SAMO-COBRA: A Fast Surrogate Assisted Constrained Multi-objective Optimization Algorithm

Roy de Winter[1,2(✉)], Bas van Stein[1], and Thomas Bäck[1]

[1] Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands
{r.de.winter,b.van.stein,t.h.w.baeck}@liacs.leidenuniv.nl
[2] C-Job Naval Architects, Hoofddorp, The Netherlands
r.dewinter@c-job.com

Abstract. This paper proposes a novel Self-Adaptive algorithm for Multi-Objective Constrained Optimization by using Radial Basis Function Approximations, SAMO-COBRA. The 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. The algorithm also uses hyper-parameter tuning on the fly to improve its local search strategy. In every iteration one solution is added and evaluated, resulting in a strategy requiring only a small number of function evaluations for finding a set of feasible solutions on the Pareto frontier. The proposed algorithm is compared to a wide set of other state-of-the-art algorithms (NSGA-II, NSGA-III, CEGO, SMES-RBF) on 18 constrained multi-objective problems. In the experiments we show that our algorithm outperforms the other algorithms in terms of achieved Hypervolume after given a fixed small evaluation budget. These results suggest that SAMO-COBRA is a good choice for optimizing constrained multi-objective optimization problems with expensive function evaluations.

Keywords: Constrained optimization · Multi-objective optimization · Optimization under limited budgets

1 Introduction

According to a survey about adaptive sampling for global meta-modeling, the interest in efficient global meta-models and adaptive sampling techniques has increased over the past years [17]. Additionally, preliminary results of a recently executed questionnaire [5] shows that real world problems often involve continuous optimization variables, constraints, and one or more objectives. The questionnaire responses also indicate that it is often the case that the topological characteristics of the objective space are unknown and long evaluation times make the problems hard to solve. However, to the author's best knowledge, there

H. Ishibuchi et al. (Eds.): EMO 2021, LNCS 12654, pp. 270–282, 2021.
https://doi.org/10.1007/978-3-030-72062-9_22

is no fast multi-objective constrained optimization algorithm capable of approximating the Pareto frontier (PF) with a small number of function evaluations.

In this paper, the authors propose a novel *Self-Adaptive algorithm for Multi-Objective Constrained Optimization by using Radial Basis Function Approximations* (SAMO-COBRA). The algorithm is designed to efficiently determine the Pareto frontier of constrained multi-objective problems. The algorithm models the constraints and objectives with independent Radial Basis Functions (RBFs) using different kernels, automatically chooses the best RBF-fit, automatically tunes hyper-parameters, and aims to find feasible Pareto-optimal solutions. Here a *constrained multi-objective problem* is defined as follows:

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

In this formulation k is the number of objectives, m is the number of constraints and d is the number of parameters. Without loss of generality, maximisation problems are transformed to minimization problems. A *feasible Pareto-optimal solution* is defined in Definition 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* $g_i(\mathbf{x}') \leq 0 \ \forall i \in \{1, \ldots, m\}$.

The algorithm performance is compared to the performance of NSGA-II [10], NSGA-III [15], CEGO [29], and SMES-RBF [8]. In the results we show that SAMO-COBRA only requires a small number of function evaluations to find feasible Pareto-optimal solutions. This makes SAMO-COBRA a good choice when optimizing problems with expensive or time consuming function evaluations.

The remainder of this paper is organized as follows: In Sect. 2 related work is discussed. In Sect. 3 the newly proposed SAMO-COBRA algorithm is described. In Sect. 4 the experiments conducted to compare SAMO-COBRA to state-of-the-art algorithms are described. In Sect. 5 the results are reported and discussed, and in Sect. 6 the conclusions are drawn.

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 solving constrained single objective problems fast (e.g. SACOBRA [3]), for multi-objective optimization without efficient constraint handling techniques (e.g. SMS-EGO [21] and PAREGO [16]), or for constrained multi-objective optimization without using meta-models, leading to a lot of required function evaluations (e.g. NSGA-II [10], NSGA-III [15], SPEA2 [30], and SMS-EMOA [4]). Some recently proposed algorithms address

all three requirements, however with computational efforts that grows cubically in iterations and exponentially for each additional decision parameter due to the use of Kriging surrogates (e.g. CEGO [29], ECMO [25]).

