint scores, $\mathbf{G} \in \mathbb{R}^{m \times N}$
5 $\quad$ **for** $i \leftarrow 1$ **to** $k + m$ **do**
6 $\quad\quad RBF_i^* \leftarrow (kernel = cubic,$ PLOG=$True)$ ▷ Initialize best RBF configuration
7 $\quad$ **end**
8 $\quad$ **while** $N < N_{max}$ **do**
9 $\quad\quad \hat{\mathbf{X}} \leftarrow$ SCALE($\mathbf{X}, [-1, 1]^d$) ▷ Scale input space to $[-1, 1]^d$
10 $\quad\quad \tilde{\mathbf{F}} \leftarrow$ PLOG($\mathbf{F}$) ▷ See function plog in Eq. (5)
11 $\quad\quad \tilde{\mathbf{G}} \leftarrow$ PLOG($\mathbf{G}$) ▷ See function plog in Eq. (5)
12 $\quad\quad \hat{\mathbf{F}} \leftarrow$ STANDARDIZE($\mathbf{F}$) ▷ Standardize objective space
13 $\quad\quad \hat{\mathbf{G}} \leftarrow$ SCALE CONSTRAINT($\mathbf{G}$) ▷ 0 remains feasibility boundary
14 $\quad\quad$ **for** $\varphi \in RBF_{kernels}$ **do** ▷ For each kernel
15 $\quad\quad\quad$ **for** $i \leftarrow 1$ **to** k **do** ▷ For each objective
16 $\quad\quad\quad\quad \hat{S}_i^{\varphi} \leftarrow$ FITRBF($\hat{\mathbf{X}}, \hat{\mathbf{F}}_{(i,\cdot)}, \varphi$) ▷ Fit RBF with standardized objective values
17 $\quad\quad\quad\quad \tilde{S}_i^{\varphi} \leftarrow$ FITRBF($\hat{\mathbf{X}}, \tilde{\mathbf{F}}_{(i,\cdot)}, \varphi$) ▷ Fit RBF with Plog transformed objective values
18 $\quad\quad\quad$ **end**
19 $\quad\quad\quad$ **for** $j \leftarrow 1$ **to** m **do** ▷ For each constraint
20 $\quad\quad\quad\quad \hat{S}_{k+j}^{\varphi} \leftarrow$ FITRBF($\hat{\mathbf{X}}, \hat{\mathbf{G}}_{(j,\cdot)}, \varphi$) ▷ Fit RBF with scaled constrained scores
21 $\quad\quad\quad\quad \tilde{S}_{k+j}^{\varphi} \leftarrow$ FITRBF($\hat{\mathbf{X}}, \tilde{\mathbf{G}}_{(j,\cdot)}, \varphi$) ▷ Fit RBF with Plog transformed constraint values
22 $\quad\quad\quad$ **end**
23 $\quad\quad$ **end**
24 $\quad\quad S^* \leftarrow \left\{ S_i^{(RBF_i^*)} \mid \forall i = 1, \ldots, (k+m) \right\}$ ▷ Apply best RBF configuration defined on line 32
25 $\quad\quad \mathbf{PF} \leftarrow$ PARETO($\mathbf{X}, \mathbf{F}, \mathbf{G}$) ▷ PF indicator see Def. 1, $\mathbf{PF} \in \{0, 1\}^N$
26 $\quad\quad \mathbf{x}^* \leftarrow$ MAXIMIZE(HV, **PF**, **ref**, S^*) ▷ Get best solution based on HV contribution, see Section 3.4
27 $\quad\quad \mathbf{x}_{new} \leftarrow$ SCALE($\mathbf{x}^*$, [**lb**, **ub**]) ▷ Scale to original scale
28 $\quad\quad N \leftarrow N + 1$ ▷ Increase iteration counter to new matrix sizes
29 $\quad\quad \mathbf{X} \leftarrow [\mathbf{X}\ \mathbf{x}_{new}]$ ▷ Add new solution, $\mathbf{X} \in \mathbb{R}^{d \times N}$
30 $\quad\quad \mathbf{F} \leftarrow [\mathbf{F}\ f(\mathbf{x}_{new})]$ ▷ Add evaluated objectives, $\mathbf{F} \in \mathbb{R}^{k \times N}$
31 $\quad\quad \mathbf{G} \leftarrow [\mathbf{G}\ g(\mathbf{x}_{new})]$ ▷ Add evaluated constraints, $\mathbf{G} \in \mathbb{R}^{m \times N}$
32 $\quad\quad RBF^*, \mathbf{SE} \leftarrow$ SELECTBESTRBF(**SE**, S, $\mathbf{x}^*$, **F**, **G**, **PF**, N) ▷ Get best RBF configuration, see Section 3.6
33 $\quad$ **end**
34 **return** $(\mathbf{F}_{(\cdot,\mathbf{PF})}, \mathbf{G}_{(\cdot,\mathbf{PF})}, \mathbf{X}_{(\cdot,\mathbf{PF})})$

similarities between RBF and Kriging surrogates to come up with an uncertainty quantification method [2]. The formula for this uncertainty quantification method is given in Eq. (4).

$$\widehat{U}_{RBF} = \varphi(\|\mathbf{x}' - \mathbf{x}'\|) - \boldsymbol{\Phi}'^{\top} \boldsymbol{\Phi}^{-1} \boldsymbol{\Phi}' \tag{4}$$

The uncertainty ($\widehat{U}_{RBF}$) of solutions far away from earlier evaluated solutions is higher compared to solutions close to earlier evaluated solutions. This uncertainty quantification method can therefore help in exploration, and prevent the algorithm from getting stuck in a local optimal solution. However, as can be derived from Eq. (4), the uncertainty quantification method is only dependent on the input space and not on the scale of the objective and/or weights of the RBF models. The objective values are therefore standardized as $y' = (y - \bar{y})/\sigma$ so that the uncertainty scale and the objective scale match. Here σ is the standard deviation of y, and $\bar{y}$ the mean of y.

