Surrogate-Assisted Partial Order-Based Evolutionary Optimisation

Vanessa Volz[1], Günter Rudolph[1], and Boris Naujoks[2]

[1] TU Dortmund University
{vanessa.volz, guenter.rudolph}@tu-dortmund.de

[2] TH Köln - University of Applied Sciences
boris.naujoks@th-koeln.de

Abstract. In this paper, we propose a novel approach (SAPEO) to support the survival selection process in evolutionary multi-objective algorithms with surrogate models. The approach dynamically chooses individuals to evaluate exactly based on the model uncertainty and the distinctness of the population. We introduce multiple SAPEO variants that differ in terms of the uncertainty they allow for survival selection and evaluate their anytime performance on the BBOB bi-objective benchmark. In this paper, we use a Kriging model in conjunction with an SMS-EMOA for SAPEO. We compare the obtained results with the performance of the regular SMS-EMOA, as well as another surrogate-assisted approach. The results open up general questions about the applicability and required conditions for surrogate-assisted evolutionary multi-objective algorithms to be tackled in the future.

Keywords: partial order, multi-objective, surrogates, evolutionary algorithms, bbob

1 Introduction

Surrogate model-assisted evolutionary multi-objective algorithms (SA-EMOAs) are a group of fairly recent but popular approaches[1] to solve multi-objective problems with expensive fitness functions. Using surrogate model predictions of the function values instead of / to complement exact evaluations within an evolutionary algorithm (EA) can save computational time and in some cases make the problem tractable at all.

Many EMOAs only consider objective values for the purpose of sorting and then selecting the best individuals in a population. Assuming that individuals can confidently be distinguished based on surrogate model predictions, knowing the individuals' exact objective values is not necessary. Under this assumption, the algorithm and its evolutionary path would not be affected at all by trusting the predicted sorting, and the computational budget could be reduced.

In this paper, we present a novel approach to integrate surrogate models

[1] Workshop on the topic in 2016: http://samco.gforge.inria.fr/doku.php

and evolutionary (multi-objective) algorithms (dubbed SAPEO for **S**urrogate-**A**ssisted **P**artial Order-Based **E**volutionary **O**ptimisation[2]) that seeks to reduce the number of function evaluations while simultaneously controlling the probability of detrimental effects on the solution quality. The idea is to choose the individuals for exact evaluation dynamically based on the model uncertainty and the distinctness of the population. Preliminary experiments on single-objective problems showed promising results, so in this paper, we investigate the approach in the currently sought-after multi-objective context. We also present different versions that allow differing levels and types of uncertainties for the survival selection process, which, in turn, can potentially effect the solution quality.

In the following, we describe our extensive analysis of the anytime performance of SAPEO using the BBOB-BIOBJ benchmark (refer to section 2.2), focusing on use cases with low budgets to simulate applications with expensive functions. Our SAPEO implementation[3] uses a Kriging surrogate model [14] in conjunction with the SMS-EMOA [2]. We further compare the algorithm and its variants to the underlying SMS-EMOA and an SA-EMOA approach called *pre-selection* [4] (SA-SMS in the following).

We specifically investigate if and under which conditions SAPEO outperforms the SMS-EMOA and SA-SMS in terms of the hypervolume indicator that all algorithms use to evaluate populations. Surprisingly, none of the surrogate-assisted algorithms can convincingly beat out the baseline SMS-EMOA, even for small function budgets. This result opens up questions about SA-EMOAs in general and about the necessary quality of the integrated surrogate models.

A potential explanation for this performance is the increased uncertainty of the surrogate model predictions when compared to the single-objective experiments. Thus, we analyse the effects of prediction uncertainty on the overall performance of the SA-EMOAs. In the future, the resulting insights could become important when (1) deciding whether using a surrogate model is beneficial on a given problem at all and (2) when choosing the sample size and further parameters for the model. This is especially crucial for multi- and many-objective problems, where learning an accurate surrogate model becomes increasingly expensive and thus renders analysing the trade-off between surrogate model computation and function evaluations critical. Additionally, the stated questions and insights are also relevant for noisy optimisation problems where uncertainty can be reduced by repeated evaluations (although not eliminated).

In the following, we present related work in section 2 and introduce the proposed SAPEO algorithm in section 3. The description and visualisation of the benchmarking results are found in section 4. Section 5 concludes the paper with an analysis of the results and lists open research problems.

[2] Acknowledgement: The SAPEO concept was developed during the SAMCO Workshop in March 2016 at the Lorentz Center (Leiden, NL). https://www.lorentzcenter.nl This work is part of a project that has received funding from the European Unions Horizon 2020 research and innova- tion program under grant agreement No 692286.

[3] Code and visualisations available at: http://url.tu-dortmund.de/volz

2 Background and Related work

2.1 Surrogate-Assisted Evolutionary Multi-Objective Optimisation

Let $X_1, \ldots, X_\lambda \in \mathbb{R}^n$ be a population and the corresponding fitness function $f : \mathbb{R}^n \to \mathbb{R}^d$. General concepts of multi-objective optimisation will not be discussed here (refer to e.g. [16]). We will be referring to Pareto-dominance as $\preceq$, and to its weak and strong versions as $\precsim$ and $\prec$, respectively. We use the same notation to compare vectors in objective space $\mathbb{R}^d$, i.e. let $a, b \in \mathbb{R}^d$, then $a \preceq b \iff \forall k \in \{1 \ldots d\} : a_k \leq b_k \wedge \exists k \in \{1 \ldots d\} : a_k < b_k$.

