PSA: A Family of Probabilistic Surrogate-Assisted Algorithms for Single-Objective Optimization

COIN Report Number 2021005

Julian Blank
Michigan State University
East Lansing, Michigan
blankjul@msu.edu

Kalyanmoy Deb
Michigan State University
East Lansing, Michigan
kdeb@msu.edu

ABSTRACT

In the last two decades, significant effort has been made to solve computationally expensive optimization problems using surrogate models. Regardless of whether surrogates are the primary drivers of an algorithm or improve the convergence of an existing method, most proposed concepts are rather specific and not very generalizable. Some important considerations are the selection of a baseline optimization algorithm, a suitable surrogate methodology, and the surrogate's involvement in the overall algorithm design. This paper proposes a family of probabilistic surrogate-assisted algorithms (PSA), demonstrating its applicability to a broad category of single-objective optimization algorithms. The concept introduces knowledge from a surrogate into an existing algorithm through a tournament-based procedure and continuing optimization on the surrogate's predictions. The surrogate's involvement is determined by updating a replacement probability based on the accuracy from past iterations. A study of four well-known population-based optimization algorithms with and without the proposed probabilistic surrogate-assistance indicates its usefulness in achieving a better convergence. The concept should provide a generic and comprehensive use of surrogates within an optimization algorithm and pave the way for new surrogate-assisted algorithms dealing with challenges in less frequently addressed computationally expensive functions, such as different variable types, large dimensional problems, multiple objectives, and constraints.

CCS CONCEPTS

• **Theory of computation** → **Nonconvex optimization**; **Bio-inspired optimization**;

KEYWORDS

Surrogate-Assisted Optimization, Metamodel-based Optimization, Simulation Optimization, Evolutionary Computing, Genetic Algorithms

1 INTRODUCTION

Many optimization problems in practice are computationally expensive and time-consuming. On the one hand, the computational expense for evaluating objective or constraint functions requires an algorithm to be more careful in selecting new solutions to be evaluated. On the other hand, it justifies significant more algorithmic overhead to find new solutions. Thus, most research focuses on developing methods that use the additional computational time to return well-suited solutions. One widely-used technique utilizes so-called surrogates [22] (or metamodels) to approximate the search space's objective and constraint functions. The essential question is how one or even multiple surrogate models shall be incorporated into an algorithm's design and how the surrogate's prediction error is addressed [8]. This includes defining the surrogate's involvement and impact, which is crucial for all surrogate-based algorithms. Existing methods can be put into one of the following categories depending on the surrogate's involvement: influenced, biased, centered, or non-adaptive (see Figure 1). In the early phase of surrogate-based optimization, the surrogate was fitted *only once* and optimized. Thus, the algorithm's performance entirely depended on the accuracy of the surrogate model. In 1998, an adaptive surrogate-based approach called EGO (Efficient Global Optimization) was proposed in a seminal publication by Jones et al. [15]. The idea is based on finding a so-called infill solution in each iteration using an optimization problem based on the surrogate's prediction and uncertainty. This adaptive surrogate-based method was the origin of all kinds of extensions, for instance new infill criteria dealing with the surrogate's uncertainty differently or adding the capability to optimize more than one objective at a time. The original EGO and all its extensions have in common that the algorithm design is *centered* on the surrogate model and, thus, the surrogate's accuracy having a significant impact on the performance. Also, for such a surrogate-centered approach, all intelligence needs to be put into the "fit-and-optimize" procedure, making it challenging to design an algorithm that benefits from decades of research on optimizing more computationally inexpensive functions.

In contrast to making the surrogate the primary driver of an algorithm, an existing optimization method could be extended to utilize surrogate predictions. Researchers refer to such approaches as *surrogate-assisted* algorithms, emphasizing the surrogate's role as an assistant of an already existing optimization algorithm. The advantage of such a method is the capability of benefiting from features of the underlying algorithm directly. In our schema, surrogate-assisted algorithms are split up into two categories. On the one hand, algorithms being *influenced* by a surrogate playing only a minor role; on the other hand, surrogate-*biased* methods where the algorithm has a bigger impact on the algorithm's design. Because the judgment of impact is somewhat subjective, the transitions between influenced and biased are fluid.

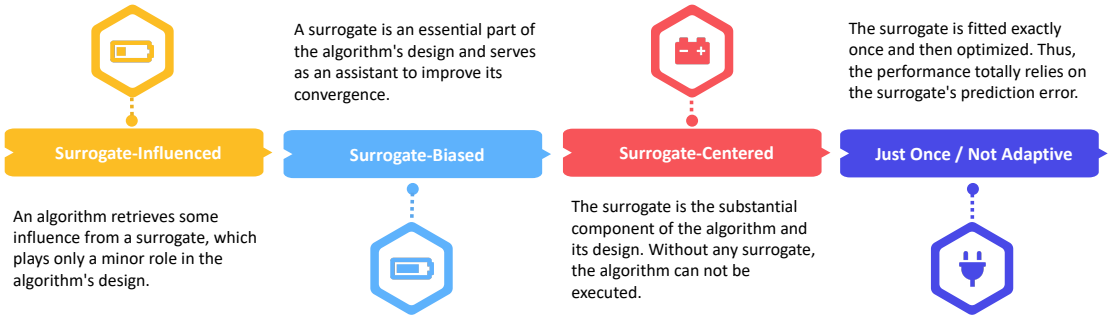

Figure 1: Different Roles of Surrogates in the Design of an Algorithm.