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 [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.

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, and multiple complex constraints 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-front (PF). The pseudocode of SAMO-COBRA can be found in Algorithm 1, a Python implementation can be found on GitHub [28]. In the subsections below, the algorithm is explained in more detail.

3.1 Initial Design

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 with an as small as possible initial sample [6]. Several experiments confirmed the hypothesis that the smallest possible initial sample size also leads to the best results when applied on most 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 $d+1$ initial samples. It is therefore advised when using SAMO-COBRA to create an initial Halton sample of size $d+1$ before the sequential optimization procedure starts. Every sample in the initial design is then evaluated (line 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

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}, $RBF_{kernels}$ (φ) = {*cubic*, *gaussian*, *multiquadric*, *invquadric*, *invmultiquadric*, *thinplatespline*} **Output:** Evaluated feasible Pareto efficient solutions.

Function SAMO-COBRA(f, g, $[\mathbf{lb}, \mathbf{ub}]$, **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, $\mathbf{G} \in \mathbb{R}^{m \times N}$
  for $i \leftarrow 1$ **to** $k + m$ **do**
    $RBF_i^* \leftarrow$ (φ=*cubic*, PLOG=0) ▷ Initialize RBF configuration
  end
  while $N < N_{max}$ **do**
    $\hat{\mathbf{X}} \leftarrow \text{SCALE}(\mathbf{X}, [-1, 1]^d)$ ▷ Scale input space to $[-1, 1]^d$
    $\tilde{\mathbf{F}} \leftarrow \text{PLOG}(\mathbf{F})$ ▷ See function plog in Eq. (5)
    $\tilde{\mathbf{G}} \leftarrow \text{PLOG}(\mathbf{G})$ ▷ See function plog in Eq. (5)
    $\hat{\mathbf{F}} \leftarrow \text{STANDARDIZE}(\mathbf{F})$ ▷ Standardize objective space
    $\hat{\mathbf{G}} \leftarrow \text{SCALE CONSTRAINT}(\mathbf{G})$ ▷ 0 remains feasibility boundary
    for $\varphi \in RBF_{kernels}$ **do** ▷ For each kernel
      for $i \leftarrow 1$ **to** k **do** ▷ for each objective
        $\hat{S}_i^{\varphi} \leftarrow \text{FITRBF}(\hat{\mathbf{X}}, \hat{\mathbf{F}}_{(i,\cdot)}, \varphi)$ ▷ Fit RBF without PLOG
        $\tilde{S}_i^{\varphi} \leftarrow \text{FITRBF}(\hat{\mathbf{X}}, \tilde{\mathbf{F}}_{(i,\cdot)}, \varphi)$ ▷ Fit RBF with PLOG
      end
      for $j \leftarrow 1$ **to** m **do** ▷ for each constraint
        $\hat{S}_{k+j}^{\varphi} \leftarrow \text{FITRBF}(\hat{\mathbf{X}}, \hat{\mathbf{G}}_{(j,\cdot)}, \varphi)$ ▷ Fit RBF without PLOG
        $\tilde{S}_{k+j}^{\varphi} \leftarrow \text{FITRBF}(\hat{\mathbf{X}}, \tilde{\mathbf{G}}_{(j,\cdot)}, \varphi)$ ▷ Fit RBF with PLOG
      end
    end
    $S^* \leftarrow \left\{ S_i^{(RBF_i^*)} \mid \forall i = 1, \ldots, (k+m) \right\}$ ▷ Apply best configuration
    $\mathbf{PF} \leftarrow \text{PARETO}(\mathbf{X}, \mathbf{F}, \mathbf{G})$ ▷ PF indicator see Definition 1, $\mathbf{PF} \in \{0, 1\}^N$
    $\mathbf{x}^* \leftarrow \text{MAXIMIZE}(\text{HV}, \mathbf{PF}, \mathbf{ref}, S^*)$ ▷ Find best new solution
    $\mathbf{x}_{new} \leftarrow \text{SCALE}(\mathbf{x}^*, [\mathbf{lb}, \mathbf{ub}])$ ▷ 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, $\mathbf{G} \in \mathbb{R}^{m \times N}$
    ▷ Get best RBF configuration and Squared Errors, Section 3.6
    $RBF^*, \mathbf{SE} \leftarrow \text{SELECTBESTRBF}(\mathbf{SE}, \hat{S}, \tilde{S}, \mathbf{x}^*, \mathbf{F}, \mathbf{G}, \mathbf{PF}, N)$
  end
return $(\mathbf{F}_{(\cdot,\mathbf{PF})}, \mathbf{G}_{(\cdot,\mathbf{PF})}, \mathbf{X}_{(\cdot,\mathbf{PF})})$