Surrogate-assisted evolutionary multi-objective optimisation is surveyed in [7,8], where several approaches for the integration of surrogates and EMOAs are described. According to the surveys, the selection approaches can generally be divided into individual-based (e.g. SAPEO), generation-based, and population-based strategies. Additionally, there is pre-selection (e.g. SA-SMS [4]), which is similar to individual-based strategies but does not retain any individuals with uncertain fitness values.

However, neither of the cited surveys, nor [9], features any algorithm that chooses to propagate uncertainty instead of assuming a distribution and using aggregated metrics such as the expected value. In [10], uncertainty propagation is implemented for noisy optimisation problems using a partial order based on confidence intervals (or hypercubes in higher dimensions) as examined in [13]. The only work transferring this approach to SA-EMOAs we are aware of is GP-DEMO [11] and the authors' previous publications who use a differential evolution algorithm.

Apart from the differences owed to the underlying optimisation algorithms (e.g. crowding distance vs. hypervolume), GP-DEMO is very similar to the SAPEO variant that allows the least uncertainty (SAPEO-uf-ho, cf. 4.1). An important difference, however, is SAPEOs dynamic adaptation of allowed uncertainty throughout the runtime of the algorithm and executing different partial orders in sequence. In addition to the partial orders inspired by [13] used in both [11] and SAPEO, we propose another order that interprets the confidence interval bounds as objectives and thus deals differently with overlapping intervals. A further difference is the choice of samples for the surrogate models: GP-DEMO uses the Pareto front, whereas SAPEO always uses a local model relative to the solution in question.

2.2 Benchmarking with BBOB

BBOB-BIOBJ is a bi-objective Black-Box Optimisation Benchmarking test suite [15]. It consists of 55 bi-objective functions that are a combination of 10 of the 24 single-objective functions in the BBOB test suite established in 2009 [6]. In order to measure general algorithm performance across function types, single-objective functions were chosen such that the resulting benchmark would be diverse in terms of separability, conditioning, modality and global structure [6]. Based on these properties, the single-objective functions are divided into 5

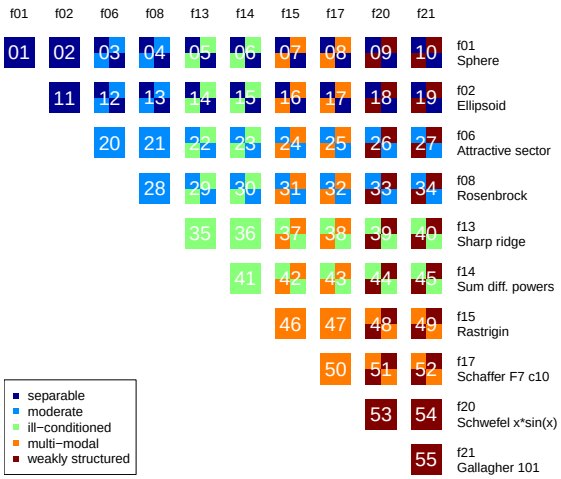

Fig. 1: The 55 BBOB-BIOBJ functions are combinations of 10 single-objective functions (on the top and right). The groups the single-objective and the resulting bi-objective functions belong to are colour-coded according to the legend.

function groups, from which 2 functions are chosen each. The resulting problems and corresponding properties are visualised in figure 1.

In an effort to measure performance on the different function types more accurately, each of the functions in the bi-objective test suite has 10 instances that differ in terms of some properties, e.g. the location of optima. As an effect, the scale of the optima and achievable absolute improvement of the objective values also vary significantly across instances (thus also between objectives). The robustness of an algorithm's performance on a function group can be evaluated with a higher confidence by testing on multiple members of that group.

All of the functions in the test suites are defined for search spaces of multiple dimensions, of which we will be considering dimensions $2, 3, 5, 10$ and 20 in order to be able to evaluate a wide range of problem sizes. The search space of each function is limited to $[-100, 100] \subset \mathbb{R}$ per dimension in BBOB-BIOBJ.

The performance of an algorithm on the benchmarking suite is measured using a quality indicator expressing both the size of the obtained Pareto set and the proximity to a reference front. Since the true Pareto front is not known for the functions in the test suite, an approximation is obtained by combining all known solutions from popular algorithms. The ideal and nadir points are known, however, and used to normalise the quality indicator to enable comparisons across functions [3]. The metric reported as a performance measure for the algorithm is called precision. It is the difference of the quality indicator of the reference set I_{ref} and the indicator value of the obtained set. 58 target precisions are fixed and the number of function evaluations needed to achieve them is reported during a benchmark run. This way, the COCO platform enables an anytime comparison of algorithms, i.e. an evaluation of algorithm performance for each target precision and number of function evaluations [3].

3 Surrogate-Assisted Partial Order-Based Evolutionary Optimisation

3.1 Formal description

Let $\tilde{f}(X_i) \in \mathbb{R}^d$ be the predicted fitness for individual X_i as computed by a local surrogate model with uncertainty $\tilde{\sigma}_i$ modelled by

$$\tilde{f}_k(X_i) = f_k(X_i) + e_i, \quad e_i \sim \mathcal{N}(0, \tilde{\sigma}_i),\ k \in \{1 \dots d\}.$$

Assuming (Assumption **A1**) that the assumptions made by Kriging models [14] hold and $\tilde{\sigma}_i$ was estimated correctly, it follows that

$$\mathbb{P}\Big(f_k(x_i) \in [\tilde{f}_k(x_i) - u_i, \tilde{f}_k(x_i) + u_i] \Big) = 1 - \alpha \quad \text{with}$$
$$u_i = \tilde{\sigma}_i z \left(1 - \frac{\alpha}{2}\right) \tag{1}$$

since $P(|e_i| \leq u_i) = 1 - \alpha$. Here, z denotes the quantile function of the standard normal distribution. In case the objective values are stochastically independent, which is true for the BBOB-BIOBJ benchmark, we can therefore conclude that $f(X_i)$ lies within the hypercube (or bounding box) bounded by the described confidence interval in each dimension with probability $(1 - \alpha)^d$.

