Multi-Point Acquisition Function for Constraint Parallel Efficient Multi-Objective Optimization

Roy de Winter
r.de.winter@liacs.leidenuniv.nl
Leiden Institute of Advanced Computer Science
Leiden, The Netherlands

Bas van Stein
b.van.stein@liacs.leidenuniv.nl
Leiden Institute of Advanced Computer Science
Leiden, The Netherlands

Thomas Bäck
t.h.w.baeck@liacs.leidenuniv.nl
Leiden Institute of Advanced Computer Science
Leiden, The Netherlands

ABSTRACT

Bayesian optimization is often used to optimize expensive black box optimization problems with long simulation times. Typically Bayesian optimization algorithms propose one solution per iteration. The downside of this strategy is the sub-optimal use of available computing power. To efficiently use the available computing power (or a number of licenses etc.) we introduce a multi-point acquisition function for parallel efficient multi-objective optimization algorithms. The multi-point acquisition function is based on the hypervolume contribution of multiple solutions simultaneously, leading to well spread solutions along the Pareto frontier. By combining this acquisition function with a constraint handling technique, multiple feasible solutions can be proposed and evaluated in parallel every iteration. The hypervolume and feasibility of the solutions can easily be estimated by using multiple cheap radial basis functions as surrogates with different configurations. The acquisition function can be used with different population sizes and even for one shot optimization. The strength and generalizability of the new acquisition function is demonstrated by optimizing a set of black box constraint multi-objective problem instances. The experiments show a huge time saving factor by using our novel multi-point acquisition function, while only marginally worsening the hypervolume after the same number of function evaluations.

KEYWORDS

Parallel Computing, Bayesian Optimization, Multi-Objective Optimization, Constraint Optimization, Radial Basis Functions

ACM Reference Format:
Roy de Winter, Bas van Stein, and Thomas Bäck. 2022. Multi-Point Acquisition Function for Constraint Parallel Efficient Multi-Objective Optimization. In *Proceedings of The Genetic and Evolutionary Computation Conference 2022 (GECCO '22).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3512290.3528696

GECCO '22, July 9–13, 2022, Boston, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9237-2/22/07…$15.00
https://doi.org/10.1145/3512290.3528696

1 INTRODUCTION

Often engineering goes hand in hand with optimization by using expensive simulations. According to [23], solving expensive optimization problems faster, three things can be done: (1) problem approximation and substitution, (2) algorithm design enhancement, (3) parallel and distributed computation. In this paper, we demonstrate how a variety of these techniques are combined in one algorithm. Besides the proposed improvements, in this paper we also assume that the evaluation of one solution, of the problem at hand, is computationally expensive, making the computational cost of the algorithm negligible.

Bayesian optimization (BO) or Efficient Global Optimization (EGO) [22] is a common solution for expensive black-box optimization problems. EGO approximates the fitness and constraint functions by using meta models or surrogates. In each iteration, EGO finds a new promising solution by optimizing an inexpensive function using the response surface of the meta models. These so called inexpensive functions which are used to find the optimum on the meta models in each iteration are better known as acquisition function or infill criterion. After selection of a promising solution, and the evaluation of this candidate solution, the meta models are updated with the new results. Over time, different acquisition functions have been published for different meta modelling techniques and for different purposes e.g; for single objective optimization [22], with an emphasis on exploration/exploitation [33], for constraint optimization [3], for parallel optimization [18], multi-objective optimization [29] and for constraint multi-objective optimization [12]. However, not much attention has been spend on an acquisition function that can both handle constraints, multiple objectives, and propose multiple solutions for evaluation in parallel in an efficient manner.

For many real-world problems, candidate solutions can be evaluated in parallel using large computer clusters and multiple simulations. To make use of these resources, the optimization algorithm needs to be able to propose multiple candidate solutions in each iteration. Evaluating multiple solutions in parallel can reduce the total wall clock time significantly. Following the example of Li et al. [18], the total evaluation time, also referred to as the total cost of solving a computationally challenging optimization problem can be formulated as follows: $Totalcost = O(C) \cdot O(N)$. Here $O(C)$ is the average cost of the expensive evaluation, and $O(N)$ the average number of iterations of the optimization algorithm until a satisfactory solution is found. If two expensive evaluations can be run in parallel the cost can already be cut in half in terms of wall clock time ($p = 2$), i.e., $Totalcost = \frac{O(C) \cdot O(N)}{p}$. Obviously, when

p solutions are proposed per iteration, the total cost can also be reduced by a factor p. The downside of proposing multiple solutions simultaneously is that the new batch of p solutions is selected based on the meta models trained on $p-1$ less samples as opposed to the sequential optimization procedure (where $p=1$). This means that using parallel evaluations, potentially results in additional required function evaluations compared to a sequential optimization run with a single solution per iteration.