Scale Constraint: The constraint evaluation function should return a continuous value, namely the amount by which the constraint is violated. Since it is possible to have multiple constraints, and each constraint is equally important, every constraint out-

put is scaled with $c' = c/(\max(c) - \min(c))$, where $\max(c)$ is the maximum constraint violation encountered so far, and $\min(c)$ is the smallest constraint value seen so far. After scaling, the difference between $\min(c)$ and $\max(c)$ becomes 1 for all constraints, making every constraint equally important while 0 remains the feasibility boundary.

Plog: In cases where there are very steep slopes, a logarithmic transformation of the objective and/or constraint scores can be beneficial for the predictive accuracy [32]. Therefore, the scores are transformed with the PLOG transformation function. The extension to a matrix argument $\mathbf{Y}$ is defined component-wise, i.e., each matrix element y_{ij} is subject to PLOG.

$$\text{PLOG}(y) = \begin{cases} +\ln(1+y), & \text{if } y \geq 0 \\ -\ln(1-y), & \text{if } y < 0 \end{cases} \quad (5)$$

3.4 Maximize hypervolume contribution

After modelling the relationship between the input space and the response variables with the RBFs, the RBFs are used as cheap surrogates. For each constraint and objective, the best RBF configuration is chosen as described in Sect. 3.6. By using Eq. (3) 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 highly 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 criteria (SMS), and the Predicted Hyper-Volume criteria (PHV). Computation of the two infill criteria is done as follows:

1. Compute all objective scores for a given solution $\mathbf{x}'$ with Eq. (3). With the interpolated objective scores, compute the additional Predicted Hyper-Volume (PHV) score this solution adds to the PF. This is a purely exploitative infill criteria without any uncertainty quantification method.
2. Compute all objective scores for a given solution $\mathbf{x}'$ with Eq. (3) and subtract the uncertainty of each objective given $\mathbf{x}'$ and Eq. (4). With the interpolated objective score minus the uncertainty, the potential HV that this solution could add to the PF is calculated. This infill criteria is similar to the Kriging $\mathcal{S}$-metric Selection (SMS) criterion from Emmerich et al. [6]. Because of the subtracted uncertainty, it will be more exploratory compared to the PHV criterion.

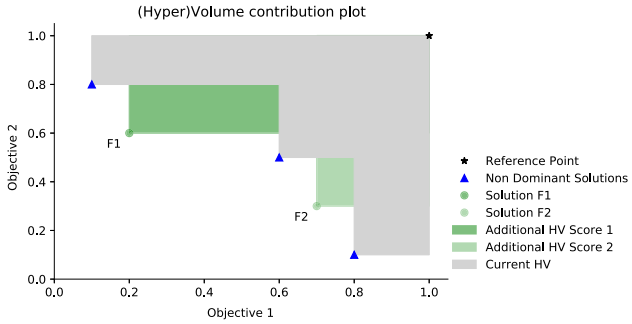

Fig. 1 Visual representation of hypervolume contribution of two solutions F1 and F2. The additional hypervolume of F1 is larger compared to the additional hypervolume F2. For this reason, F1 is the more preferred solution

How much a solution adds to the PF 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 1. By using any of the two infill criteria, the constrained multi-objective problem has been translated into a constrained single-objective problem.

After an infill criteria is chosen by the user, the constrained single-objective problem can be formulated and optimized. The COBYLA (Constrained Optimization BY Linear Approximations) algorithm [30] is used to maximize the infill criteria (Line 26 of Algorithm 1). COBYLA is allowed to vary $\mathbf{x}'$ between the lower and the upper bound of the design space $\mathbf{x}' \in [\mathbf{lb},\mathbf{ub}]$. COBYLA deals with constraints by preferring feasible solutions above infeasible solutions. This way, COBYLA searches for a Pareto-optimal solution which does not violate any of the constraints and has the highest possible infill criteria score.

If no feasible solution can be found by COBYLA, the solution with the smallest constraint violation is searched for. 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 surrogate for the real objective and constraint functions. Only after the next best solution on the surrogates is found, it is evaluated on the real objective and constraint functions (Lines 27-31 of Algorithm 1).

3.5 Surrogate exploration and RBF adaptation

To increase the chance of finding the best feasible pareto-optimal solution in every iteration, two hyper parameter updates are done. In every iteration of SAMO-COBRA the surrogate search budget and the allowed constraint RBF approximation error margin are updated.

The chances of finding the best feasible Pareto-optimal solution can be increased by starting the surrogate search not from one solution but from multiple randomly generated solutions independently. Each independent local search done by COBYLA gets an allocated search budget, which is updated every iteration together with the number of starting points. The problem characteristics (number of variables d, constraints m, and objectives k) influence the optimization problem complexity. Therefore, the surrogate evaluation budget and number of starting points are empirically chosen and set at $50 \cdot (d + m + k)$ and $2 \cdot (d + m + k)$ respectively. In every iteration of SAMO-COBRA the convergence of COBYLA is checked. If COBYLA converges every time to a feasible solution, the number of randomly generated points is increased by 10% and the surrogate search budget is decreased by 10%. The opposite update is done if COBYLA did not converge from one of the starting points. The 10% search budget update step size is empirically chosen as a well working step size (See Appendix A.2 for the experiment). The search budget is updated this way to limit the time spent on exploring the surrogates and to further increase the chances of finding a solution that adds the most HV to the PF.

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%. The 10% ϵ update step size is empirically chosen as a well working step size (See Appendix A.2 for the experiment).

3.6 Selection of the best RBF

In every iteration, the best RBF kernel and transformation strategy is chosen (Line 32 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 [1]. 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.