Assuming the function values lie within the defined hypercubes (Assumption **A2**), we can distinguish individuals confidently just based on the predicted hypercubes. Of course, in case of large uncertainties $\tilde{\sigma}_i$, a distinction can rarely be meaningful. To combat this problem, SAPEO introduces a threshold ε_g for the uncertainty that is adapted in each iteration g of the EMOA and decreased over the runtime of the algorithm depending on the distinctness of the population (more details in section 3.2). Individuals X_i with an uncertainty higher than the threshold ($u_i > \varepsilon_g$) are evaluated exactly in generation g.

To distinguish between individuals, we propose binary relations incorporating the information on confidence bounds to varying degrees. All of these relations induce a strict partial order (irreflexivity, transitivity) on a population, including and akin to the Pareto dominance commonly used in EMOAs for the first part of the selection process. We analyse the proposed relations in terms of the probability and magnitude of a **sorting error** e_o that per our definition are the pairwise differences between the different orders induced by Pareto dominance and the proposed relations, respectively. We define the probability $\mathbb{P}(e_{o,r}^{i,j})$ of a sorting error made by relation $\preceq_r$ on individuals X_i, X_j and the magnitude of the error $e_{o,r}^{i,j}$:

$$\mathbb{P}(e_{o,r}^{i,j}) = \mathbb{P}\Big(X_i \not\preceq X_j \ \mid \ X_i \preceq_r X_j \Big)$$
$$e_{o,r}^{i,j} = \begin{cases} |f(X_i) - f(X_j)| & \text{if } (X_i \preceq_r X_j) \wedge (X_i \not\preceq X_j) \\ 0 & \text{else} \end{cases}$$

A single or more sorting errors can but do not have to lead to **selection errors** e_s, where the individuals selected differ from the baseline comparison. This type of error will not be analysed in this paper, but it is bounded by e_o.

We define:

$\preceq_f$*: Pareto dominance on function values* This relation is the standard in EMOAs and, since only f is considered, it is obvious that $\mathbb{P}(e_{o,f}^{i,j}) = 0$.

$$X_i \preceq_f X_j := f(X_i) \preceq f(X_j)$$

$\preceq_u$*: Confidence interval dominance* (cf. [11,13]) Assuming **A2**, if

$$X_i \preceq_u X_j := \bigwedge_{k\in[1\dots d]} \tilde{f}_k(X_i) + u_i < \tilde{f}_k(X_j) - u_j$$

holds, it is guaranteed that $X_i \prec X_j$. Assuming stochastic independence of the errors on predicted uncertainty, we can compute an upper bound for the probability of sorting errors per dimension:

$$\begin{aligned}\mathbb{P}(e_{o,u}^{i,j}(k)) &= \mathbb{P}\Big(f_k(X_i) \geq f_k(X_j) \mid X_i \preceq_u X_j \Big) \\ &\leq \mathbb{P}\Big(f_k(X_i) \geq \tilde{f}_k(X_i) + u_i \ \vee \ f_k(X_j) \leq \tilde{f}_k(X_j) - u_j \Big) \\ &\leq \frac{\alpha}{2} + \frac{\alpha}{2} = \alpha\end{aligned}$$

Only if the confidence hypercubes of two individuals intersect, the probability of them being incomparable is greater than 0. Because of the way $X_i \preceq_u X_j$ is defined, this is only possible if a sorting error is made in every dimension. It follows that $\mathbb{P}(e_{o,u}^{i,j}) \leq \alpha^d$ assuming **A1**, making sorting errors controllable.

$\preceq_c$*: Confidence interval bounds as objectives* Another way of limiting the prediction errors potentially perpetuated through the algorithm is to limit the magnitude of the sorting error. For this reason we define:

$$\begin{aligned}X_i \preceq_c X_j := \bigwedge_{k\in[1\dots d]} \begin{pmatrix}\tilde{f}_k(X_i)-u_i \\ \tilde{f}_k(X_i)+u_i\end{pmatrix} &\precsim \begin{pmatrix}\tilde{f}_k(X_j)-u_j \\ \tilde{f}_k(X_j)+u_j\end{pmatrix} \\ \wedge\, \exists k \in [1\dots d] : \begin{pmatrix}\tilde{f}_k(X_i)-u_i \\ \tilde{f}_k(X_i)+u_i\end{pmatrix} &\preceq \begin{pmatrix}\tilde{f}_k(X_j)-u_j \\ \tilde{f}_k(X_j)+u_j\end{pmatrix}\end{aligned}$$

Under assumption **A2**, the error per dimension is bounded by the length of intersection of the confidence intervals, which is in turn bounded by the width of the smaller interval. Therefore, it holds that $e_{o,c}^{i,j} \leq 2^d \min(u_i, u_j)^n$.