The scientific contribution of this paper is a new acquisition function which incorporates problem approximation and substitution, algorithm design enhancement and parallel and distributed computing techniques. This acquisition function is introduced and used in the SAMO-COBRA algorithm [11] to demonstrate the effectiveness of proposing multiple solutions simultaneously. This makes the parallel SAMO-COBRA algorithm capable of doing multi-objective optimization while dealing with constraints and doing batch or one-shot optimization. The results of the algorithm indicate that the missed information per iteration can also be beneficial since it also provides a mean of exploration.

The paper is organized as follows, in Section 2 the related work is discussed. In Section 3 the mathematical problem definition is presented in more detail. In Section 4 the new acquisition function is introduced and explained. In Section 5 different algorithm configuration and variants are discussed. In Section 6 various experiments are described. In Section 7 the results are presented and discussed. Finally in Section 8 the final conclusions are drawn.

2 RELATED WORKS

Originally, evolutionary algorithms, genetic algorithms, particle swarm optimization are population based [2]. The evaluation of these populations can naturally be parallelized. However, evolutionary algorithms typically require a lot of function evaluations because they move in small steps before they converge to the global optimum. In case the objective and/or the constraint functions are expensive to evaluate, the functions are typically mimicked with a meta model or cheap surrogate. The surrogate is then used to search promising solutions which can be evaluated. These surrogate assisted optimization algorithms on the other hand typically do not use acquisition functions which can propose multiple solutions simultaneously. Allowing the surrogate assisted algorithms to only propose one solution per iteration, leads to longer running times and ineffective use of available resources.

2.1 Parallel Single Objective Optimization

According to a survey on parallel single objective optimization [18], the three most obvious techniques for parallelization are; multi-start local searches (if derivatives of the objective function are available), multiple parallel optimization runs (optionally in different sub-regions), and as described above with a population of designs. Other parallelization techniques often tend to combine different acquisition functions with different hyper-parameters to balance exploration and exploitation. Wang et al. [39] for example proposed a single objective multi-point acquisition function for Bayesian optimization. This acquisition function is based on the moment-generating function where the expected improvement is raised to the power t. For different values of t, the moment-generating function will therefore result in different proposed solutions with different trade offs between exploration and exploitation.

Other techniques used to select p different solutions simultaneously are, for example, the following: Assuming a found solution is correct, optimize the acquisition function again until p solutions are found [16], also known as the knowledge or Kriging believer. It is also possible to use different surrogates (or weighted combinations of surrogates) fitted on the same data and optimize these different surrogate models [20].

2.2 Parallel Multi-Objective Optimization

Besides the algorithm described in this paper, several surrogate assisted multi-objective algorithms are already proposed where multiple points are proposed per iteration e.g. MIP-EGO [36], MMBO [38], MOPLS-N [1]. The downside is that they all lack a constraint handling mechanism and fail to propose solutions on the constraint boundaries.

Important to keep in mind when doing multi-objective optimization is that as described in our earlier related work [11], multi-objective optimization algorithms are due to the problem characteristics, already forced to explore more, compared to single objective optimization algorithms. This way exploration is naturally inherited in multi-objective optimization algorithms. In our earlier related work [11], it is numerically shown that for constrained multi-objective problems the purely exploiting infill criterion named Predicted HyperVolume (PHV) lead to better Pareto frontier approximations compared to the more exploring infill criterion S-Metric Selection criteria [30].

Mixed-Integer Parallel Efficient Global Optimization (MIP-EGO) [36] for example is designed to automatically optimize the configuration of artificial neural networks. MIP-EGO uses multiple random forests as surrogates and different infill criteria are optimized to propose different solutions simultaneously.

Wada and Hino proposed MMBO [38], a Bayesian multi-objective multi-point optimization algorithm together with a gradient approximation of the acquisition function. This algorithm proposes multiple points simultaneously every iteration based on multi-point expected hypervolume improvement. This algorithm uses the expected hypervolume improvement as infill criteria and therefore uses the uncertainty quantification of the solutions to balance exploration and exploitation.

Akhtar and Schoemaker proposed MOPLS-N [1], a Multi Objective Population-based Parallel Local Surrogate-Assisted Search. MOPLS-N uses Radial Basis Functions (RBF) as surrogates, uses parallel local candidate search from the parent population centers, and uses boxed hypervolume improvement to judge one candidate solution in a box around one center.

2.3 Constrained Parallel Multi-objective Optimization

Additionally, as also mentioned in the survey [18], there is still a lack of well-performing adaptive sampling algorithms for constrained optimization. Constraint optimization is traditionally done by making use of penalty functions [18]. Tuning these penalty functions demand a lot of function evaluations [3]. To save function evaluations during the optimization process, just like for the objective functions, surrogates can be used to model the constraint functions.