To give insight into this RBF kernel and transformation switching approach, an additional experiment is conducted. In Appendix A.3, an experiment is described and results are given on how frequent the RBF kernel is changed and how often the PLOG transformation strategy is changed.

4 Experimental setup

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 (PHV), and one with the uncertainty quantification method (SMS). The performance of the two variants are compared to the performance of the following algorithms: CEGO [42], IC-SA-NSGA-II [8], SA-NSGA-II [8], NSGA-II [14], NSGA-III [24], and SMES-RBF [11]. The performance of the algorithms except for SMES-RBF are assessed on 18 academic benchmark functions. SMES-RBF is not 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.

The 18 test functions and their characteristics are listed in Table 1. Some of the problems are real-world-like-problems while others are artificially created benchmark problems, proposed by several authors over the past years [37]. 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.

4.1 Hyper parameter 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 hyper parameters from the original implementations are used. Since there are no clear recommendations for the

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 Efficient Solutions in vector
  $T \leftarrow \{T_i \leftarrow \infty \mid \forall i = 1, \ldots, (k+m)\}$ ▷ Temporary vector for smallest sum of approximation errors
  for $\varphi \in RBF_{kernels}$ **do** ▷ For each kernel
    for $i \leftarrow 1$ **to** k **do** ▷ For each objective
      $\hat{\mathbf{SE}}^{\varphi}_{i,N} \leftarrow \left(\text{INTERPOLATE}(\hat{S}^{\varphi}_i, \mathbf{x}^*) - \mathbf{F}_{i,N}\right)^2$ ▷ Error^2 of RBF trained with standardized values
      $\tilde{\mathbf{SE}}^{\varphi}_{i,N} \leftarrow \left(\text{INTERPOLATE}(\tilde{S}^{\varphi}_i, \mathbf{x}^*) - \mathbf{F}_{i,N}\right)^2$ ▷ Error^2 of RBF trained with Plog transformed values
    end
    for $j \leftarrow 1$ **to** m **do** ▷ For each constraint
      $\hat{\mathbf{SE}}^{\varphi}_{k+j,N} \leftarrow \left(\text{INTERPOLATE}(\hat{S}^{\varphi}_{k+j}, \mathbf{x}^*) - \mathbf{G}_{j,N}\right)^2$ ▷ Error^2 of RBF trained with standardized values
      $\tilde{\mathbf{SE}}^{\varphi}_{k+j,N} \leftarrow \left(\text{INTERPOLATE}(\tilde{S}^{\varphi}_{k+j}, \mathbf{x}^*) - \mathbf{G}_{j,N}\right)^2$ ▷ Error^2 of RBF trained with Plog transformed values
    end
    for $i \leftarrow 1$ **to** $k+m$ **do** ▷ For each constraint and objective find best RBF kernel and Plog strategy
      if $(\sum_{n=1}^{N} \mathbf{ID}_n \cdot \hat{\mathbf{SE}}^{\varphi}_{i,n}) < T_i$ **then** ▷ If cumulative sum of marked solutions is smaller then temp
        $T_i \leftarrow \sum_{n=1}^{N} \mathbf{ID}_n \cdot \hat{\mathbf{SE}}^{\varphi}_{i,n}$ ▷ Store sum of smallest approximation errors in temp
        $RBF^*_i \leftarrow (kernel = \varphi, \text{PLOG}{=}False)$ ▷ Store best RBF kernel and Plog strategy
      if $(\sum_{n=1}^{N} \mathbf{ID}_n \cdot \tilde{\mathbf{SE}}^{\varphi}_{i,n}) < T_i$ **then** ▷ If cumulative sum of marked solutions is smaller then temp
        $T_i \leftarrow \mathbf{ID}_n \cdot \tilde{\mathbf{SE}}^{\varphi}_{i,n}$ ▷ Store sum of smallest approximation errors in temp
        $RBF^*_i \leftarrow (kernel = \varphi, \text{PLOG}{=}True)$ ▷ Store best RBF kernel and Plog strategy
    end
  end
return $(RBF_*, \mathbf{SE})$

hyper parameters 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.[1] The original implementation of IC-SA-NSGA-II and SA-NSGA-II can be found on the personal page of Julian Blank.[2] For NSGA-II and NSGA-III the implementation of Platypus is used.[3] The implementation of the SMES-RBF algorithm is not provided. Therefore, only the reported results from the SMES-RBF paper [11] can be compared.

More details concerning the implementation of SAMO-COBRA, the experiments, and the statistical comparison can be found on Github [39].

[1] https://github.com/RoydeZomer/CEGO.

[2] https://julianblank.com/static/misc/pycheapconstr.zip.

[3] https://platypus.readthedocs.io/.

4.2 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 [7]. Each algorithm is allowed to do $40 \cdot d$ function evaluations, here d represents the number of decision variables of the optimization test function.

The HV metric is selected as the performance metric to quantify the results. This because HV simultaneously measures accuracy and diversity, and because it is the most common performance metric [34]. The HV is computed between the obtained Feasible Pareto-optimal solutions and the reference point reported in Table 1. Higher HV scores mean that more HV is covered and therefore a better approximation of the Pareto-frontier is found.