This paper's primary contribution is the proposal of a probabilistic surrogate-assisted (PSA) platform to use surrogates in an optimization algorithm in a generic manner. The underlying algorithm only needs to fulfill minimum requirements, making this a broadly applicable concept to various methods. Such a general concept shall provide a suitable alternative to the widely-used EGO approach. PSA does not require the surrogate model to return an uncertainty measure but keeps itself track of the surrogate's accuracy. Thus, it is in contrast to other surrogate-based algorithms not limited to a specific type of surrogate model. Another question frequently raised is a good number of the initial design of experiments, which becomes a less critical parameter with an underlying baseline algorithm inducing exploration naturally. Moreover, the proposed method introduces only three intuitively explainable hyper-parameters discussed in detail. Altogether, this study addresses various essential research questions of surrogate-assisted optimization raised over the last years.

In the remainder of this paper, we first discuss related work of surrogate-assisted optimization and its challenges. In Section 3, we propose the probabilistic surrogate-assisted concept and its components. A comparison of the proposed algorithm family with their underlying baseline and other state-of-the-art surrogate-based methods is provided in Section 4. Finally, conclusions are drawn in Section 5.

2 RELATED WORK

While many surrogate-based algorithms have been proposed for a specific problem or a problem class, only a few studies focused on more general approaches.

In 1998, Jones et. al. has proposed a methodology for efficient global optimization, which has significantly influenced the research direction of surrogate-based optimization [15]. In his seminal work, Kriging [18] is used as a surrogate, which provides, besides predictions, a measure of uncertainty. The prediction and uncertainty together define the so-called acquisition functions (or infill criteria), such as the expected improvement [15] or probability of improvement [14]. One critical requirement for acquisition functions is to balance exploitation and exploration. The optimization of the acquisition function results in an infill solution, which is first evaluated and then added to the model. The procedure is repeated until a termination criterion is met. The limitation of finding only a single new solution in each iteration has been investigated thoroughly, and multi-point EGO approaches [3, 4, 25] have been proposed. Moreover, the concept has been generalized to solve multi-objective optimization problems by using decomposition [17] or replacing the objective with a performance indicator based metric [21]. The idea has also been extended to handle constraints, which is especially important for solving real-world optimization problems [2]. Because much effort has been made to extend the initially proposed concept, this can be considered a rather general approach. However, because of the large number of variants addressing one or multiple initial limitations, it can be challenging to find a suitable implementation when facing real-world optimization problems.

In 2007, Lim et al. proposed a generalized framework that unifies diverse surrogate models synergistically in a genetic algorithm's local search [19]. The algorithm aims to benefit from the "bless of uncertainty" by using ensemble and landscape smoothing surrogate models and replaces solutions using Lamarckian learning. The generalization is based on proposing an algorithm that solves real-world single and multi-objective optimization problems robustly. Results on complex problems with a multi-modal fitness landscape indicate that this general approach shows competitive performance to other surrogate-assisted algorithms.

In 2019, Cai et al. [6] proposed a concept incorporating a surrogate-based pre-screening and local search strategy. The surrogate's approximation error is addressed by a trust-region method. Besides a surrogate-based local search, the evolutionary algorithm's mating is biased by replacing one parent with the best performing (based on surrogate predictions) solution in a randomly chosen solution's neighborhood. The proposed method showed promising results on a variety of low and high-dimensional test problems. The applicability of the concept to all kinds of algorithms based on evolution with local searches makes this a generalizable concept.

Most existing studies focus on generalizing surrogate-assisted optimization by proposing one concept that suits different kinds of optimization problems. However, mostly *one* specific type of algorithm is chosen, and surrogate-bias is added by modifying the underlying operations or adding a local search. A surrogate-assisted concept being only applicable to one specific algorithm makes it challenging to benefit from research on optimizing computationally inexpensive functions. Thus, researchers have noted that quite common challenges, for instance, handling multiple objectives at a time or the existence of constraints, need to be investigated in more detail for computationally expensive problems [23].

3 METHODOLOGY

The overall outline of the algorithm is shown in Algorithm 1. The PSA concept requires a baseline algorithm A which implements two methods – `infill()` and `advance(X,F)`. For instance, for genetic algorithms, this corresponds to the mating and environmental survival, or in particle swarm optimization to applying the swarm equations and replacing the personal if a better solution has been found. Moreover, three hyper-parameters, α, β, and $\rho^{(\text{max})}$ are passed to balance the surrogate's influence on the baseline algorithm. First, the design of experiments $X^{(\text{doe})}$ are generated in a space-filling manner and evaluated $F^{(\text{doe})}$ using the time-consuming evaluation function (Line 1). An iterative procedure introducing surrogate bias to the baseline algorithm continues until the user-defined termination criterion is met. Each iteration begins with asking the baseline algorithm for new infill solutions. All steps from there on until the evaluation of X and the advancement of the algorithm (Line 28 and 29) aim to introduce surrogate bias. The surrogate bias consists of two phases: the α-phase with a light influence using a tournament selection based on surrogate predictions (Line 6 to 12); and the β-phase simulating the algorithm for a number of generations on the surrogate and accepting solutions with the probability ρ derived from the surrogate's accuracy (Line 13 to 27).