One multi-objective optimization algorithm is found with both a constraint handling mechanism and a multi-point infill criteria. Wauters et al. [40] proposed the Generalized Asynchronous Multi-objective Expected Improvement infill criteria (GAMOEI). GAMOEI allows multiple points to be selected for evaluation asynchronously while balancing exploration and exploitation in an adaptive manner. The expected improvement infill criteria depends on the regular multi-objective expected improvement raised to a higher power. Constraints are dealt with by multiplying the probability of feasibility with the expected improvement. In their experiments, this resulted in undesirable points far away from the Pareto frontier with little to no points on the constraint boundaries.

2.4 One Shot Optimization

One shot optimization [5, 6] or global surrogate modeling can be characterized by surrogate assisted optimization algorithms where a surrogate is fitted only once with training data of an initial sample. After the surrogate is fitted, an optimal solution (or set of optimal solutions) is found on the surrogate, the obtained solutions are evaluated and the algorithm terminates. This means that in contrast to other surrogate assisted optimization algorithms there is no evaluation budget for adaptive sampling. One shot optimization is very popular and a classical approach in the maritime [34], automotive [35], aerospace [27] and other engineering domains. As already stated in the introduction, a lot of potentially available information for the last evaluation is missing when using this approach as all new solutions should be found based on the first initial samples. A benefit of one shot optimization is that when a lot of computational resources are available they can easily be exploited.

3 PROBLEM DEFINITION

This section is organized as follows, first the constrained multi-objective problem is defined. Secondly, the single point acquisition is described and mathematically formulated.

3.1 Constrained Multi-Objective Problem

A Constrained multi-objective problem can be defined as follows:

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

In this formulation, Ω is the decision space, $\mathbf{x}$ is a solution in the decision space, k is the number of objectives functions, f_i is the i^{th} objective function, m is the number of constraints, g_j is the j^{th} constraint function, and d is the number of decision variables.

To determine if a solution $\mathbf{x}$ is a good solution, the solution should be evaluated by all constraint and objective functions. For expensive engineering problems it is often not feasible to evaluate all possible solutions in a brute force manner to find the optimal solutions and the trade-offs between the objectives. To avoid the evaluation of solutions which are not interesting during the optimization process, the real objectives and constraint functions are replaced with cheap surrogates.

The surrogates considered in this research are a weighted combination of Radial Basis Functions (RBFs) with a second order polynomial tail [7, 24, 31]. For each objective and for each constraint, a unique and independent RBF surrogate is fitted. The RBF surrogate for the i^{th} objective is denoted by f_i', the RBF surrogate for the j^{th} constraint is denoted by g_j'. The k RBF surrogates for the objectives functions together are denoted by $\mathbf{f}'$, the m RBF surrogates for the constraints are denoted by $\mathbf{g}'$. In the optimization process, the RBFs are refitted every time a new solution is evaluated on the real objective and constraint function. This way, the RBFs are always based on all the available information from all the evaluated solutions. Given the RBF surrogates, the constraint and objective values can be predicted for each candidate solution. With the RBF prediction for the constraints, it can be predicted for any solution whether it is feasible or how much the solution violates the constraints. With the prediction for the objectives, it can be predicted how good a solution scores in every objective. The choice of selecting a solution to be evaluated on the real objective and constraint functions is based on the acquisition function. Finding a new solution is therefore based on optimizing an acquisition function.

3.2 Single-Point Acquisition Function

The acquisition function score for a solution $\mathbf{x}$ is dependent on the RBF surrogates of the objectives. With the predicted objective values, it can be predicted how good the solution is in each objective. Whether a solution $\mathbf{x}_A$ is preferred above an alternative solution $\mathbf{x}_B$ is dependent on the constraint violation and the acquisition function score. A successful acquisition function from our earlier work for constrained multi-objective problems is the PHV infill criteria [11]. The PHV infill criteria (denoted by $\mathcal{P}_{hv}$) returns the size of the space that is dominated by a solution between the solution on the Pareto frontier and a predefined reference point. The PHV infill criteria only returns the size of the dominated space of the new solution if this space is not already dominated by solutions which are evaluated before. In the case of two objectives, the space a solution dominates is equal to the size of a surface that is dominated. In the case of three objectives, the space is a volume that is dominated. For more objectives the space becomes a hypervolume. A visual representation of two dimensions is given in Figure 1.

With the RBF surrogates for the objectives ($\mathbf{f}'$) and for the constraints ($\mathbf{g}'$), and the PHV infill criteria ($\mathcal{P}_{hv}$), any solution ($\mathbf{x}$) can be judged on feasibility and desirability for all objectives. The single point acquisition function optimization problem can now be mathematically defined as follows:

$$\begin{aligned} \mathbf{x}^* \in \underset{\mathbf{x} \in \Omega \subset \mathbb{R}^d}{\operatorname{argmax}}\ &\mathcal{P}_{hv}(\mathbf{f}'(\mathbf{x})) \\ \text{subject to } &\mathbf{g}'(\mathbf{x}) \leq \mathbf{0} \end{aligned}$$

4 MULTI-POINT ACQUISITION FUNCTION

Besides judging the solution quality of one solution at a time with the PHV infill criteria, the PHV infill criteria can also be used to evaluate multiple solutions simultaneously. For this to happen, first the optimization problem should be redefined such that multiple solutions can be judged on solution quality. Multiple solutions can