4.3 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 1 and the Pareto-frontier. Since the Pareto-front is not known for every function, NSGA-II is used to find the maximal HV between

Table 1 Test functions with citation, the reference points, the number of objectives k, number of parameters d, number of constraints m and feasibility percentage $P(\%)$ based on 1 million random samples

Function	Reference point	k	d	m	$P(\%)$
BNH [10]	(140, 50)	2	2	2	96.92
CEXP [12]	(1, 9)	2	2	2	57.14
SRN [15]	(301, 72)	2	2	2	16.18
TNK [15]	(2, 2)	2	2	2	5.05
CTP1 [12]	(1, 2)	2	2	2	92.67
C3DTLZ4 [37]	(3, 3)	2	6	2	22.22
OSY [10,15]	(0,386)	2	6	6	2.78
TBTD [18]	(0.1, 50000)	2	3	2	19.46
NBP [17]	(11150, 12500)	2	2	5	41.34
DBD [18]	(5,50)	2	4	5	28.55
SPD [28]	(16, 19000, -260000)	3	6	9	3.27
CSI [24]	(42, 4.5, 13)	3	7	10	18.17
SRD [27]	(7000, 1700)	2	7	11	96.92
WB [18]	(350, 0.1)	2	4	5	35.28
BICOP1 [11]	(9, 9)	2	10	1	100
BICOP2 [11]	(70, 70)	2	10	2	10.55
TRIPCOP [11]	(34, -4, 90)	3	2	3	15.85
WP [24]	(83000, 1350, 2.85, 15989825, 25000)	5	3	7	92.06

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 evaluation metric.

4.3.1 SMES-RBF convergence 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 [11].

5 Results

The complete set of results from the experiments can be found on Github [39]. It is important to keep in mind when analysing the results that IC-SA-NSGA-II as opposed to the other algorithms, uses many more constraint function evaluations.

5.1 Fixed budget experiment results

The results of the first experiment, in which the HV is computed after $40 \cdot d$ function evaluations, can be found in Table 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 compared to the algorithm with the lesser results.

Interestingly, the SAMO-COBRA with the predicted HV infill criterion (PHV) outperforms the state-of-the art algorithms in 16 out of the 18 test functions with the exception of the SAMO-COBRA SMS variant. Only on the *WB* and *C3DTLZ4* test function the IC-SA-NSGA-II algorithm finds a significantly larger HV score. However, for the IC-SA-NSGA-II algorithm only the objective function evaluations are counted and not the constraint function calls. It is therefore not remarkable that for some test functions IC-SA-NSGA-II finds a higher HV.

Inspection showed that IC-SA-NSGA-II uses 10 000 constraint function evaluations to find a well spread feasible initial sample, after which IC-SA-NSGA-II starts optimizing. In order to obtain the Pareto frontier, the IC-SA-NSGA-II used on average 10 537 and 10776 constraint function evaluations for respectively the *WB* and *C3DTLZ4* test function.

5.2 Convergence experiment results

In Table 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. Note again that for IC-SA-NSGA-II only the objective function calls are reported and not the constrained function evaluations.

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.

As can be seen in Table 3, SAMO-COBRA with the Predicted HV (PHV) 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.

5.2.1 SMES-RBF convergence experiment results

As mentioned before, the implementation of SMES-RBF is not provided. Therefore, the reported results of SMES-RBF are compared with the results of SAMO-COBRA. In Table 4 the number of function evaluations are reported that are required to obtain the same HV as reported in the SMES-RBF paper.

Table 2 Mean HV after $40 \cdot d$ function evaluations for each algorithm on each test function.

Function	PHV	SMS	CEGO	IC-SA-NSGA-II	SA-NSGA-II	NSGA-II	NSGA-III
BNH	**5256.0**	5251.0	5218.0	5154.6	5158.3	5089.7	4848.8
CEXP	**3.7968**	3.7967	3.7658	3.7072	3.6749	3.1544	2.9561
SRN	**62385**	62377	62307	61074	60767	54233	52585
TNK	8.0430	**8.0474**	8.0309	8.0033	7.8031	6.4282	6.2948
CTP1	**1.3026**	1.3023	1.2972	1.2941	1.2924	1.1929	1.1864
C3DTLZ4	6.3016	6.4697	6.2664	**6.6326**	5.9269	5.2489	5.2500
OSY	**100577**	100458	100181	96978	96889	46631	42204
TBTD	4029.3	4027.5	**4055.0**	4026.3	3841.9	3421.5	3446.0
NBP	$\mathbf{1.0785 \cdot 10^8}$	$1.0785 \cdot 10^8$	$1.0784 \cdot 10^8$	$1.0740 \cdot 10^8$	$1.0552 \cdot 10^8$	$9.8573 \cdot 10^7$	$9.5816 \cdot 10^7$
DBD	**228.77**	228.34	227.85	227.25	227.80	218.47	218.33
SPD	$\mathbf{3.8859 \cdot 10^{10}}$	$3.7602 \cdot 10^{10}$	$3.4057 \cdot 10^{10}$	$3.6956 \cdot 10^{10}$	$3.6939 \cdot 10^{10}$	$2.6069 \cdot 10^{10}$	$2.5689 \cdot 10^{10}$
CSI	**27.800**	25.167	*terminated*	25.605	25.532	17.296	16.993
SRD	**4205165**	4164992	4148333	4200148	4198861	2766367	2673599
WB	34.395	34.392	34.522	**34.618**	34.457	34.0270	33.995
BICOP1	**80.664**	78.160	*terminated*	80.618	80.635	67.6506	70.911
BICOP2	**4834.33**	4822.13	*terminated*	4826.74	4816.30	4816.12	4816.72
TRIPCOP	20610	**20611**	20609	20568	20544	20101	19982
WP	$\mathbf{1.5991 \cdot 10^{19}}$	$1.5698 \cdot 10^{19}$	$1.5350 \cdot 10^{19}$	$1.4173 \cdot 10^{19}$	$1.3903 \cdot 10^{19}$	$1.1623 \cdot 10^{19}$	$1.1797 \cdot 10^{19}$

Note that for IC-SA-NSGA-II only the objective function evaluations are counted. PHV and SMS represent the SAMO-COBRA variants. The highest mean HV per test function are presented in **bold**. The Wilcoxon rank-sum test (with Bonferroni correction) significance is represented with a grayscale. Background colours represent: $p \leq 0.001$, $p \leq 0.01$, $p \leq 0.05$. Red shows that the algorithm took more than 24 hours

Table 3 Test Function, Threshold Hypervolume, and number of function evaluations required to achieve this threshold per optimization algorithm.