3.1 Influence of Surrogate through Tournament Selection Pressure (α)

A well-known concept in evolutionary computation to introduce a bias towards more promising solutions is *tournament selection*. An individual from the population has to win a tournament to be selected to contribute to the mating process. The number of competitors (α) balances how greedy the selection procedure shall be. On the one hand, a larger value of α allows only elitist solutions to participate in mating, while a smaller value introduces a light selection pressure. For genetic algorithms, the most frequently used tournament mode is the binary tournament ($\alpha = 2$), which compares a pair of solutions regarding one or multiple metrics. For example, a standard binary tournament implementation for constrained single-objective optimization declares the less infeasible solution as the winner if one or both solutions are infeasible or otherwise returns the solution with the smaller function value.

The tournament concept is made use of to introduce a surrogate bias while creating new infill solutions. Whereas in genetic algorithms already evaluated solutions are compared, no exact information about the individual's fitness yet exists in PSA at the time of comparison. Thus, the surrogate serves as a referee in a tournament by providing approximation values for each individual. In Figure 2 surrogate-assisted tournament selection with three competitors ($\alpha = 3$) for four infill solutions is shown. Initially, the algorithm's `infill` function has been called three times to generate X^{α_1}, X^{α_2}, and X^{α_3}. Then, a tournament takes place where the i-th solutions of the j-th infill solution set $X_i^{\alpha_j}$ compete with each other. For instance, for the first tournament the winner of $X_1^{\alpha_1}$, $X_2^{\alpha_1}$, and $X_3^{\alpha_1}$ has to be declared. As a comparison function, the surrogate's approximation function $\hat{F}_i^{\alpha_j}$ is used. In general, setting $\alpha = 1$ disables the tournament selection and serves as a fallback. By involving the surrogate in the tournament selection ($\alpha > 1$), the infill solutions X get a smaller or larger influence based on the number of competitors.

Algorithm 1: PSA: A family of **P**robablistic **S**urrogate-**A**ssisted Algorithms

Input: Algorithm A with `infill()` and `advance(X, F)`
Surrogate Tournament Pressure α (≥ 1)
Number of Simulated Iterations β (≥ 0),
Maximum Surrogate-Bias $\rho^{(\text{max})}$ (≥ 0.0)

```
/* Sample Design of Experiments (DOE)                          */
1  X^(doe) ← doe(); F^(doe) ← evaluate(X^(doe))
2  while has_terminated() do
       /* Default infill solutions from baseline algorithm     */
3      X ← A.infill()
       /* Fit a surrogate and predict values for infills       */
4      S ← fit(X^(doe), F^(doe))
5      F̂ ← S.predict(X)
       /* Surrogate-assisted Tournament Pressure (α)           */
6      foreach k ← 2 to α do
7          X^αk ← A.infill()
8          F̂^αk ← S.predict(X^αk)
9          foreach j ← 1 to size(F^αk) do
10             if F̂_j^αk < F̂_j then X_j ← X_j^αk; F̂_j ← F̂_j^αk ;
11         end
12     end
       /* Bias by Continuing the Algorithm on Surrogate (β)    */
13     A' ← copy(A)
14     X^β ← X; F̂^β ← F̂
15     foreach k ← 1 to β do
16         X^βk ← A'.infill()
17         F̂^βk ← S.predict(X^βk)
18         foreach j ← 1 to size(F̂^βk) do
19             i ← closest(X_j^βk, X^β)
20             if F̂_j^βk < F̂_i^β then X_i^β ← X_j^βk; F̂_i^β ← F̂_j^βk ;
21         end
22         A'.advance(X^βk, F^βk)
23     end
24     ρ ← min(estm_surr_bias(S), ρ^(max))
25     foreach j ← 1 to size(X^β) do
26         if random() < ρ then X_j ← X_j^β ;
27     end
       /* Next iteration of the overall algorithm              */
28     F ← evaluate(X)
29     A.advance(X, F)
30     X^(doe) ← X^(doe) ∪ X
31     F^(doe) ← F^(doe) ∪ F
32 end
```

3.2 Continue Optimization on Surrogate (β)

Whereas the tournament is an effective concept to incorporate the surrogate's approximation, it is limited by looking only a *single* iteration into the future. To further increase the surrogate's impact, the baseline algorithm is continued to run for β more consecutive iterations on the surrogate's approximations. Inevitably, the

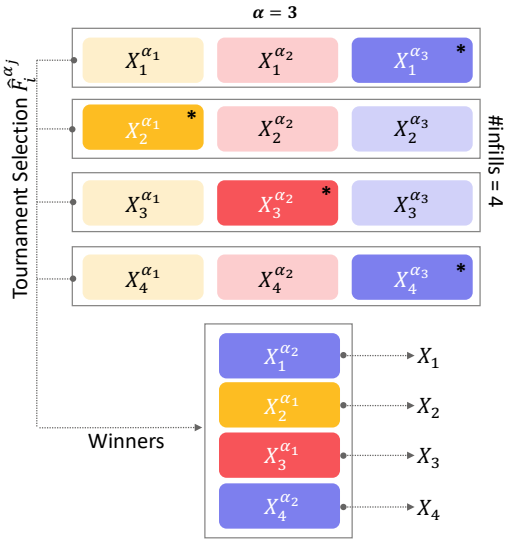

Figure 2: Tournament selection with α competitors to create a surrogate-influenced infill solutions.