$\preceq_p$*: Pareto dominance on predicted values* This relation is the most straightforward, but it does not take the uncertainties of the predictions into account.

$$X_i \preceq_p X_j := \tilde{f}(X_i) \preceq \tilde{f}(X_j)$$

Assuming **A2** again, a sorting error can only be committed if the confidence intervals intersect. Because of the symmetric nature of the interval, it holds that $e_{o,p}^{i,j} \leq (u_i + u_j)^d$, as the magnitude of the sorting error is again bounded by the confidence interval widths.

$\preceq_o$*: Pareto dominance on lower bounds* This optimistic relation was motivated by [4] where SA-EMOAs performed better using $\preceq_o$ instead of $\preceq_p$.

$$X_i \preceq_o X_j := \bigwedge_{k\in[1\ldots d]} \tilde{f}_k(X_i) - u_i \leq \tilde{f}_k(X_j) - u_j$$
$$\wedge\, \exists k \in [1\ldots d] : \tilde{f}_k(X_i) - u_i < \tilde{f}_k(X_j) - u_j$$

The maximum error occurs when the lower confidence interval bounds are close together, but in the wrong order, making $e^{i,j}_{o,o} \leq 2^n \max(u_i, u_j)^d$.

Now assume we have obtained a strict partial order based on any of the given binary relations. Let r_c be the rank of the μ-th individual. Then, all individuals with rank less than r_c can confidently (with maximum errors as described above) be selected. In case a selection has to be made from the individuals with the critical rank r_c, one option is to apply another dominance relation to the individuals in question in hopes that the required distinction can be made. In case of $\prec_f$, this always means evaluating uncertain individuals until a confident distinction can be made according to the previous relation.

Another option is to use a relation inducing a total preorder (transitivity, totality) as a secondary selection criterion (and random choice in case of further ties) as most EMOAs do. Again incorporating different information, we have tested the following hypervolume-based (hv) relations for this purpose:

- Hypervolume contribution of objective values:
  $X_i \leq_{ho} X_j := hv(f_o(X_i)) \geq hv(f_o(X_j))$, where $f_o(X_i) = \begin{cases} f(X_i) & u_i = 0 \\ \tilde{f}(X_i) & \text{else} \end{cases}$
- Hypervolume contribution of confidence interval bounds:
  $X_i \leq_{hc} X_j := \prod_{k\in[1\ldots d]} hv\left(\begin{smallmatrix} \tilde{f}_k(X_i)-u_i \\ \tilde{f}_k(X_i)+u_i \end{smallmatrix}\right) \geq \prod_{k\in[1\ldots d]} hv\left(\begin{smallmatrix} \tilde{f}_k(X_j)-u_j \\ \tilde{f}_k(X_j)+u_j \end{smallmatrix}\right)$

3.2 SAPEO algorithm

Algorithm 1 describes the basic SAPEO algorithm, which any EMOA and surrogate model with uncertainty estimates can be plugged into. As inputs, the algorithm receives the fitness function `fun`, the number of points considered for the surrogate model `local_size` and the `budget` of function evaluations. The order of dominance relations (`strategies`) and the secondary criterion (`scnd_crit`) are used for selection (cf. section 3.1). The output is the final population.

The algorithm starts with mandatory data structures; the population is initialised randomly (line 1) and evaluated using the considered fitness function `fun` (line 2), the EMOA is set up (line 3) and the error tolerance ε as well as the generation counter are set to their initial values (line 4).

The core optimisation loop starts in line 5 and stops if either the considered optimiser terminates (due to the allocated budget or convergence), but not while both the error tolerance ε is larger than 0 and there are function evaluations left to avoid convergence on imprecise values. Within the loop, new candidate solutions X are first generated by the optimisation algorithm (line 6) and evaluated

Algorithm 1 SAPEO

Input: fun, local_size, budget, strategies, scnd_crit
Output: X_{final} ▷ final population

1: $X_0.genome \Leftarrow [random(n) : i \in 1 \dots pop\_size]$ ▷ random initialisation
2: $X_0.f \Leftarrow [fun(x) : x \in X_0]$; $X_0.e = 0$ ▷ evaluate sampled individuals
3: $O \Leftarrow init(X_0, budget)$ ▷ init optimiser with initial population
4: $\varepsilon_0 \Leftarrow \infty$; $g \Leftarrow 1$ ▷ init error tolerance, generation counter
5: **while** $(\neg O.stop()) \vee (\varepsilon > 0 \wedge budget > 0)$ **do**
6: $X_g.phenome \Leftarrow O.evolve(X_{g-1})$ ▷ get new population
7: $X_g.f, X_g.e \Leftarrow [model(x, knearest(x, X[X.e == 0], local\_size)) : x \in X_g]$
8: ▷ predict value, error with surrogate from evaluated neighbours
9: $\varepsilon_g \Leftarrow \min(e_{g-1}, \alpha\text{-percentiles}(\text{diff}(X_g.f))$ ▷ update error tolerance
10: **for** $x \in X_g$ **do**
11: **if** $x.e > \varepsilon_g \vee O.select(X_g, strategy, scnd\_crit) ==$ NULL **then**
12: $x.f = fun(x)$ ▷ evaluate individual
13: $bbob.recommend(X[X.e > 0, \text{last}])$ ▷ recommend solution
14: **end if**
15: **end for**
16: $X_g = O.select(X_g, strategy, scnd\_crit)$ ▷ SAPEO survival selection
17: $g = g + 1$ ▷ increase generation counter
18: **end while**
19: $X_{\text{final}} = X_{g-1}; X_{\text{final}}.f = [fun(x) : x \in X_{\text{final}}]$ ▷ evaluate final population

based on a local surrogate model trained from the `local_size` evaluated individuals closest in design space (line 8). The predicted function and the expected model errors (cf. equation 1) are stored. The error tolerance threshold is then adapted (line 9). We reduce the threshold during the course of the algorithm in order to limit the probability of sorting errors with large effects on the final population. Therefore, ε_g is the minimum of the previous threshold ε_{g-1} and the α-percentiles of the euclidian distances in objective space per dimension. The distances are a way to measure the distinctness of a population and thus the potential of overlapping confidence intervals. By adapting ϵ_g accordingly, we reduce the number and magnitude of potential sorting errors.