Function	Threshold	PHV	SMS	CEGO	IC-SA-NSGA-II	SA-NSGA-II	NSGA-II	NSGA-III
BNH	5005.5	**11** ↑	16	12	31	36	56	114
CEXP	3.6181	**13** ↑	16	23	61	71	392	404
SRN	59441	**15** ↑	**15** ↑	17	36	66	200	227
TNK	7.6568	11	**9**↑	**9** ↑	21	66	432	586
CTP1	1.2398	**10** ↑	14	14	26	36	140	170
C3DTLZ4	6.4430	179	181	226	**100** ↑	275	+5000	+5000
OSY	95592	**15** ↑	31	16	110	105	+5000	+5000
TBTD	3925	**31** ↑	58	49	47	357	324	369
NBP	1.024E8	**5** ↑	9	6	21	36	102	206
DBD	217.31	**13** ↑	19	16	43	48	112	142
SPD	3.6887E10	**43** ↑	125	-	185	205	+5000	+5000
CSI	25.717	**59** ↑	484	-	276	376	+5000	+5000
SRD	3997308	**17** ↑	55	28	81	81	952	1357
WB	32.9034	**7** ↑	10	10	43	43	24	24
BICOP1	76.6328	**22** ↑	25	35	114	119	1700	1975
BICOP2	4606.57	17	18	18	109	109	**10** ↑	12
TRIPCOP	19578.0	**7** ↑	8	**7** ↑	21	21	42	8
WP	1.5147E19	**48** ↑	66	111	232	292	3120	3876

The results of the algorithm with the smallest number of function evaluations are reported in bold and accompanied with a ↑. PHV and SMS represents the SAMO-COBRA variants. Experiments that required more than 5000 function evaluations or more than 24 hours are terminated

Table 4 Number of function evaluations after which SAMO-COBRA with PHV infill criterion achieved same hypervolume for 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 4, the number of function evaluations of SAMO-COBRA is much smaller. Interestingly, the HV results for the BICOP1 test function after 1000, 2000, and 5000 function evaluations of SMES-RBF reported in the original paper are, given the Nadir point, not possible.

To further inspect the performance of the algorithms over time, convergence plots are made for the BNH and TRICOP test function. The convergence plots show the HV computed in each iteration. In this experiment the same estimation of Nadir points as in the original SMES-RBF paper [11] are used as the reference points. The convergence of the HV on the BNH test function can be found in Figure 2. The convergence of the HV on the TRICOP test function can be found in Figure 3.

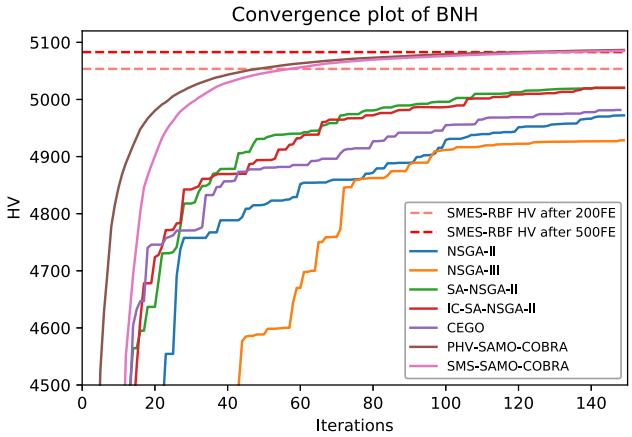

Fig. 2 Convergence plot of BNH test problem for NSGA-II, NSGA-III, SA-NSGA-II, IC-SA-NSGA-II, CEGO, PHV-SAMO-COBRA, SMS-SAMO-COBRA. The dashed lines represents final obtained Hypervolume of SMES-RBF after 200 and 500 function evaluations

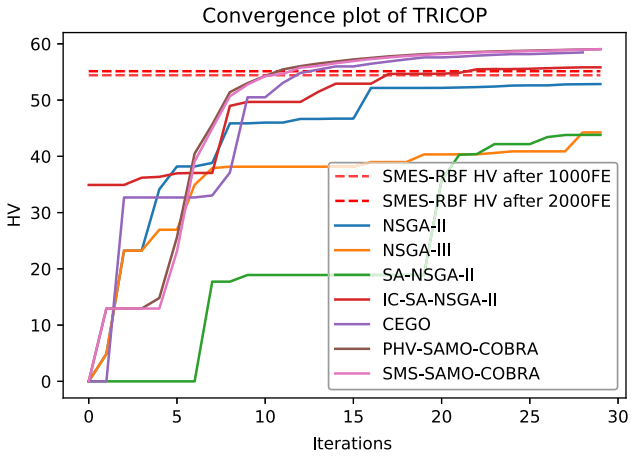

Fig. 3 Convergence plot of TRICOP test problem for NSGA-II, NSGA-III, SA-NSGA-II, IC-SA-NSGA-II, CEGO, PHV-SAMO-COBRA, SMS-SAMO-COBRA. The dashed lines represents final obtained Hypervolume of SMES-RBF after 1000 and 2000 function evaluations

5.3 PHV vs SMS infill criterion

An interesting conclusion from all the experiments is that the exploiting strategy of the PHV 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 constrained multi-objective setting, since a similar effect was already shown by Rehbach et al. [33]. 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 2 and Table 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 get stuck in a local optimum, thereby making the uncertainty quantification method redundant.

6 Real-world application

SAMO-COBRA has been used in practice to design a wind feeder vessel to support the installation of windmills at sea.