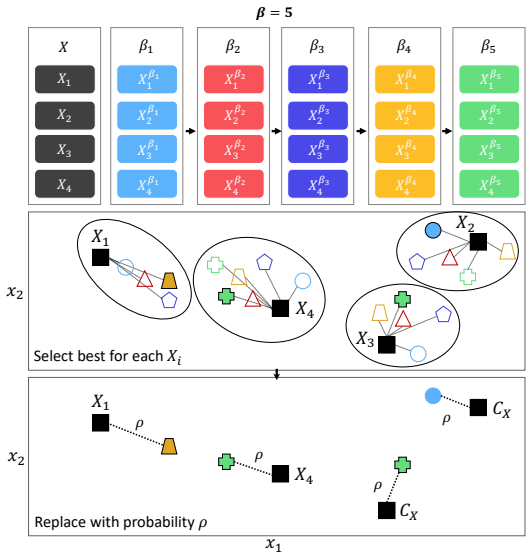

Figure 3: Continue Running the Algorithm for β Iteration on the Surrogate.

question of how many iterations are suitable arises and indicates the importance of tuning β. Nevertheless, even more critical, how shall the algorithm profit from simulating the algorithm on the surrogate? An inappropriate choice of β will cause the surrogate's optimum repeatedly to be found and discarding the baseline algorithm's default infill procedure entirely. Automatically, this also causes a diversity loss of infill solutions and does not account for the surrogate's approximation error. Thus, we propose a probabilistic surrogate-assisted approach that balances the surrogate's impact on the baseline algorithm to address these issues.

The probabilistic procedure is described in Algorithm 1 from Line 13 to 27. Because the iterations are only simulated on the surrogate, the original algorithm object must be copied to avoid any modifications of the current algorithm's state (Line 13). Then, the algorithm is continued on the surrogate model for β iterations, by calling in each iteration k the `infill` method returning X^{β_k} and feeding back to the algorithm the approximations $\hat{F}^{\beta_k}$ by calling `advance` (Line 16, 17 and 22). The goal of these iterations is to introduce more surrogate-bias into X. Therefore, a surrogate-biased population X^β is obtained by initializing $X^\beta = X$ and $\hat{F}^\beta = \hat{F}$ (Line 14) and assigning in each iteration (k) every solution $X_j^{\beta_k}$ to its closest solution i in X^β. The closest solution is determined based on the smallest (normalized) Euclidean distance in the design space. The infill X_i^β and the corresponding prediction $\hat{F}_i^\beta$ is replaced if the newly found solution performs better considering the surrogate's prediction. Finally, a biased candidate solution X_j^β replaces X_j with probability ρ bounded by $\rho^{(\max)}$. Clearly, the value of ρ determines the impact of the β-phase on the baseline algorithm.

An exemplary with five iterations ($\beta = 5$) and four infill solutions X_1, X_2, X_3, and X_4 is also illustrated in Figure 3. Calling the infill function of the baseline algorithm results in five solution sets with four solutions each. When running the algorithm, the assignment takes place, and for instance, X_1 has four solutions being the closest to, and X_4 has six. The assignment of the closest solution will show cluster like arrangements and preserve diversity.

In general, optimizing the surrogate model is a common technique used in surrogate-assisted algorithms. However, a crucial aspect is addressing the utilization of knowledge from these iterations. An assignment-based and probabilistic approach keeps the balance between the default algorithm's behavior and surrogate bias. The strategy to determine the "bottleneck" variable ρ of the β-phase is described next.

3.3 Balancing the Utilization of Surrogate (ρ)

The number of infill solutions being finally biased by β iterations on the surrogate is critical and balances the whole surrogate's involvement. In industry projects finding a suitable surrogate can be rather challenging and is often done manually. Comparing different types of surrogates and selecting the most suitable one is usually based on a metric which judges a model's trustworthiness. A well-known metric to estimate the accuracy is the coefficient of determination (also known as R^2):

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{f}_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{\mathtt{MSE}(y, \hat{f})}{\mathtt{MSE}(y, \hat{f}^{\bar{y}})} \tag{1}$$

where y_i represents the output and $\hat{f}_i$ the prediction of the i-th value, and $\bar{y}$ the arithmetic mean of all output values. The denominator $\sum_i (y_i - \bar{y})^2 = \mathtt{MSE}(y, \hat{f}^{\bar{y}})$ represents the Mean Squared Error (MSE) of a surrogate $\hat{f}^{\bar{y}}$ always predicting the average of all output values. This error serves as the normalization constant for coefficient of determination. For a surrogate performing worse than $\hat{f}^{\bar{y}}$, the right-hand side of the equation results in a value greater than 1 and, thus, R^2 becomes negative. If the surrogate performs equally good the value is zero and otherwise positive. The upper bound of the R^2 metrics upper bound is 1, which could theoretically be reached with

an MSE of zero. These characteristics turn out to be very suitable for defining a probability and thus have been the inspiration for balancing the surrogate-bias.

The replacement probability ρ is given by bounding R^2 on the lower end to zero:

$$\rho = \max\left(0,\ 1 - \frac{\mathsf{MSE}(y, \hat{f})}{\mathsf{MSE}(y, \hat{f}^b)}\right) \tag{2}$$

where $\hat{f}^b$ represents a *baseline* predictor. The formula of ρ generalizes the definition of R^2 by considering an arbitrary baseline predictor $\hat{f}^b$ instead of $\hat{f}^{\bar{y}}$. Given that R^2 has an upper bound of one and it is at least zero, $\rho \in (0, 1)$ holds and thus is a valid probability.