If any of the individuals in the population need to be evaluated - either because the predicted uncertainty is above the threshold or because the individuals cannot be distinguished (see line 11) - they are evaluated in line 12 and updated accordingly. In order to simulate anytime behaviour of the algorithm, each time a solution is evaluated, an individual is recommended to the BBOB framework in line 13. This serves the purpose of measuring the solution quality of the algorithm had it been stopped at the time more accurately.

The set of candidate solutions, along with the (predicted but reasonably certain) function values and the expected prediction errors are then passed to the optimiser in line 16. Depending on the selected strategy, the optimiser then selects the succeeding population as described above with regard to the predicted function values and uncertainties and resumes its regular process.

Finally, after the optimisation loop terminates, the function values of the individuals in the final population are computed using the real fitness function in line 19, in case there are any individuals left that have not been evaluated.

4 Evaluation

4.1 Experimental Setup

Each experiment was run with 550 parallel jobs that took less than 3 hours each with specifications according to table 1. Since the performance is strictly measured in terms of function evaluations (target precision reached per function evaluation, cf. section 2.2), the runtime does not influence it.

Table 1: Experiment specifications and parameters

budget	1000 per dimension (usecase: expensive function)
variation operators	standard for all algorithms (cf. [2])
populations size	100 (as suggested in [4])
sample size for surrogate	15 (due to computational concerns)[4]
number of candidate offspring	15 (for SA-SMS, same as sample size)
correlation assumption	squared exponential
trend assumption	constant
regression weights	maximum likelihood using COBYLA start: 10^{-2}, bounds: $[10^{-4}, 10^{1}]$

We compare the performances with a standard [2] and surrogate-assisted SMS-EMOA with pre-selection as proposed by [4], since we are not aware of any other SA-EMOAs using the SMS-EMOA with individual-based surrogate management strategies. Specifically, we look at the following algorithms:

SMS-EMOA Standard SMS-EMOA as baseline comparison.

SA-SMS-p Surrogate assisted SMS-EMOA using $\preceq_p$ for pre-selection.

SA-SMS-o Surrogate assisted SMS-EMOA using $\preceq_o$ instead (experimentally shown to improve the performance of pre-selection for the NSGA-II [4]).

SAPEO-uf-ho SAPEO using $\preceq_u$ to rank the offspring, thus accepting a risk of sorting errors of only α^2 (cf. 3.1). For as long as the population cannot be distinguished by $\preceq_u$, the invididuals are evaluated according to $\preceq_f$, thus avoiding making any further sorting errors. The hypervolume relation $\leq_{ho}$ is used as secondary criterion. This algorithm should therefore only take small risks and behave like the SMS-EMOA while saving function evaluations.

SAPEO-ucp-ho SAPEO using increasingly risky relations $\preceq_u$, $\preceq_c$, $\preceq_p$ to avoid evaluations completely if not forced by the uncertainty threshold ε, taking the opposite approach as SAPEO-uf-ho. $\leq_{ho}$ is used as secondary criterion.

SAPEO-uc-hc SAPEO using multi-objectification of the confidence interval boundaries fully. It uses $\preceq_u$ as a first safer way of ranking, followed by $\preceq_c$ on critical individuals. Secondary criterion is $\leq_{hc}$.

[4] In a real-world application, the sample size should be chosen considering the tradeoff between computation times for the model and the fitness function.

4.2 Visualisation and Interpretation of Results

There are two main angles to evaluating the anytime performance of algorithms: (1) measuring the performance indicator after a predefined number of function evaluations (*fixed budget*) and (2) recording the function evaluation when target performances are reached (*fixed target*) [6]. In the following, we use the latter.

For a detailed depiction of an algorithm's performance for a fixed target, we use heatmaps (cf. figure 2) that show the percentage of budget used per dimension until a target was reached according to the colour scale on the right. If the target is not reached within the allocated budget, the corresponding square is white. The dimensions and instances of each function are shown separately to enable analysis of the generalisation of algorithm performance across function instances and dimensions. This is very important to justify the aggregation of performance measures across instances. The functions have colour codes according to the legend in figure 1 that specify their function groups.