Fig. 4 Impression of the Wind Feeder Vessel design by C-Job Naval Architects

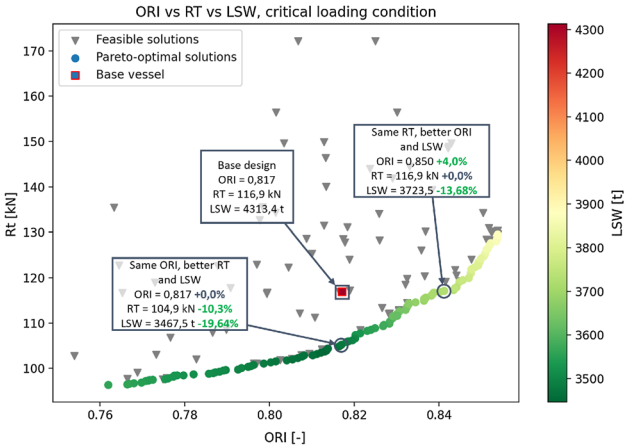

Fig. 5 Pareto Frontier of Ship Design case with Original Design by human expert represented by a square. Objectives are maximize the Operability Robustness Index (ORI[-]), minimize ship resistance (Rt [kN]), and minimize Lightship Weight (LSW[t])

Although high winds are good for power production, they usually also result in rough seas. These rough seas around the wind park installation sites increase the demand for reliable vessels. An example of such a reliable vessel (See Figure 4) has been designed and later optimized with SAMO-COBRA at C-Job Naval Architects.[4] This vessel is designed to support the construction of wind farms and to transport the materials from the shore to the installation sites.

The objectives of the optimization case of the wind feeder vessel are to have a robust seakeeping performance to maximize the year-round operability, while also keeping the operational cost and capital expenses at a minimum. The operability can be optimized by maximizing the so-called Operability Robustness Index (ORI) [20]. The seakeeping assessment is done with a strip theory code of NAPA.[5] Strip theory is proven to be fast and reliable with a sufficient accuracy for conventional hull forms [5,19]. The capital expenses can be translated into the cost of steel that is required to build the vessel, this is roughly equal to the Lightship Weight (LSW) of the vessel. The LSW is calculated by summing the weight of all the equipment plus the minimum amount of steel that is required to fulfil the longitudinal strength requirements. The operational expenses can be dealt with by minimizing the ship resistance in the water at service speed (Rt[kN]). This resistance is calculated with the Holtrop Mennen method [23].

All objectives, practical constraints, relevant rules, regulations, and loading conditions can be evaluated with the modular Accelerated Concept Design framework [40]. In this software, a parametric 3D model of the ship is set up by a naval architect after which automated software tools can evaluate any design variation in one function call. In the wind feeder vessel case, five design parameters are defined: *Aftship Length*, *Midship Length*, *Foreship length*, *Beam at Waterline*, and *Draught*. Since sea-keeping and longitudinal strength are already captured in the objectives, only two constraints are needed. The two constraints are for space reservation of the wind turbine blades and the meta-centric height for intact stability of the vessel. Based on 50 Halton samples, 36% of the design space is feasible.

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

After analysing the results from the optimization study, the base design by the naval architect was shown to be much too large, causing the ship to be too heavy with a sub-optimal performance. If we zoom in on a few Pareto-optimal solutions and compare them with the original, then the solution with the same ORI score has a 10.3% smaller resistance value, and 19.64% less light ship weight. The solution with the same resistance score has a 4% better ORI score, and 13.68% less light ship weight. All the evaluated feasible solutions are visualized on the Pareto frontier in Figure 5.

7 Conclusion and future work

In this paper, two variants of the SAMO-COBRA algorithm are introduced, based on using two different infill criteria: S-Metric-Selection and Predicted Hypervolume (PHV), of which the latter is more exploitative than the former. The performance of the two SAMO-COBRA variants is compared to six other state-of-the-art algorithms: IC-SA-NSGA-II, SA-

[4] Dedicated Naval Architects, https://c-job.com/.

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

NSGA-II, NSGA-II, NSGA-III and SMES-RBF. On 16 out of the 18 test functions, SAMO-COBRA with the PHV infill criterion showed similar or better results. On two test functions, IC-SA-NSGA-II obtained significantly better results which can be explained by the fact that IC-SA-NSGA-II uses at least 10 000 more constraint function evaluations.

The SAMO-COBRA algorithm with the PHV infill criterion showed to be very efficient at solving constrained 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.

Besides such a performance comparison on test functions, SAMO-COBRA has also already been used in practice on a ship design optimization problem with three objectives, two constraints, and five decision variables. In this application, the algorithm demonstrates its ability to outperform the human expert in all objectives simultaneously.

Further research efforts will be put into creating infill criteria which 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. Besides parallelization, effort will be put into dealing with mixed integer decision parameters and solving multi-fidelity optimization problems.

A Appendix

The SAMO-COBRA algorithm has several hyper-parameters which can be tuned. For the most important one, the infill criteria (SMS or PHV) the results from our experiments are shown in the Experiment Sect. 4. Other hyper-parameters like the initial sample strategy, the RBF kernel, and ϵ adaptation step size have a minor influence on the results as presented in the experiments below.

Table 5 Hyper-volume after $40 \cdot d$ function evaluations for SAMO-COBRA with different initial sampling sizes and strategies. **Bold** indicates significant better or indifferent results according to a Wilcoxon rank-sum test with $p \leq 0.05$

Function	Halton $d+1$	Halton $3 \cdot d$	LHS $d+1$
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

A.1 Sample strategy experiment

The initial sample strategy chosen is based on the comparison of the Hyper-volume results of six test functions after $40 \cdot d$ function evaluations in 10 independent runs. The strategy resulting in the highest Hyper-volume in most cases is recommended as the best sampling strategy for the SAMO-COBRA algorithm. The following subset of test functions are selected for the experiments: BNH, CEXP, SRN, TNK, CTP1, and TRICOP.