But what does ρ as a metric defining the surrogate-bias imply in the context of an algorithm's iteration? If the surrogate performs worse than $\hat{f}^b$ ($\rho = 0$), no solution will be replaced. On the opposite, if the surrogate has no prediction error, all solutions will be surrogate-biased. Even more importantly, if none of these two extreme cases occur, the value of ρ and therefore the surrogate-bias will be adjusted proportional to the accuracy of the model normalized by the performance of $\hat{f}^b$. In our implementation, we chose a k-nearest neighbor model ($k = n+1$ where n represents the number of variables) as a baseline predictor $\hat{f}^b$ and average ρ over the last five iterations (sliding window). For estimating the model's accuracy, we use all data points observed until the last generation as training and newly evaluated infill solutions as a validation set. In the initial iteration where only the design of experiments and no infill solutions exist, a k-fold cross-validation is performed. Our experiments have shown an assessment of the surrogate prediction error is essential and, thus, directly incorporating it into the algorithm design recommended.

3.4 Surrogate Management

The algorithm's outline has already shown that the surrogate model has to be fitted through data points and is used as a predictor for new infill solutions. In general, all matters related to fitting or updating a surrogate is referred to as surrogate management. It is noteworthy that in practice, not only a single but multiple surrogates are recommended to provide a more robust model with less approximation error. With multiple surrogates, we refer not only to the type of surrogate but also concrete hyper-parameters. In our implementation, in total 15 surrogates, consisting of the model types RBF [13] and Kriging [18], are validated. The hyper-parameters instantiate models with different mean functions, kernel, and noise. Finally, the model with the highest ρ value is chosen. On the one hand, an increasing number of points from optimization increase the time spent for surrogate management and, on the other hand, can lead to precision issues. The precision issues are caused by solutions by very close to each other in the design space. Thus, we employ an ϵ-clearing approach, which always selects the solution with the smallest function value and then clears all solutions with less than ϵ distance to it ($\epsilon = 0.005$). We reduce the overall amount of points by only considering the 200 best solutions from the ϵ-cleared solution set.

4 RESULTS

This study focuses on computationally expensive functions, which shall be imitated by considering a very limited evaluation budget for test problems. This is a commonly used principle in surrogate-assisted research, especially for more general approaches that need to be tested on several optimization problems. In this study, we have limited the function evaluations to 200-300 and considered problems with up to 10 variables. For comparison, we use well-known test problems, such as Sphere, Ackley, Rosenbrock, and others from the single-objective BBOB test suite [11]. To demonstrate the generalizability of PSA, we (i) conduct an experiment focusing on the most suitable hyper-parameter combination, (ii) investigate the impact of dynamically determining the surrogate-bias by ρ and (iii), lastly, compare the proposed family of surrogate-assisted algorithms with other recently proposed methods for computationally expensive problems.

4.1 What are suitable values for α, β and ρ?

In our first experiment, ρ is kept fixed and not updated. This shall give insights if ρ is a problem-dependent variable and, in fact, benefits from being updated based on the surrogate's prediction error. Moreover, the impact of α and β on an algorithm's performance is of interest.

For this hyper-parameter study, we chose CMA-ES [12] as a baseline algorithm. We normalize each variable between zero and one to avoid any scaling irregularities and initialized the algorithm with a standard deviation of $\sigma = 0.15$. The algorithm's initial starting point is determined by the best solution found by generating 20 points using Latin Hypercube Sampling [20]. We employ grid-based optimization by setting the hyper-parameters $\alpha \in (1, 2, 3, 5, 10)$, $\beta \in (0, 5, 10, 20, 30, 40, 50)$ and $\rho \in (0.1, 0.2, \ldots, 0.9, 1.0)$ for PSA-CMA-ES . Because $\beta = 0$ makes the value of ρ irrelevant, there is not need to consider any run with $\rho = 0$ in addition. Because of the stochastic nature of the algorithm, we execute each parameter combination 11 times. This resulted in 60,588 runs in total for all test problems. As a performance criterion, we address the so-called anytime performance $f^{(\mathsf{any})}$ of the algorithm and calculate the *integral* of the convergence curve based on the gap to the optimum $f^{(\mathsf{gap})} = f - f^{(\mathsf{opt})}$. Measuring the convergence and not only the final function value addresses the desire of a surrogate-assisted algorithm converging as quickly as possible with a very limited function evaluation budget.

In Figure 4 results of the hyper-parameter experiment for three exemplary optimization problems are shown in the form of a Parallel Coordinate Plot [26] are shown. The first three vertical lines represent the parameters α, β and ρ, and the last the performance metric $f^{(\mathsf{any})}$, relative to the baseline algorithm CMA-ES. Thus, the baseline algorithm's performance (blue) always ends up being 1.0, and the resulting values indicate the proportional improvement/deterioration. Moreover, the best performing parameter combination (red) and the second to tenth best (yellow) are highlighted.

(i) One can observe that adding surrogate-bias has successfully improved the performance of the baseline algorithm. For almost all parameter combinations, the PSA variant achieved values less than one and improved the baseline algorithm's performance. Moreover,