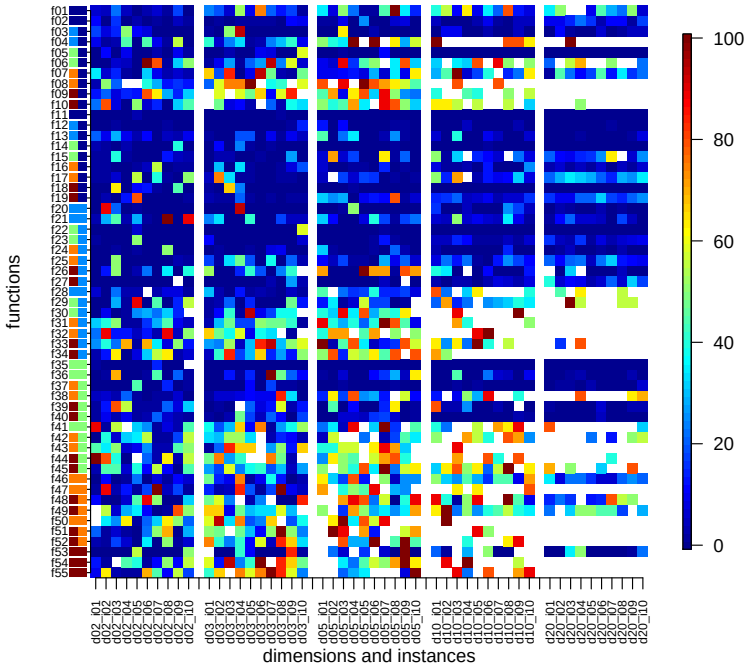

Fig. 2: SAPEO-uf-ho performance in terms of the percentage of the budget per dimension used to reach target 10^0 for all function instances and dimensions.

From the plot, it is apparent that for the selected target 10^0, the algorithm SAPEO-uf-ho has trouble with a number of functions even in small dimensions. Additionally, for those functions, the algorithm's performance seems to drop with increasing dimension of the search space. Especially the Rosenbrock function seems to be problematic for the algorithm: SAPEO-uf-ho rarely reaches the target for dimensions 10 or 20 when the Rosenbrock function is part of the problem (f 04, 13, 21, 28-34). A potential cause is an inaccurate representation

of its narrow valley containing the optimum with the surrogate model. Another explanation could be a mismatch of the Rosenbrock function and the variation operators, causing difficulty for the underlying SMS-EMOA. As expected, some of the weakly structured problems were difficult for SAPEO-uf-ho as well.

A discussion of the potential causes of the performances is only possible with reference to other algorithms. In order to get a better overview of the performances of all algorithms and to detect patterns, we have compiled figure 3, which is an assembly of 30 heatmaps like the one in figure 2 for all algorithms and different targets. The same colour scale as in figure 2 is used. Recall that white spaces signify targets that were not reached within the allocated budget.

In figure 3, the most obvious trend is the declining performance for each target, which was of course expected. It is also apparent that the SMS-EMOA performs better in general than all other algorithms for each target precision. We

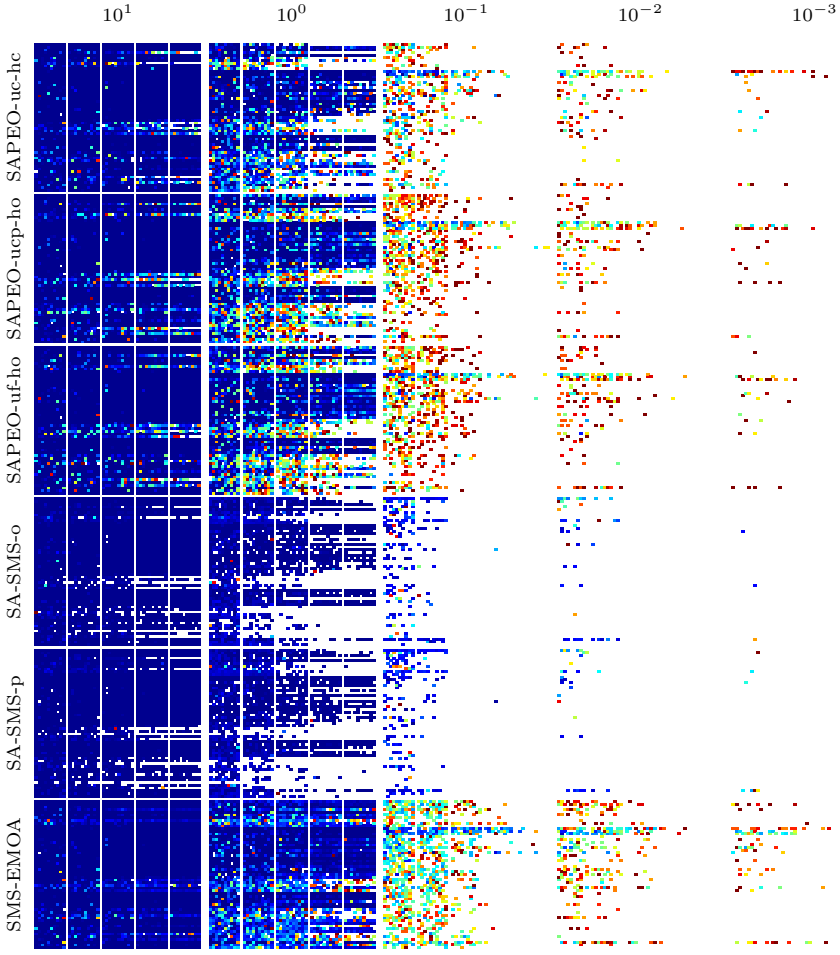

Fig. 3: Heatmaps visualising target performances for all algorithms (rows) across multiple targets (columns). Refer to figure 2 for a detailed explanation.

can also see that all SAPEO versions are an improvement when compared to the SA-SMS algorithms. Interestingly, we also see similar patterns in terms of which functions are more difficult for all algorithms, indicating that the added surrogate models do not influence the underlying optimisation behaviour significantly.