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 we keep the shape/width parameter ϵ for every individual RBF constant such as proposed by Urquhart et al. [27]. 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 [18]. In this work we use augmented RBFs 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 we can compute the weights $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ with Eq. 1 (Lines 16, 17, 20, 21 of Algorithm 1), we can for each unseen input $\mathbf{x}'$ interpolate/predict the function value (f') by using Eq. (3).

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

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, we can prevent computationally singular (ill-conditioned) coefficient matrices in

Eq. (1). In case we keep large values in the input space, 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 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).

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

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. We therefore standardize the objective values 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, we chose to scale every constraint output with as $c' = c/(\max(c) - \min(c))$, 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 [23]. Therefore, we transform the scores with the Plog transformation function. The extension to a matrix argument $\mathbf{Y}$ is defined componentwise, i.e., each matrix element y_{ij} is subject to Plog.

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

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. The COBYLA (Constrained Optimization BY

Linear Approximations) algorithm [22] is then used to search for a new feasible Pareto-optimal solution $\mathbf{x}' \in [\mathbf{lb},\mathbf{ub}]$. This is achieved by maximizing the hypervolume (HV) a solution $\mathbf{x}'$ adds to the HV between a predefined reference point and the already evaluated solutions on the PF (Line 26 of Algorithm 1). The HV contribution of a solution to the PF is computed with two infill criteria:

1. Compute all objective scores for a given solution $\mathbf{x}'$ with Eq. (3). Then with the interpolated objective scores compute the additional Predicted HV (PHV) score this solution adds to the PF.
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), which is similar to the Kriging $\mathcal{S}$-metric Selection (SMS) criterion from Emmerich et al. [4].

Besides the highest possible SMS or PHV score, the potential solution should not violate any of the constraints. This can easily be checked by using the RBF surrogates of the constraints and Eq. (3). COBYLA then searches for a solution which does not violate any of the constraints and has the highest SMS or PHV score. If no feasible solution can be found, the solution with the smallest constraint violation is evaluated. The best candidate is then evaluated on the real objective and constraint functions (Lines 27–31 of Algorithm 1).

3.5 Surrogate Exploration and RBF Adaptation Rules

The surrogate search budget and the number of starting points are updated and optimized every iteration. This is done to limit the time spend on exploring the surrogates and to further increase the chance of finding a solution that adds the most HV to the PF. The problem characteristics (number of variables, constraints, and objectives) influences the optimization problem complexity. Therefore, in the first iteration of SAMO-COBRA, 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.

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, so the error margin of the corresponding constraint is increased by 10%.

3.6 Select Best RBF

In every iteration, the best RBF kernel and transformation strategy is chosen (Line 32 of Algorithm 1). This 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 converged 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. We therefore experimented with this and confirmed that the approximation errors of the feasible Pareto-optimal solutions and the last four solutions should be considered. The approximation error of the last four solutions make sure that the algorithm does not get stuck on one RBF configuration and the error of the Pareto efficient solutions make sure that all the vicinity 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.

4 Experimental Setup

The SMS and PHV infill criteria in combination with the SAMO-COBRA algorithm are tested on 18 constrained multi-objective test functions. The two variants of the algorithm are compared to CEGO [29], NSGA-II [10], and NSGA-III [15]. Each algorithm is allowed to do $40 \cdot d$ function evaluations after which the HV between the PF and a fixed reference point is computed. The 18 test functions with their sources, their dimensions, feasibility rate, and the average algorithm iteration times are presented in Table 1. The iteration times of NSGA-II and NSGA-III are so low ($\sim$0.001 s) that they are not reported. The experiments are executed on a laptop with i7 processor, 4 cores and a clock speed of 2.5 GHz.