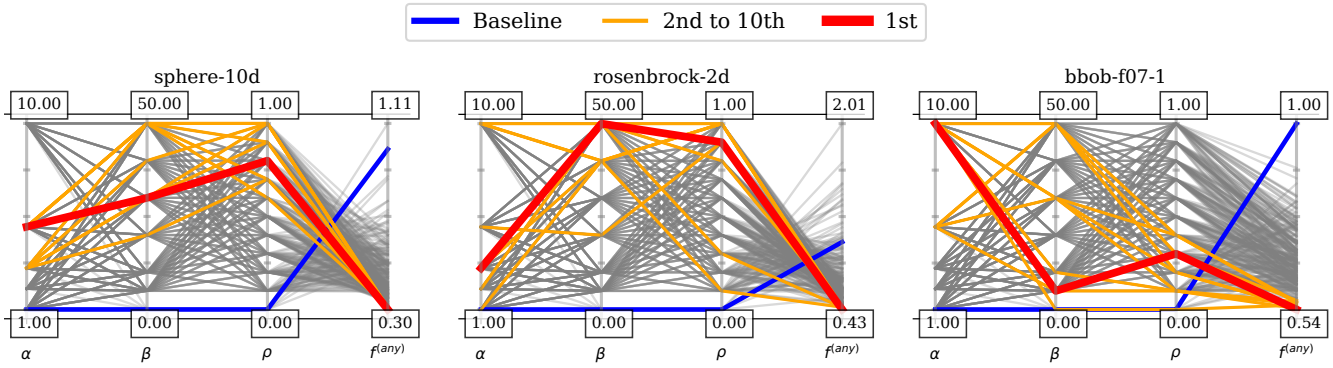

Figure 4: Hyper-parameter Analysis for PSA-CMA-ES with varying α, β and ρ. Shown are the baseline algorithm CMA-ES (blue), the 2nd to 10th best (orange) and the best (red). The performance $f^{(\text{any})}$ is normalized with respect to the baseline algorithm.

for the most suitable hyper-parameter values, PSA showed a remarkable improvement by reducing the convergence integral to 30%.

(ii) One might think that already introducing a strong bias in the β-phase makes the α-phase irrelevant. However, results indicate that it is beneficial for PSA to have pre-filtering. The fundamental difference between these two phases is that α is applied no matter how good the surrogate performs but only gives some light influence in the form of a tournament. Nevertheless, more experiments need to be conducted to determine a clear winner for a value of α across all problems.

(iii) It becomes evident that for `sphere` and for `rosenbrock` rather large values of β and ρ and thus a stronger surrogate-bias are a better choice. In contrast to `bbob-f07-1` where less surrogate involvement has turned out to be more effective. Moreover, even for a relatively simple 10-dimensional quadratic function, the surrogate shall not be used 100% ($\rho = 1.0$) of the time. Analysis has shown this can be attributed to the limited number of points initially. Even for problems with almost no complexity, a small initial number of design of experiments (here 20) requires the baseline algorithm to do some exploration until the surrogate starts to recognize the characteristics of the function and has a suitable accuracy.

(iv) Besides visualization, we have also performed a ranking based analysis to find suitable parameter combinations. Thus, we have averaged the ranking in *percentage* across all problems. For instance, rank 30 out of 307 results in a value of ≈ 0.0977. The average percentage ranks with their standard deviations are shown in Table 1. Results indicate that an α value between 5 to 10, a β value between 30 to 50 and a value of ρ between 0.3 to 0.5 perform best.

Table 1: Rankings of best performing hyper-parameters.

α	β	ρ	rank in %	std of rank in %
10	**40**	**0.4**	**0.2485**	**0.1556**
5	40	0.5	0.2485	0.1949
10	50	0.3	0.2586	0.1664
10	30	0.3	0.2595	0.1914
5	40	0.3	0.2606	0.1728

4.2 Is it beneficial to update ρ each iteration?

Our next study addresses the impact of updating ρ each iteration. The relatively small ρ values found to perform best might indicate that trusting the surrogate too much slows down the overall convergence. Thus, the effect of updating ρ in each iteration based on the surrogate's prediction error shall be investigated (Algorithm 1 and Section 3.3). Considering the insights gained from the hyper-parameter study, we define an upper bound for ρ determining the maximum influence of the β-phase to a reasonable value of $\rho^{(\text{max})} = 0.7$. Moreover, we have used good performing parameter combinations from the previous hyper-parameter study. The experiment reveals that an update of ρ performs significantly better than the best parameter combination from before and is, thus, recommended (see Table 2).

Table 2: Ranking with adaptive ρ.

α	β	ρ	rank in %	std of rank in %
10	**40**	**adaptive**	**0.1375**	**0.1831**
10	40	0.4	0.2485	0.1556

4.3 How does PSA perform compared to other surrogate-based algorithms?

For the remainder of this study, we fix the hyper-parameters to a $\alpha = 10$, $\beta = 40$ and perform a dynamic update of ρ with $\rho^{(\text{max})} = 0.7$. So far, our experiments have been based on CMA-ES to avoid an immense amount of runs for drawing conclusions about suitable hyper-parameters. However, for a comparison with other methods we have applied PSA to the following well-known population-based algorithms besides CMA-ES [12]: Differential Evolution (DE) [24], Particle Swarm Optimization [16] with adaptive c_1 and c_2 [27] and a standard genetic algorithm [9]. For all algorithms, the population size and number of infills solutions (or, depending on the algorithm, called particles or offsprings) has been set to 10. The algorithm implementations available in the optimization framework *pymoo* [5] have been used.

First, we like to confirm that PSA improves the convergence of the considered baseline algorithms on various test problems.