Unfortunately, while providing a good overview, figure 3 is not well suited to interpret the performance of each algorithm per function. While very detailed, the plots are not easy to interpret due to the abundance of information displayed at once. In order to analyse the circumstances of different performance patterns, we compile a plot that aggregates the different instances of a function. This way, the general performance of an algorithm per function can be expressed without risk of overfitting, as intended by the COCO framework. To do that, we use the *expected runtime* (expected number of function evaluations) to reach a target [5] as a performance measure. The measure is estimated for a restart algorithm with 1000 samples. The results are again displayed in a heatmap (figure 4). The colour visualises the estimated expected runtime per dimension according to the scale on the right in log10-scale. Higher values than the maximum budget (> 3) occur if a target is not reached in all instances. White spaces occur if the target was never reached by the algorithm in all instances.

The plot displays expected runtime for all dimensions in different columns according to the labels above. Each of these columns is again divided into 3, displaying the results for different algorithms according to the labels on the bottom. There are two algorithms per column, whose results are displayed on top of each other for each row corresponding to a function. For each algorithm, the expected runtimes for targets $10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}$ are depicted in that order. In case a target was never reached for all algorithms in a column, it is omitted. For example, the expected runtime for SAPEO-uf-ho on function 01, target 10^1 and dimension 2 is on the top left corner and encoded in a light blue. Therefore, the expected runtime to reach target 10^1 is around $10^0 * 2 = 2$. The SMS-EMOA is directly below that and a shade lighter, so has a slightly higher expected runtime. Like in figure 2, the groups each function belongs to are encoded according to the colour scheme in the legend of figure 1.

The general trends as seen in figure 3 can be observed here as well. However, we can also see that SAPEO-uf-ho beats the SMS-EMOA in terms of precision reached on very rare occasions, for example on functions f03 and f41 in dimension 2 and function f20 in dimension 10. Still, the SMS-EMOA generally reaches the same or more precision targets than the other algorithms. However, in most cases where a higher precision target is reached by only a single algorithm, the corresponding colour indicates a very high expected runtime. This means that the algorithm did not reach the higher target for most instances, which speaks against a robust performance of that algorithm. More importantly, the SA-SMS variants often reach less targets than the other algorithms, especially in higher dimensional problems, meaning they are clearly outperformed.

The colour gradients in most functions are remarkably alike, indicating similar behaviour and difficulties experienced with each problem. This is expected, as the intention of SA-EMOAs is to avoid function evaluations with only controlled

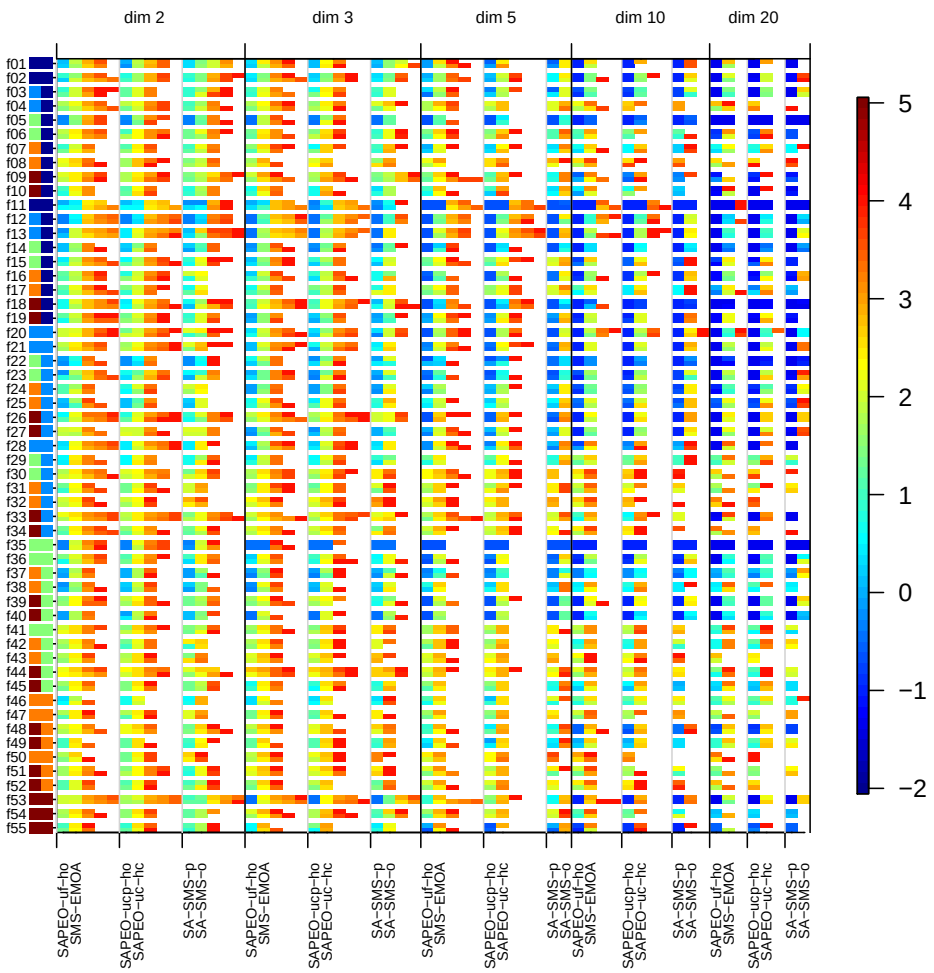

Fig. 4: BBOB-BIOBJ performance results for all algorithms regarding expected runtime (colour coded in log-scale) for targets $10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}$

effects on the evolutionary path. Possibly due to the aggregating nature of the expected runtime measure, the performance contrast does not appear to be as stark as in figure 3. The gradient and number of performance targets reached per function is in fact relatively similar for all algorithms. In most cases, differences occur towards the end of the gradient, indicating that the precision improvement of the surrogate-assisted algorithms is less steep than for the SMS-EMOA. However, in order to analyse the algorithms' behaviour appropriately in that regard, a more thorough analysis of the separate selection steps is required.