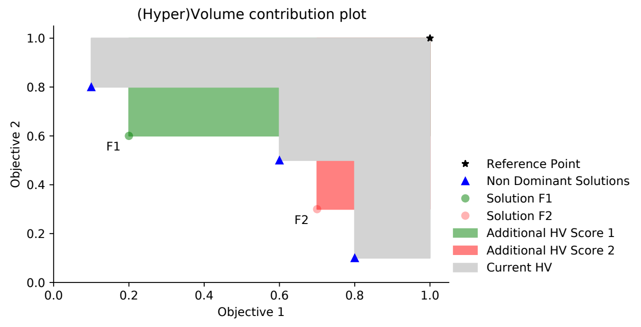


Figure 1: Visual representation of hypervolume contribution of two solutions. The hypervolume contribution of solution F1 is equal to $0.2 \cdot 0.4 = 0.08$, the hypervolume contribution of solution F2 is equal to $0.1 \cdot 0.1 = 0.01$. This makes solution F1 more desirable compared to solution F2.

be represented in one vector by simply concatenating the different solution vectors. Suppose one solution still contains d decision variables, then p solutions together can be formulated as a vector of $d \cdot p$ real values $\mathbb{R}^{p \cdot d}$. In this formulation the first d values represent the first solution, where the last d values in this vector represents the p^{th} solution.

Given the p solutions $\mathbf{x}_i$, and the cheap RBF surrogate for each objective, also p predictions can be made for each objective. Since there are p solutions and k objectives, the RBF predictions can be combined in a vector as follows:

$$\mathbf{A} = \left(f_1'(\mathbf{x}_1), \ldots, f_k'(\mathbf{x}_1), \ldots, f_1'(\mathbf{x}_p), \ldots, f_k'(\mathbf{x}_p) \right)$$

Here the vector $\mathbf{A}$ has a length of $p \cdot k$. The p solutions with the corresponding $p \cdot k$ predictions for the k objectives can, after this step, be split into p solutions with k objective values. These p solutions can then be mapped to the objective space so that their combined performance in terms of hypervolume contribution can be judged. The judging of how good the combined p solutions are can again be computed with the PHV infill criteria resulting in a Multi-Point Acquisition Function ($\mathcal{MP}_{hv}$). The hypervolume contribution of a set of solutions can be computed with the individual hypervolume contribution of each solution minus the overlap. Because this multi-point acquisition function evaluates p solutions simultaneously, it will automatically prefer a set of solutions with diverse objective scores above a set of similar solutions with similar objective scores. This is the case because, a set with diverse solutions with little overlapping hypervolume will dominate more objective space compared to a set of solutions with very similar scores with a lot of overlapping hypervolume. Note that after these steps are taken, a set of p solutions are mapped into $p \cdot k$ objective predictions, and this is then translated with the multi-point acquisition function into a single real value which represents the hypervolume contribution of the p solutions. Predicting p solutions simultaneously does not increase the total number of RBF surrogates, only the RBF surrogates are now used p times when evaluating p new solutions simultaneously. A visual representation of the multi-point acquisition function is given in Figure 2.

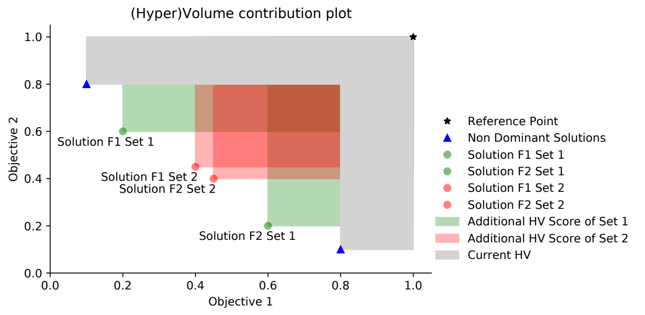


Figure 2: Visual representation of hypervolume contributions of two sets containing two solutions each. The hypervolume contribution of set 1 is equal to $2(0.6 \cdot 0.2) - (0.2 \cdot 0.2) = 0.2$. The hypervolume contribution of set 2 is equal to $2(0.35 \cdot 0.4) - (0.35 \cdot 0.35) = 0.1575$. So although the individual hypervolume contributions of the solutions of set 2 are higher compared to the individual hypervolume contributions of the solutions in set 1, the hypervolume contribution of set 2 is smaller compared to the hypervolume contribution of set 1. This makes set 1 more desirable compared to set 2.

A similar formulation and strategy is used for the constraints. Because multiple solutions are now to be dealt with, also all the p solutions should be judged on feasibility simultaneously. Each solution has m constraints, leading to $p \cdot m$ constraint values to consider. With the RBF surrogates ($\mathbf{g}$) representing the m constraints, each RBF surrogate (g_j) can be used p times to predict the constraint values for the p solutions. This results in one long constraint vector of length $p \cdot m$, the first m constraint values represent the m constraint values for the first solution, the last m constraint predictions represent the constraints for the p^{th} solution.