As can be seen from the results in Table 5, the Halton sampling strategy with $d+1$ initial samples in most cases lead to better or similar results compared to the other two initial sampling strategies.

A.2 RBF and ϵ adaptation experiment

The 10% update step size for the RBF evaluation budget, the number of starting points, and the ϵ constraint margin are empirically chosen based on the below experiments. 5%, 10%, 20%, and 50% are the update step sizes experimented with. The update step size which leads to the highest Hyper-volume in most cases is selected and recommended for the SAMO-COBRA algorithm. The following subset of test functions are selected for the experiments: BNH, CEXP, SRN, TNK, CTP1, and TRICOP. Each function is optimized 10 times with independent runs.

As can be seen from the results in Table 6, the 10% update step size lead to better or statistically indifferent results compared to the other tried step sizes.

A.3 RBF kernel and scaling selection experiment

In each iteration of the SAMO-COBRA algorithm, the best RBF kernel (Cubic, Gaussian, Multiquadric, Inverse Multiquadric, Thin Plate Spline) and best transformation function are chosen (SCALE, PLOG, and STANDARDIZE) for each objective and constraint independently. In Table 7 the amount of strategy switches per test function and the most frequent cho-

Table 6 Hyper-volume after $40 \cdot d$ function evaluations for SAMO-COBRA with different update step sizes. **Bold** indicates significant better or indifferent results according to a Wilcoxon rank-sum test with $p \leq 0.05$

Function	5%	10%	20%	50%
BNH	**5256.4**	**5256.4**	**5256.4**	**5256.4**
CEXP	**3.7971**	**3.7973**	3.7969	3.7952
SRN	**62389**	**62392**	**62384**	62374
TNK	8.0441	**8.0505**	8.0473	8.0485
CTP1	**1.3032**	1.3030	1.3029	1.3028
TRICOP1	**20611**	**20611**	**20611**	**20611**

Table 7 Column **A** shows the number of iterations the kernel and/or transformation is switched for any objective or constraint, column **K** shows the number of iterations the RBF kernel is switched for any objective or constraint, column **T** shows the number of iterations for which the transformation strategy is switched for any constraint or constraint, column **O1** shows the number of iterations the kernel and/or transformation is switched for the first objective, column **C1** shows the number of iterations the kernel and/or transformation is switched for the first constraint, column **Kernel** shows the most frequent used kernel for this test function. The maximum possible number of switches could have been 77 for all the test functions

Function	A	K	T	O1	C1	Kernel
BNH	27	27	3	15	10	Cubic
CEXP	15	15	2	11	4	Multiquadric
SRN	24	24	4	21	3	Multiquadric
TNK	22	22	4	9	3	ThinPlateSpline
CTP1	16	16	4	7	9	ThinPlateSpline
TRICOP1	28	28	2	23	5	ThinPlateSpline

sen RBF kernel per test function are displayed. The results are based on one run with 3 initial samples and 77 iterations of the SAMO-COBRA algorithm with the PHV infill criteria on the BNH, CEXP, SRN, TNK, CTP1, and TRICOP test functions.

References

1. Bagheri S, Konen W, Bäck T (2016) Online selection of surrogate models for constrained black-box optimization. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8. IEEE
2. Bagheri S, Konen W, Bäck T (2017) Comparing kriging and radial basis function surrogates. In: Proc. 27. Workshop Computational Intelligence, pp. 243–259
3. Bagheri S, Konen W, Emmerich M, Bäck T (2017) Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. Appl Soft Comput 61:377–393. https://doi.org/10.1016/j.asoc.2017.07.060
4. Bandaru S, Ng AH, Deb K (2017) Data mining methods for knowledge discovery in multi-objective optimization: Part a-survey. Expert Syst Appl 70:139–159
5. Beck RF, Reed AM, Sclavounos PD, Hutchison BL (2001) Modern computational methods for ships in a seaway. discussion. author's closure. Trans-Soc Naval Archit Marine Eng 109:1–51
6. Beume N, Naujoks B, Emmerich M (2007) SMS-EMOA: Multiobjective selection based on dominated hypervolume. European J Oper Res 181(3):1653–1669. https://doi.org/10.1016/j.ejor.2006.08.008
7. Beume N, Naujoks B, Emmerich M (2007) Sms-emoa: Multiobjective selection based on dominated hypervolume. European J Oper Res 181(3):1653–1669
8. Blank J, Deb K (2021) Constrained bi-objective surrogate-assisted optimization of problems with heterogeneous evaluation times: Expensive objectives and inexpensive constraints. In: Ishibuchi H, Zhang Q, Cheng R, Li K, Li H, Wang H, Zhou A (eds) Evolutionary multi-criterion optimization. Springer International Publishing, Cham, pp 257–269
9. Bossek J, Doerr C, Kerschke P (2020) Initial design strategies and their effects on sequential model-based optimization. arXiv preprint arXiv:2003.13826
10. Coello CAC, Lamont GB, Van Veldhuizen DA et al (2007) Evolutionary algorithms for solving multi-objective problems, vol 5. Springer, New York
11. Datta R, Regis RG (2016) A surrogate-assisted evolution strategy for constrained multi-objective optimization. Expert Syst Appl 57:270–284. https://doi.org/10.1016/j.eswa.2016.03.044
12. Deb K (2001) Multi-objective optimization using evolutionary algorithms, vol 16. John Wiley & Sons, New Jersey
13. Deb K (2011) Multi-objective optimisation using evolutionary algorithms: an introduction. In: Multi-objective evolutionary optimisation for product design and manufacturing, pp. 3–34. Springer
14. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evolut Comput 6(2):182–197. https://doi.org/10.1109/4235.996017
15. Deb K, Pratap A, Meyarivan T (2001) Constrained test problems for multi-objective evolutionary optimization. In: International conference on evolutionary multi-criterion optimization, pp. 284–298. Springer. https://doi.org/10.1007/3-540-44719-9_20
16. Fan Z, Fang Y, Li W, Lu J, Cai X, Wei C (2017) A comparative study of constrained multi-objective evolutionary algorithms on constrained multi-objective optimization problems. In: 2017 IEEE congress on evolutionary computation (CEC), pp. 209–216. IEEE
17. Forrester A, Sobester A, Keane A (2008) Engineering design via surrogate modelling: a practical guide. John Wiley & Sons, New Jersey. https://doi.org/10.2514/4.479557
18. Gong W, Cai Z, Zhu L (2009) An efficient multiobjective differential evolution algorithm for engineering design. Struct Multidiscip Optim 38(2):137–157. https://doi.org/10.1007/s00158-008-0269-9
19. Gourlay T, von Graefe A, Shigunov V, Lataire E (2015) Comparison of aqwa, gl rankine, moses, octopus, pdstrip and wamit with model test results for cargo ship wave-induced motions in shallow water. In: International Conference on Offshore Mechanics and Arctic Engineering, vol. 56598, p. V011T12A006. American Society of Mechanical Engineers
20. Gutsch M, Steen S, Sprenger F (2020) Operability robustness index as seakeeping performance criterion for offshore vessels. Ocean Eng 217:107931
21. Halton JH (1960) On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik 2(1):84–90
22. Hardin D, Saff E (2005) Minimal riesz energy point configurations for rectifiable d-dimensional manifolds. Adv Math 193(1):174–204
23. Holtrop J, Mennen G (1982) An approximate power prediction method. Int Shipbuild Prog 29(335):166–170
24. Jain H, Deb K (2014) An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting

approach, part II: Handling constraints and extending to an adaptive approach. IEEE Trans Evolut Comput 18(4):602–622. https://doi.org/10.1109/tevc.2013.2281534
25. Knowles J (2006) ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans Evolut Comput 10(1):50–66. https://doi.org/10.1109/tevc.2005.851274
26. Micchelli CA (1986) Interpolation of scattered data: distance matrices and conditionally positive definite functions. Construct Approx 2(1):11–22
27. Mirjalili S, Jangir P, Saremi S (2017) Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. Appl Intell 46(1):79–95. https://doi.org/10.1007/s10489-016-0825-8
28. Parsons MG, Scott RL (2004) Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. J Ship Res 48(1):61–76. https://doi.org/10.1007/s10489-016-0825-8
29. Ponweiser W, Wagner T, Biermann D, Vincze M (2008) Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection. In: International Conference on Parallel Problem Solving from Nature, pp. 784–794. Springer. https://doi.org/10.1007/978-3-540-87700-4_78
30. Powell MJD (1994) A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Advances in Optimization and Numerical Analysis, pp. 51–67. Springer Netherlands. https://doi.org/10.1007/978-94-015-8330-5_4
31. Regis RG (2014) Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Eng Optim 46(2):218–243
32. Regis RG, Shoemaker CA (2013) A quasi-multistart framework for global optimization of expensive functions using response surface models. J Global Optim 56(4):1719–1753. https://doi.org/10.1007/s10898-012-9940-1
33. Rehbach F, Zaefferer M, Naujoks B, Bartz-Beielstein T (2020) Expected improvement versus predicted value in surrogate-based optimization. arXiv preprint arXiv:2001.02957. https://doi.org/10.1145/3377930.3389816
34. Riquelme N, Von Lücken C, Baran B (2015) Performance metrics in multi-objective optimization. In: 2015 Latin American Computing Conference (CLEI), pp. 1–11. IEEE
35. Santana-Quintero LV, Montano AA, Coello CAC (2010) A review of techniques for handling expensive functions in evolutionary multi-objective optimization. Comput Intell Expens Optim Prob 29–59
36. Singh P, Couckuyt I, Ferranti F, Dhaene T (2014) A constrained multi-objective surrogate-based optimization algorithm. In: 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE. https://doi.org/10.1109/cec.2014.6900581
37. Tanabe R, Oyama A (2017) A note on constrained multi-objective optimization benchmark problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1127–1134. IEEE
38. Urquhart M, Ljungskog E, Sebben S (2020) Surrogate-based optimisation using adaptively scaled radial basis functions. Appl Soft Comput 88:106050
39. de Winter R (2020) Roydezomer/samo-cobra: Release with new experiments. https://doi.org/10.5281/zenodo.5105636
40. de Winter R, Furustam J, Bäck T, Muller T (2021) Optimizing ships using the holistic accelerated concept design methodology. In: Okada T, Suzuki K, Kawamura Y (eds) Practical design of ships and other floating structures. Springer Singapore, Singapore, pp 38–50
41. de Winter R, van Stein B, Bäck T (2021) Samo-cobra: A fast surrogate assisted constrained multi-objective optimization algorithm. In: Ishibuchi H, Zhang Q, Cheng R, Li K, Li H, Wang H, Zhou A (eds) Evolutionary multi-criterion optimization. Springer International Publishing, Cham, pp 270–282
42. de Winter R, van Stein B, Dijkman M, Bäck T (2018) Designing ships using constrained multi-objective efficient global optimization. In: International Conference on Machine Learning, Optimization, and Data Science, pp. 191–203. Springer. https://doi.org/10.1007/978-3-030-13709-0_16
43. Yang Y, Liu J, Tan S (2021) A multi-objective evolutionary algorithm for steady-state constrained multi-objective optimization problems. Appl Soft Comput 101:107042
44. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength pareto evolutionary algorithm. TIK-report 103