In the experiment we ran the algorithms 10, 10, 10, 100, 100 times with different seeds for the SAMO-COBRA (PHV), SAMO-COBRA (SMS), CEGO, NSGA-II, and NSGA-III algorithm respectively. This number of runs is chosen because the iteration time of SAMO-COBRA and CEGO are higher, and the HV after $40 \cdot d$ function evaluations vary a lot for NSGA-II and NSGA-III.

The implementation of the SMES-RBF algorithm is not provided. We therefore show in one additional experiment on GitHub [28], that SAMO-COBRA requires fewer function evaluations to achieve the same HV as SMES-RBF [8].

4.1 Algorithm Parameter Settings

As already mentioned in Sect. 3.1, a small initial Design of Experiments (DoE) leads to higher final HV. The SAMO-COBRA algorithm is therefore set up with

Table 1. Test functions with citation, the reference points, the number of objectives k, number of parameters d, number of constraints m, feasibility percentage $P(\%)$ based on 1 million random samples, mean running time per iteration in seconds.

Function	Reference point	k	d	m	P(%)	SAMO-COBRA	CEGO
BNH [7]	(140, 50)	2	2	2	96.92	3.5	10.5
CEXP [9]	(1, 9)	2	2	2	57.14	2.2	10.1
SRN [11]	(301, 72)	2	2	2	16.18	4.9	20.6
TNK [11]	(2, 2)	2	2	2	5.05	0.8	8.2
CTP1 [9]	(1, 2)	2	2	2	92.67	0.8	7.3
C3DTLZ4 [26]	(3, 3)	2	6	2	22.22	18.1	338
OSY [7,11]	(0, 386)	2	6	6	2.78	9.8	424
TBTD [13]	(0.1, 50000)	2	3	2	19.46	5.5	151
NBP [12]	(11150, 12500)	2	2	5	41.34	2.8	9.7
DBD [13]	(5, 50)	2	4	5	28.55	34.9	159
SPD [20]	(16, 19000, −260000)	3	6	9	3.27	34.6	405
CSI [15]	(42, 4.5, 13)	3	7	10	18.17	45.3	1h+
SRD [19]	(7000, 1700)	2	7	10	96.92	63.2	172
WB [13]	(350, 0.1)	2	4	5	35.28	19.1	258
BICOP1 [8]	(9, 9)	2	10	1	100	24.2	1h+
BICOP2 [8]	(70, 70)	2	10	2	10.55	12.1	1h+
TRIPCOP [8]	(34, −4, 90)	3	2	3	15.85	3.8	9.6
WP [15]	(83000, 1350, 2.85, 15989825, 25000)	5	3	7	92.06	16.1	225

d+1 initial Halton samples. The Kriging surrogates of CEGO require a minimum DoE of size 2·d+1. CEGO therefore starts after a DoE of 2·d+1 Latin Hypercube Samples. Other hyper-parameters of CEGO are set as originally described [29].

The implementation of Platypus [14] is used for NSGA-II and NSGA-III. The algorithms have several sensitive parameters which influence the final results. We therefore did a grid search for different population sizes and generations for NSGA-II, and a grid search for the number of division that controls the spacing of the reference points for NSGA-III. For the sake of brevity, only the results which gave the highest HV after $40 \cdot d$ function evaluations are reported.

For more detailed parameter settings, the raw results, and the statistical comparisons, please refer to the SAMO-COBRA GitHub page [28].

5 Results

Table 2 shows the mean HV after $40 \cdot d$ function evaluations. The Kruskal-Wallis test confirmed that there is at least one result statistically different from the results from the other algorithms, a post-hoc comparison is done with a

Table 2. Mean HV after $40 \cdot d$ function evaluations for each algorithm on each test function. PHV and SMS represents 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 colors represent: $p \leq 0.001$, $p \leq 0.01$, $p \leq 0.05$. Red shows that the algorithm took more than 24 h.