With this new formulation for p solutions simultaneously, the multi-point acquisition optimization problem can be mathematically represented in the following way:

$$(\mathbf{x}_1^*, \ldots, \mathbf{x}_p^*) \in \underset{\mathbf{x}_i \in \Omega \subset \mathbb{R}^d}{\operatorname{argmax}} \ \mathcal{MP}_{hv}(\mathbf{f}'(\mathbf{x}_1), \ldots, \mathbf{f}'(\mathbf{x}_p))$$
$$\text{subject to } \mathbf{g}'(\mathbf{x}_i) \leq \mathbf{0}$$

4.1 Optimization of Acquisition Function

For the optimization of the multi-point acquisition function, any optimizer capable of optimizing one objective and dealing with multiple constraints can be chosen. In this work, the COBYLA (Constrained Optimization BY Linear Approximations) algorithm [32] is selected for this task. COBYLA linearly approximates the acquisition function and each constraint separately and then optimizes the linear approximations in a trust region. By iteratively, refitting the linear approximations and adjusting the trust region, COBYLA converges to a solution which can not be further improved. By letting COBYLA start from a random generated vector of length $p \cdot d$ representing p solutions, COBYLA iteratively also optimizes these p solutions. Important to remember is that COBYLA does not use the real objective and constraint functions but the RBFs of the constraint and the RBFs of the objectives to optimize the the acquisition function.

Since COBYLA can get stuck in local optima [26], it is wise to start COBYLA from more then one random generated vector. Experiments showed that the optimization problem characteristics like the number of decision variables d, the number of constraints m, the number of objectives k, and the number of solutions to be optimized in parallel p, all have influence on if COBYLA can converge to good solutions. The experiments showed that more random starting points, and larger evaluation budget for COBYLA lead to better results. However, more starting points and larger evaluation budgets for COBYLA also lead to a higher computational costs. Therefore, as a rule of thumb, it is recommended to let COBYLA converge from $2(d+m+k)$ solutions when using the single point acquisition function, let COBYLA converge from $4(d+m+k)$ when using the multi-point acquisition function, and when doing one shot optimization, let COBYLA converge from $8(d+m+k)$ solutions. A similar rule is created for the evaluation budget of COBYLA, the budget for when using the single point acquisition function is $50(d+m+k)$, when using multi-point acquisition function $100(d+m+k)$, and when doing one shot optimization the evaluation budget for COBYLA is $200(d+m+k)$. After COBYLA has converged from the starting points, the solution set with the highest acquisition score is selected to be evaluated on the real objectives and constraint functions. If COBYLA can not find a feasible solution, the solution set with the smallest cumulative constraint violation is selected and evaluated on the real objective and constraint functions.

5 ALGORITHM CONFIGURATION

With the RBFs, acquisition function, and the constraint handling technique, two algorithms are configured. One algorithm proposes and evaluates a predefined number of solutions simultaneously so that the computer infrastructure and available licences can be optimally used. The other algorithm is configured for one shot optimization for when a cluster of computers is available and simulation licenses are not an issue. Before the algorithm variants are explained in more detail, the RBF configurations and scaling techniques are explained.

5.1 RBF Hyper-Parameter Settings

RBFs are in contrary to Kriging or Gaussian process regression models cheap to train. This allows us to fit a lot of different RBFs with different hyper-parameters. Hyper-parameters considered in this work are the type of RBF kernel (Cubic, Gaussian, Multiquadric, Inverse Multiquadric, Thin Plate Spline), and transformation strategies of the input data (Scale, Plog, and Standardize). More details about the RBF hyperparameters can be found in [11].

The selection of the optimal hyper-parameter settings is not straightforward, in the first iteration the surrogate models have exactly enough training points for a first RBF to be fitted, and after a surrogate is fitted, the RBFs exactly interpolate all the solutions that are used for training. For this reason, only after evaluation of a real objective or constraint function which has not yet been used for fitting the RBF, the approximation error can be computed. These approximation errors can be determined every time a new solution is evaluated on all the constraints. These approximation errors are stored as historic approximation errors for every RBF configuration. The optimal RBF configurations are selected based on the historic approximation errors of the last two sets of evaluated solutions and the historic approximation errors of the Pareto efficient solutions. The historic approximation errors of the Pareto efficient solutions are selected so that the vicinity of the Pareto efficient solutions are well fitted. The historic approximation errors of the last two sets are selected so that the RBF configuration does not get stuck in a local optimum configuration.