Regarding the different functions, there seems to be no clear performance pattern. The different SAPEO versions vary rarely. The performance of all algorithms seems to be more closely tied to the single-objective functions, e.g., Rosenbrock seems to pose problems whereas Schwefel seems more manageable.

5 Conclusions and Future Work

In this paper, we have proposed a novel approach to surrogate-assisted multi-objective evolutionary algorithms called SAPEO. An extensive analysis of its anytime performance using the BBOB-BIOBJ benchmark showed that it was outperformed by its underlying algorithm in this study, the SMS-EMOA. This fact is quite surprising, since the SAPEO-uf-ho variant allows minimal uncertainties and should rarely make different decisions. However, SAPEO still beats another SA-EMOA based on the SMS-EMOA [4] on the benchmark.

One potential source of error is the surrogate model, e.g. assumptions **A1, A2** (section 3.1) could be wrong. A large error in the predicted uncertainty could have a tremendous influence on the algorithm. However, this is controllable through the adaptation of the uncertainty threshold ε. Additionally, the uncertainties during the start of the SAPEOs were relatively large, which could also send the algorithm into a wrong direction. The uncertainties could be mitigated by using surrogate ensembles instead, distributing the samples better, increasing the sample size or selecting a fitting kernel. Additionally, the performances of local vs. global surrogates should be analysed more thoroughly. It is apparent that the quality of the surrogate model is a major concern for SA-EMOAs, which could be problematic for black-box optimisation in general.

Apart from the model, there are possible improvements regarding the binary relations used. For one, $\preceq_u$ could be defined without forcing strict Pareto dominance of the hypercubes. Furthermore, using hypercubes for the potential location of the fitness values is a simplification. Perhaps a binary relation on hyperellipsoids could provide better results.

Furthermore, while the SAPEO approach worked well for single-objective problems, the corresponding multi-objective problems pose an incomparably larger difficulty for a surrogate model. Additionally, even slightly overestimated function values could lead to an incorrect identification of dominated individuals. This is because a large number of critical values would need to be distinguished with less certain relations or expensive function evaluations.

Notice that previous studies [12,1] have shown that EAs with a larger number of offspring are less vulnerable to noisy fitness functions. Therefore, it may be conjectured that the $\mu + 1$ selection scheme of the SMS-EMOA causes the poor performance under noise induced by the surrogate. This hypothesis, however, remains a question for future research. In general, the influence of the quality of surrogate models on SA-EMOAs should be analysed more carefully. With the proper noise-robust optimisation algorithm and parametrisation, SAPEO should be able to beat its underlying algorithm as it does on single-objective problems.

References

1. Arnold, D., Beyer, H.-G.: On the Benefits of Populations for Noisy Optimization. Evolutionary Computation 11(2), 111–127 (2003)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection

based on dominated hypervolume. European Journal of Operations Research 181(3), 1653–1669 (2007)
3. Brockhoff, D., Tušar, T., Tušar, D., Wagner, T., Hansen, N., Auger, A.: Biobjective Performance Assessment with the COCO Platform. CoRR abs/1605.01746 (2016), retrieved: 22/12/2016
4. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single- and Multi-objective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. IEEE Transactions on Evolutionary Computation 10(4), 421–439 (2006)
5. Hansen, N., Auger, A., Ros, R., Finck, S., Pošík, P.: Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In: Companion of Genetic and Evolutionary Computation Conference (GECCO 2010). pp. 1689–1696. ACM Press, New York (2010)
6. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA (2009), retrieved: 22/12/2016
7. Jin, Y.: A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. Soft Computing 9(1), 3–12 (2005)
8. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation 1(2), 61–70 (2011)
9. Knowles, J., Nakayama, H.: Meta-Modeling in Multiobjective Optimization. In: Branke, J., et al. (eds.) Multiobjective Optimization - Interactive and Evolutionary Approaches, pp. 245–284. Springer, Berlin (2008)
10. Limbourg, P., Aponte, D.E.S.: An Optimization Algorithm for Imprecise Multi-Objective Problem Functions. In: IEEE Congress on Evolutionary Computation (CEC 2005). IEEE Press, Piscataway, NJ (2005)
11. Mlakar, M., Petelin, D., Tušar, T., Filipič, B.: GP-DEMO: Differential Evolution for Multiobjective Optimization based on Gaussian Process models. European Journal of Operational Research 243(2), 347–361 (2015)
12. Nissen, V., Propach, J.: Optimization with noisy function evaluations. In: Parallel Problem Solving from Nature (PPSN V). pp. 159–168. Springer, Berlin (1998)
13. Rudolph, G.: A Partial Order Approach to Noisy Fitness Functions. In: IEEE Congress on Evolutionary Computation (CEC 2001). pp. 318–325. IEEE Press, Piscataway, NJ (2001)
14. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science 4(4), 409–423 (1989)
15. Tušar, T., Brockhoff, D., Hansen, N., Auger, A.: COCO: The Bi-objective Black Box Optimization Benchmarking (bbob-biobj) Test Suite. CoRR abs/1604.00359 (2016), retrieved: 22/12/2016
16. Zitzler, E., Knowles, J., Thiele, L.: Quality Assessment of Pareto Set Approximations. In: Branke, J., et al. (eds.) Multiobjective Optimization - Interactive and Evolutionary Approaches, pp. 373–404. Springer, Berlin (2008)