Function	PHV	SMS	CEGO	NSGA-II	NSGA-III
BNH	**5256.0**	5251.0	5218.0	5089.7	4848.8
CEXP	**3.7968**	3.7967	3.7658	3.1544	2.9561
SRN	**62385**	62377	62307	54233	52585
TNK	8.0430	**8.0474**	8.0309	6.4282	6.2948
CTP1	**1.3026**	1.3023	1.2972	1.1929	1.1864
C3DTLZ4	6.3016	**6.4697**	6.2664	5.2489	5.2500
OSY	**100577**	100458	100181	46631	42204
TBTD	4029.3	4027.5	**4055.0**	3421.5	3446.0
NBP	$\mathbf{1.0785 \cdot 10^{8}}$	$1.0785 \cdot 10^{8}$	$1.0784 \cdot 10^{8}$	$9.8573 \cdot 10^{7}$	$9.5816 \cdot 10^{7}$
DBD	**228.77**	228.34	227.85	218.47	218.33
SPD	$\mathbf{3.8859 \cdot 10^{10}}$	$3.7602 \cdot 10^{10}$	$3.4057 \cdot 10^{10}$	$2.6069 \cdot 10^{10}$	$2.5689 \cdot 10^{10}$
CSI	**27.800**	25.167	*terminated*	17.296	16.993
SRD	**4205165**	4164992	4148333	2766367	2673599
WB	34.395	34.392	**34.522**	34.0270	33.995
BICOP1	**80.664**	78.160	*terminated*	67.6506	70.911
BICOP2	**4834.33**	4822.13	*terminated*	4816.12	4816.72
TRIPCOP	20610	**20611**	20609	20101	19982
WP	$\mathbf{1.5991 \cdot 10^{19}}$	$1.5698 \cdot 10^{19}$	$1.5350 \cdot 10^{19}$	$1.1623 \cdot 10^{19}$	$1.1797 \cdot 10^{19}$

Wilcoxon rank-sum test (with Bonferroni correction) to find out if there is a significant difference between the best and the lesser results.

Note that the exploiting strategy of the PHV infill criteria leads in most cases to the highest hv after $40 \cdot d$ function evaluations. 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. [24]. He showed 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. With the results as presented in Table 2 we can give an advice based on empirical results, as follows: 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 objectives, making it less likely to get stuck in a local optimum, and making the uncertainty quantification method redundant.

6 Conclusion and Future Work

In this paper the authors propose SAMO-COBRA, a fast optimization algorithm capable of handling multiple objectives, multiple constraints, and continuous decision variables. Two variants of the algorithm are evaluated and compared to CEGO, NSGA-II, NSGA-III, and SMES-RBF. The experiments show that SAMO-COBRA outperforms the other algorithms in terms of achieved HV after a fixed number of function evaluations. These characteristics make SAMO-COBRA a good choice when optimizing a real-world optimization problem with multiple objectives, multiple constraints, continuous decision variables, and expensive (in terms of time and/or money) function evaluations.

Besides these results, the authors also give the recommendation that an uncertainty quantification method in multi-objective optimization is often not required. This is the case, because the algorithm already automatically explores more of the objective space compared to when optimizing single objective problems, which makes the algorithm less likely to get stuck in a local optimum.

Further research efforts will be put into parallelizing the function evaluations and parallelizing the surrogate exploration so that the wall clock time can be further decreased. Besides parallelization, effort will be put into dealing with mixed integer decision parameters and multi fidelity simulation software.

References

1. Bagheri, S., Konen, W., Bäck, T.: Online selection of surrogate models for constrained black-box optimization. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8. IEEE (2016)
2. Bagheri, S., Konen, W., Bäck, T.: Comparing kriging and radial basis function surrogates. In: Proceedings 27 Workshop Computational Intelligence, pp. 243–259 (2017)
3. Bagheri, S., Konen, W., Emmerich, M., Bäck, T.: Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. Appl. Soft Comput. **61**, 377–393 (2017). https://doi.org/10.1016/j.asoc.2017.07.060
4. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. **181**(3), 1653–1669 (2007). https://doi.org/10.1016/j.ejor.2006.08.008
5. van der Blom, K., et al.: Towards realistic optimization benchmarks: a questionnaire on the properties of real-world problems. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion. GECCO 2020, New York, NY, USA, pp. 293–294. Association for Computing Machinery (2020)
6. Bossek, J., Doerr, C., Kerschke, P.: Initial design strategies and their effects on sequential model-based optimization. arXiv preprint arXiv:2003.13826 (2020)
7. Coello, C.A.C., Lamont, G.B., Van Veldhuizen, D.A., et al.: Evolutionary Algorithms for Solving Multi-objective Problems. Genetic and Evolutionary Computation Series. GEVO, vol. 5. Springer, Boston (2007). https://doi.org/10.1007/978-0-387-36797-2
8. Datta, R., Regis, R.G.: A surrogate-assisted evolution strategy for constrained multi-objective optimization. Expert Syst. Appl. **57**, 270–284 (2016). https://doi.org/10.1016/j.eswa.2016.03.044

9. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms, vol. 16. John Wiley & Sons, New York (2001)
10. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
11. Deb, K., Pratap, A., Meyarivan, T.: Constrained test problems for multi-objective evolutionary optimization. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) EMO 2001. LNCS, vol. 1993, pp. 284–298. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44719-9_20
12. Forrester, A., Sobester, A., Keane, A.: Engineering Design via Surrogate Modelling: A Practical Guide. John Wiley & Sons, New York (2008). https://doi.org/10.2514/4.479557
13. Gong, W., Cai, Z., Zhu, L.: An efficient multiobjective differential evolution algorithm for engineering design. Struct. Multi. Optim. **38**(2), 137–157 (2009). https://doi.org/10.1007/s00158-008-0269-9
14. Hadka, D.B.: Platypus: multiobjective optimization in python (2020). https://platypus.readthedocs.io/
15. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based non dominated sorting approach, part II: handling constraints and extending to an adaptive approach. IEEE Trans. Evol. Comput. **18**(4), 602–622 (2014). https://doi.org/10.1109/tevc.2013.2281534
16. Knowles, J.: ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. Evol. Comput. **10**(1), 50–66 (2006). https://doi.org/10.1109/tevc.2005.851274
17. Liu, H., Ong, Y.-S., Cai, J.: A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. Struct. Multi. Optim. **57**(1), 393–416 (2017). https://doi.org/10.1007/s00158-017-1739-8
18. Micchelli, C.A.: Interpolation of scattered data: distance matrices and conditionally positive definite functions. Constr. Approximation **2**(1), 11–22 (1986)
19. Mirjalili, S., Jangir, P., Saremi, S.: Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. Appl. Intell. **46**(1), 79–95 (2016). https://doi.org/10.1007/s10489-016-0825-8
20. Parsons, M.G., Scott, R.L.: Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. J. Ship Res. **48**(1), 61–76 (2004). https://doi.org/10.1007/s10489-016-0825-8
21. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 784–794. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87700-4_78
22. Powell, M.J.D.: A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Gomez, S., Hennart, J.P. (eds.) Advances in Optimization and Numerical Analysis. MAIA, vol. 275, pp. 51–67. Springer, Netherlands (1994). https://doi.org/10.1007/978-94-015-8330-5_4
23. Regis, R.G., Shoemaker, C.A.: A quasi-multistart framework for global optimization of expensive functions using response surface models. J. Global Optim. **56**(4), 1719–1753 (2013). https://doi.org/10.1007/s10898-012-9940-1
24. Rehbach, F., Zaefferer, M., Naujoks, B., Bartz-Beielstein, T.: Expected improvement versus predicted value in surrogate-based optimization. arXiv preprint arXiv:2001.02957 (2020). https://doi.org/10.1145/3377930.3389816

25. Singh, P., Couckuyt, I., Ferranti, F., Dhaene, T.: A constrained multi-objective surrogate-based optimization algorithm. In: 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE (2014). https://doi.org/10.1109/cec.2014.6900581
26. Tanabe, R., Oyama, A.: A note on constrained multi-objective optimization benchmark problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1127–1134. IEEE (2017). https://doi.org/10.1109/cec.2017.7969433
27. Urquhart, M., Ljungskog, E., Sebben, S.: Surrogate-based optimisation using adaptively scaled radial basis functions. Appl. Soft Comput. **88**, 106050 (2020)
28. de Winter, R.: SAMO-COBRA: self-adaptive algorithm for multi-objective constrained optimization by using radial basis function approximations (2020). https://doi.org/10.5281/zenodo.4281140
29. de Winter, R., van Stein, B., Dijkman, M., Bäck, T.: Designing ships using constrained multi-objective efficient global optimization. In: Nicosia, G., Pardalos, P. (eds.) Machine Learning, Optimization, and Data Science. LNCS, vol. 11331. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-13709-0_16
30. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength pareto evolutionary algorithm. TIK-report, vol. 103 (2001)