5.2 Parallel Optimization

The SAMO-COBRA algorithm, a constrained multi-objective optimization algorithm is configured such that the multi-point acquisition function is used instead of the original acquisition function which proposes one solution per iteration. The algorithm starts by evaluating an as small as possible initial Halton sample [19], if the batch size for the acquisition function is larger then $d+1$, the batch size is chosen as initial saple size. The initial sample size is chosen small so that there is more evaluation budget left for adaptive sampling steps which in the end will lead to better results. Then the first RBFs are trained, an RBF configuration and transformation strategy is selected so that in the first iteration a predefined number of solutions can be proposed. The set of solutions is then evaluated by the real objective and constraint functions. After each iteration the RBFs with different hyper-parameters are updated, the best RBF configuration is selected based on the historic approximation errors, and the multi-point acquisition function is optimized again. This continues until the evaluation budget is exhausted and the Pareto front is found.

5.3 One Shot Optimization

The one shot optimization algorithm is configured such that the initial sample is of a size equal to half the evaluation budget. After the initial Halton sample is evaluated, the RBFs with the different configurations are fitted and the best configurations are selected. Selection of the best input transformation and RBF kernel can in this case not be done based on historic approximation error. Nor can the best configuration be selected based on the RBFs trained with all the input data. Instead 10-fold cross validation is used to select the RBF kernel and transformation strategy. Selecting the optimal RBF configuration based on 10 fold cross validation requires some computation time, however spending half of the evaluation budget based on wrongly estimated solutions is for obvious reasons much more expensive. After selection of the optimal RBF configurations the multi-point acquisition function is optimized. The multi-point acquisition function is optimized such that in one run all solutions for the other half of the evaluation budget can be found. Finally the predicted optimal solutions are evaluated with the real objectives and constraint functions and the algorithm terminates.

6 EXPERIMENTS

To test the performance of the algorithm and acquisition function, several experiments are conducted. In the experiments, different batch sizes are tested for the multi-point acquisition function: 1 (original), 2, 3, 4, 5, 6, 10 and 20. Bigger batch sizes are not considered because then it becomes too similar compared to one shot optimization. The test functions from Table 1 are selected for the

Table 1: Test function with citation, the reference point used by optimization algorithm, the Nadir point approximation, k is number of objectives, d number of decision variables, m number of constraints, and $P(\%)$ the percentage of feasible solutions after 1 million random samples.

Function	Reference point	Nadir point	k	d	m	$P(\%)$
BNH [8]	(140, 50)	(136.00, 49.24)	2	2	2	96.92
CEXP [13]	(1, 9)	(1.00, 9.00)	2	2	2	57.14
SRN [14]	(301, 72)	(222.97, 2.60)	2	2	2	16.18
TNK [14]	(2, 2)	(1.04, 1.04)	2	2	2	5.05
CTP1 [13]	(1, 2)	(0.99, 1.00)	2	2	2	92.67
C3DTLZ4 [37]	(3, 3)	(2.00, 2.00)	2	6	2	22.22
OSY [8, 14]	(0, 386)	(-42.17, 76.00)	2	6	6	2.78
TBTD [17]	(0.1, 50000)	(0.1, 92774.46)	2	3	2	19.46
NBP [15]	(11150, 12500)	(12500, 114.09)	2	2	5	41.34
DBD [17]	(5,50)	(2.79, 26.40)	2	4	5	28.55
SPD [28]	(16, 19000, -260000)	(11.24, 12434.76, -292146.91)	3	6	9	3.27
CSI [21]	(42, 4.5, 13)	(41.61, 4.00, 12.52)	3	7	10	18.17
SRD [25]	(7000, 1700)	(5781.90, 1690.16)	2	7	11	96.92
WB [17]	(350, 0.1)	(53.72, 0.0145)	2	4	5	35.28
BICOP1 [9]	(9, 9)	(1, 1)	2	10	1	100
BICOP2 [9]	(70, 70)	(1.1, 1.08)	2	10	2	10.55
TRIPCOP [9]	(34, -4, 90)	(7.67, -11.77, 25.91)	3	2	3	15.85
WP [21]	(83000, 1350, 2.85, 15989825, 25000)	(75684, 1347, 2.85, 7625734, 24896)	5	3	7	92.06

experiments. Each test function is optimized in 11 independent runs with different seeds. Optimization of the test functions is done by using a reference point which is the worst possible objective score per function as possible. The Nadir point [4] of the test functions is approximated by taking the extremes of the objective scores on the Pareto frontier from all combined experiment results. The hypervolume reported in the results of the experiments are calculated by computing the hypervolume between the Pareto frontier and the Nadir point. The algorithms variants, the experiments, and the results can all be found on Github [10].

6.1 Hypervolume after Fixed Evaluation Budget

In the first experiment the hypervolume between the approximated Nadir point and the obtained Pareto frontier is calculated after a fixed evaluation budget. The algorithm with the different batch sizes has a total allowed evaluation budget of $40 \cdot d$. This evaluation budget leads to an initial Halton sample of $d+1$ samples and $39 \cdot d - 1$ iterations for the SAMO-COBRA algorithm with the single-point infill criteria. While for batch sizes larger then 1, it leads to $\max(d+1, p)$ initial Halton samples and $\frac{40 \cdot d - \max(d+1, p)}{p}$ iterations.