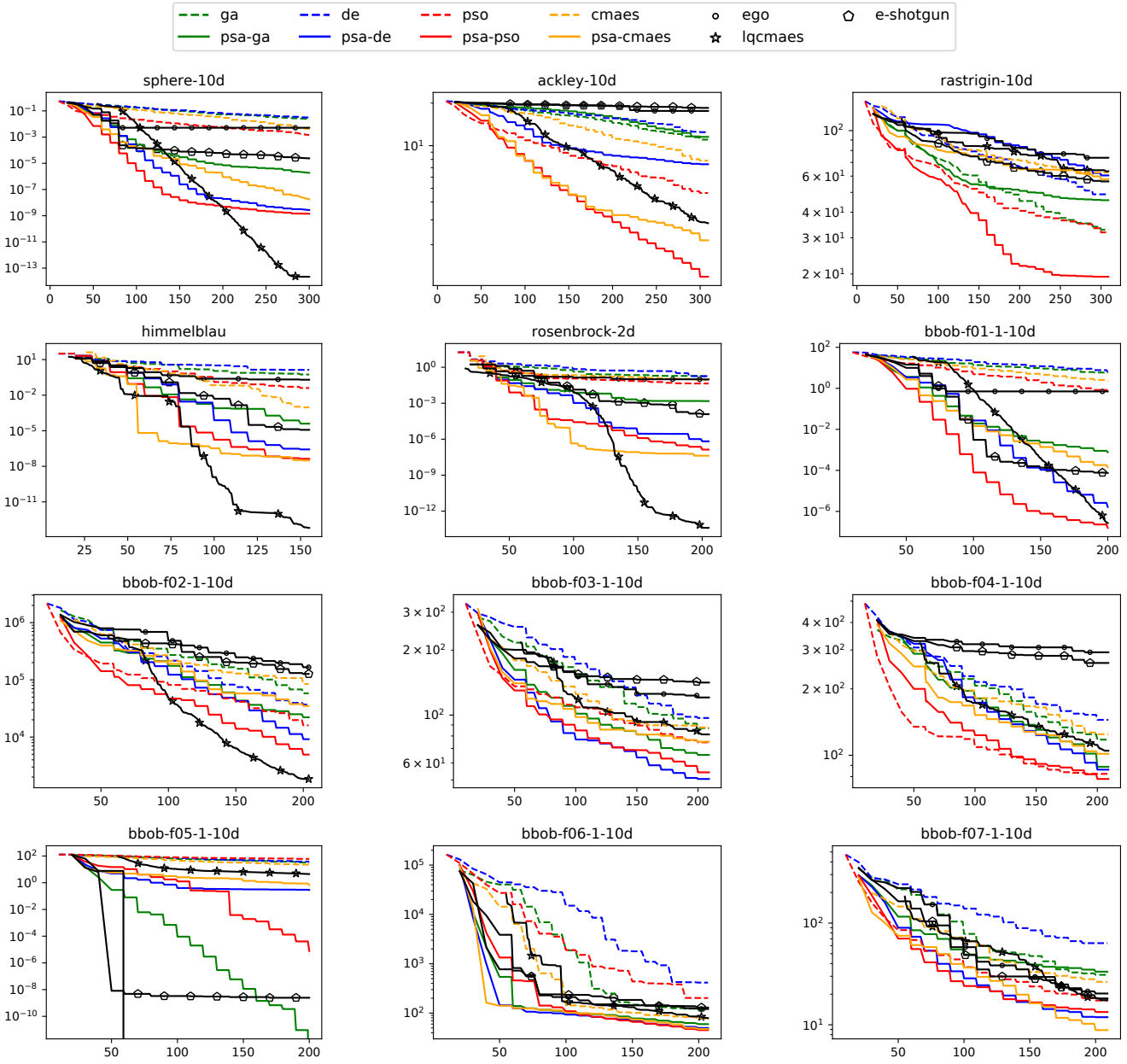

Figure 5: A convergence-based comparison of PSA with their baseline methods and other surrogate-based algorithms.

Figure 5 shows the convergence plots (averaged over 11 runs) of a variety of single-objective optimization problems. The PSA variants are plotted with straight and the baseline algorithms with dashed lines. The convergence curves demonstrate the superiority of surrogate-assisted approaches across all test problems except for `bbob-f04-1-10d`. We attribute the superiority of PSO to the problem complexity and the fact that no algorithm can convergence with the limited evaluation budget of 300.

Second, the performance compared to other surrogate-based algorithms shall be demonstrated. As a comparison we have chosen a standard EGO implementation from GPyOpt [1] (with 10 infill points in each iteration), a recently proposed method called ϵ-shotgun [7] (with a batch size of 10 and $\epsilon = 0.1$), and lqCMAES [10]. First, one can observe that lqCMA-ES, based on a quadratic model approximation, converges closer to the optimum *if* a solution near the optimum is found. Nevertheless, for problems where this is not the case, PSA variants show superior performance. Extending PSA

to perform a local search using a quadratic model might show similar convergence behavior near an optimum. Moreover, EGO and ϵ-shotgun are outperformed by almost all PSA variants except for `bbob-f05-1-10d` where a solution close to the optimum is found right away. Comparing algorithms of the PSA family with itself, does allow to declare no clear winner. Whereas PSA-PSO seems to perform well for most problems, PSA-CMA-ES converges faster for the problems with only two variables. Altogether, considering an algorithm with a gap to the optimum of less than 10^{-6} as converged, at least one PSA variant was better 50% (6/12) and equally good 30% (4/12) of the time. This can be considered as a remarkable achievement for a generalizable concept.