6.2 Convergence Experiment

As explained in the introduction, with batch optimization a lot of wall clock time can be saved. The downside of proposing multiple solutions simultaneously is that the new batch of solutions is based on less information compared to when one solution would be added per iteration. In this experiment, it is tested how much information is lost per iteration, and on the other hand how much time can be saved. This is tested by taking 90% of the maximum achievable hypervolume as a threshold, then the algorithm convergence results can tell how much algorithm iterations and total number of function evaluations are required to achieve this hypervolume threshold for the different batch sizes.

6.3 One Shot Optimization Experiment

In the last experiment the algorithm and multi-point acquisition function is tested to see if it is capable of one shot optimization. The one shot optimization algorithm configuration is tested with 40 initial Halton samples and then in one iteration 40 new optimal solutions are proposed and evaluated. The hypervolume between the Nadir Point and the obtained Pareto frontier is computed and compared to the hypervolume obtained with batch size 1.

7 RESULTS

The results of the three experiments are presented in two Pareto frontiers, two convergence plots and three tables. The overall results show that for test functions with strict constraints larger batch sizes lead to worse results after the same number of function evaluations. For other test functions with a higher feasibility rate, larger batch sizes can be very beneficial in terms of required number of evaluations and therefore iterations.

7.1 Hypervolume Results

In Table 2 the mean hypervolume and standard deviation of the hypervolume between the Pareto frontier and Nadir point is given for the different test functions. As can be seen in the table, the hypervolume in most cases slightly decreases and the standard deviation increases, when a larger batch size is chosen. In few cases, the mean hypervolume is significantly better for larger batch sizes. It is expected that this is the case because more exploration can be beneficial for functions which are hard to fit with RBF models with few initial data samples.

For two test functions, the obtained feasible solutions are plotted for the algorithm with different batch sizes. In Figure 3a all the obtained feasible solutions of the test problem TNK are presented. In this figure it can be seen that the batch size 1 almost never misses the Pareto frontier, while solutions of the larger batch sizes are often dominated by other solutions from these batch sizes. In Figure 3b all the obtained feasible solutions of the test problem C3DTLZ4 are presented. In this figure it can be seen that the solutions with the larger batch sizes show a better coverage among the Pareto frontier versus the solutions from other batch sizes.

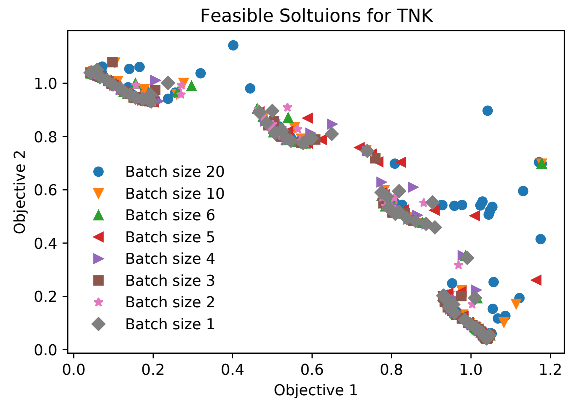
7.2 Convergence Results

The results of the second experiment can be found in Table 3. As can be seen in the table, for the majority of the test functions, the threshold of 90% was reached before the allowed number of function evaluations. When comparing batch size 1 with the larger batch sizes for each test function. The best result with the least number of iterations on average required 75% less iterations, the trade-off is that the number of evaluations on average increases with 58% to find the 90% hypervolume threshold. So in the cases where time consuming objective and constraint functions can be evaluated in parallel, the wall clock time can significantly be reduced.

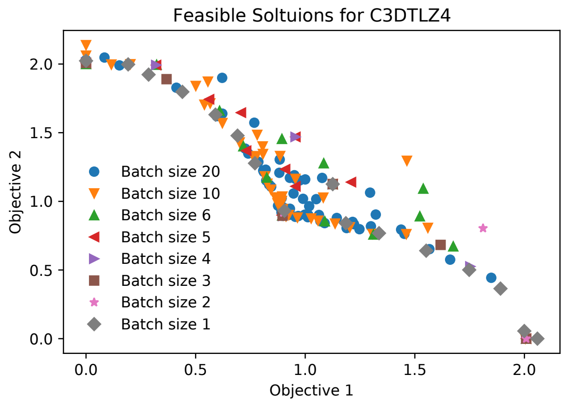
In Figure 3c the convergence plot is given for the TNK test function. For this test function, the algorithm with different batch size combinations all converge to the approximated optimum except for batch size 20. In Figure 3d the convergence plot is given for the C3DTLZ4 function. Interestingly enough, in this convergence plot the extra exploration which is naturally included for larger batch sizes seems to be beneficial since the larger batch sizes 20, 10 and 6 perform better compared to batch sizes, 2, 3, 4, and 5.