5 CONCLUSIONS

In this paper, we have proposed a family of probabilistic surrogate-assisted algorithms. The idea is based on improving the convergence of an existing algorithm by incorporating a surrogate's knowledge. The concept consists of a surrogate's influence through a tournament-based procedure with α competitors and a stronger surrogate's bias by using solutions with probability ρ derived from continuing the optimizing for β iterations on the surrogate. Experiments have shown that this effectively improves the convergence behavior on a variety of problems. A study on the surrogate-bias probability ρ has indicated that smaller or larger values are more suitable depending on the problem complexity. Thus, we have proposed an adaptive procedure of updating ρ depending on the surrogate's prediction error inspired by the well-known R^2 metric. PSA variants of CMA-ES, DE, GA, and PSO have shown competitive performance compared to other surrogate-based algorithms. Applying PSA to other variable types to further demonstrate the approach's capabilities shall be part of future work. Moreover, the effect of a local search to improve the convergence behavior near the optimum is worth investigating. Other interesting future studies for PSA are extensions to handle constraints and multiple objectives. This will require a suitable baseline algorithm and a modification of ρ estimation based on more than one surrogate. Altogether, the proposed probabilistic surrogate-assisted concept shall pave the way for new algorithms. PSA allows making use of existing algorithms' benefits to solve computationally expensive problems efficiently using a surrogate. Thus, this shall be an alternative to the widely-used *fit-and-optimize* method used in EGO and other algorithms.

REFERENCES

[1] The GPyOpt authors. 2016. GPyOpt: A bayesian optimization framework in python. (2016). http://github.com/SheffieldML/GPyOpt

[2] Samineh Bagheri, Wolfgang Konen, Richard Allmendinger, Jürgen Branke, Kalyanmoy Deb, Jonathan Fieldsend, Domenico Quagliarella, and Karthik Sindhya. 2017. Constraint handling in efficient global optimization. In *Proceedings of the genetic and evolutionary computation conference (GECCO '17)*. Association for Computing Machinery, New York, NY, USA, 673–680.

[3] P. Beaucaire, Ch. Beauthier, and C. Sainvitu. 2019. Multi-point infill sampling strategies exploiting multiple surrogate models. In *GECCO '19: Proceedings of the genetic and evolutionary computation conference companion*. ACM, New York, NY, USA, 1559–1567. Place: Prague, Czech Republic.

[4] Nicolas Berveglieri, Bilel Derbel, Arnaud Liefooghe, Hernán Aguirre, Qingfu Zhang, and Kiyoshi Tanaka. 2020. Designing parallelism in surrogate-assisted multiobjective optimization based on decomposition. In *Proceedings of the 2020 genetic and evolutionary computation conference (GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 462–470.

[5] J. Blank and K. Deb. 2020. Pymoo: Multi-objective optimization in python. *IEEE Access* 8 (2020), 89497–89509.

[6] X. Cai, L. Gao, and X. Li. 2020. Efficient generalized surrogate-assisted evolutionary algorithm for high-dimensional expensive problems. *IEEE Transactions on Evolutionary Computation* 24, 2 (2020), 365–379.

[7] George De Ath, Richard M. Everson, Jonathan E. Fieldsend, and Alma A. M. Rahat. 2020. ϵ-shotgun. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (June 2020). https://doi.org/10.1145/3377930.3390154

[8] K. Deb, R. Hussein, P. C. Roy, and G. Toscano-Pulido. 2019. A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 23, 1 (2019), 104–116.

[9] David E. Goldberg. 1989. *Genetic algorithms in search, optimization and machine learning* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., USA.

[10] Nikolaus Hansen. 2019. A global surrogate assisted CMA-ES. In *Proceedings of the genetic and evolutionary computation conference (GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 664–672.

[11] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. 2020. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software* (2020).

[12] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9, 2 (June 2001), 159–195.

[13] Rolland L. Hardy. 1971. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research (1896-1977)* 76, 8 (March 1971), 1905–1915.

[14] Donald R. Jones. 2001. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization* 21, 4 (2001), 345–383.

[15] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient global optimization of expensive black-box functions. *J. of Global Optimization* (1998).

[16] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95 - international conference on neural networks*, Vol. 4. 1942–1948 vol.4.

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

[18] D. G. Krige. 1951. A statistical approach to some basic mine valuation problems on the Witwatersrand, by D.G. Krige, published in the Journal, December 1951 : introduction by the author.

[19] D. Lim, Y. Jin, Y. Ong, and B. Sendhoff. 2010. Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation* 14, 3 (2010), 329–355.

[20] M. D. McKay, R. J. Beckman, and W. J. Conover. 1979. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21, 2 (1979), 239–245. Publisher: Taylor & Francis.

[21] Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. 2008. Multiobjective optimization on a limited budget of evaluations using model-assisted S-Metric selection. In *Parallel problem solving from nature –PPSN x*, G. Rudolph et al. (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 784–794.

[22] Nestor V. Queipo, Raphael T. Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P. Kevin Tucker. 2005. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences* 41, 1 (2005), 1–28.

[23] Jörg Stork, Martina Friese, Martin Zaefferer, Thomas Bartz-Beielstein, Andreas Fischbach, Beate Breiderhoff, Boris Naujoks, and Tea Tušar. 2020. Open issues in surrogate-assisted optimization. In *High-performance simulation-based optimization*, T. Bartz-Beielstein et al. (Ed.). Springer International Publishing, Cham, 225–244.

[24] Rainer Storn and Kenneth Price. 1997. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11, 4 (1997), 341–359.

[25] Felipe Viana and Raphael Haftka. 2010. Surrogate-based optimization with parallel simulations using the probability of improvement. In *13th AIAA/ISSMO multidisciplinary analysis optimization conference*. AIAA 2010–9392.

[26] Edward Wegman. 1990. Hyperdimensional data analysis using parallel coordinates. *J. Amer. Statist. Assoc.* 85 (1990), 664–675.

[27] Z. Zhan, J. Zhang, Y. Li, and H. S. Chung. 2009. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39, 6 (2009), 1362–1381.