Table 2: Mean and standard deviation of hypervolume (hv) on set of test functions after $40 \cdot d$ function evaluations with different batch sizes (1,2,3,4,5,6,10,20) for the $\mathcal{P}_{hv}$ infill criteria given 11 independent runs. HV Scores in bold indicate a higher mean compared to batch size 1. A * is added if the difference was significant according to the Wilcoxon rank-sum test with $p < 0.05$.

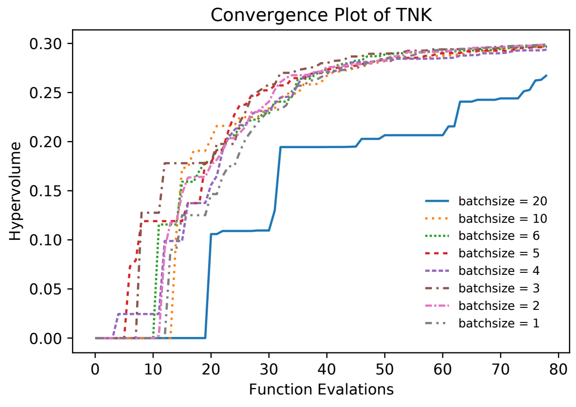
Function	Batch size 1		Batch size 2		Batch size 3		Batch size 4		Batch size 5		Batch size 6		Batch size 10		Batch size 20	
	hv	std	hv	std	hv	std	hv	std	hv	std	hv	std	hv	std	hv	std
BNH	4969.2	0.0133	4969.0	0.1	4967.5	1.5	4967.6	0.5	4967.9	0.7	4967.6	1.2	4960.3	2.3	4949.8	5.7
CEXP	3.7972	0.0005	3.7961	0.0015	3.7963	0.0016	3.7944	0.001	3.7964	0.0004	**3.7981***	0.0004	3.7925	0.0005	3.7794	0.0030
SRN	25019	5	25008	10	24977	16	24843	22	24729	53	24723	39	24583	97	24516	103
TNK	0.2988	0.0016	0.2966	0.0033	0.2965	0.0026	0.2949	0.0018	0.2953	0.0012	0.2985	0.0018	0.2957	0.0024	0.2676	0.0144
CTP1	0.2985	0.0001	0.2985	0.0001	0.2984	0.0001	0.2984	0.0001	0.2981	0.0001	0.2984	0.0002	0.2977	0.0003	0.2956	0.0008
C3DTLZ4	1.5446	0.0759	1.4288	0.17	1.3569	0.0018	1.2526	0.0473	1.3375	0.0512	1.3933	0.0813	1.5091	0.0659	**1.5827**	0.0308
OSY	12629	2	12352	97	12609	3	12526	71	12396	139	12443	90	12073	105	11501	297
TBTD	8052.6	48.5	7892.3	90.0	7690.2	153.9	7506.0	237.4	7471.3	205.5	7419.4	300.2	7237.2	272.6	7213.8	231.5
NBP	799579	190	800186	130	**799770***	258	798810	643	798377	844	797753	1496	793709	2257	776697	1517
DBD	59.9960	0.0806	**60.0550***	0.0152	**60.0614***	0.0063	**60.0034**	0.0391	59.9676	0.0334	59.8961	0.0262	59.8108	0.0296	59.6967	0.0506
SPD	$5.511 \cdot 10^9$	$2 \cdot 10^6$	**$5.513 \cdot 10^9$**	$3 \cdot 10^6$	$5.502 \cdot 10^9$	$3 \cdot 10^6$	$5.493 \cdot 10^9$	$8 \cdot 10^6$	$5.497 \cdot 10^9$	$8 \cdot 10^6$	$5.474 \cdot 10^9$	$1.3 \cdot 10^7$	$5.369 \cdot 10^9$	$1.7 \cdot 10^7$	$5.259 \cdot 10^9$	$3.0 \cdot 10^7$
CSI	7.5394	0.0038	7.5343	0.0049	**7.5438**	0.0064	7.5372	0.0077	**7.5432**	0.0076	**7.5409**	0.0112	7.4329	0.018	6.8714	0.0704
SRD	2952123	95	2949030	574	2945522	755	2941958	935	2940695	1019	2939470	2003	2934512	941	2925597	3690
WB	0.6375	0.0185	**0.6435**	0.0133	0.6373	0.0138	**0.6416**	0.0209	**0.6475**	0.0128	0.6089	0.0263	0.6213	0.0169	0.5952	0.0125
BICOP1	0.6640	0.0004	0.6609	0.0010	0.6442	0.0052	0.6226	0.0111	0.6029	0.0160	0.5901	0.0139	0.4302	0.1118	0.3375	0.0809
BICOP2	0.2549	0.0381	**0.2623**	0.0161	0.2289	0.0358	0.2294	0.0364	0.2320	0.0356	0.2267	0.0160	0.2198	0.0435	0.2138	0.0250
TRICOP	49.6407	0.0430	**49.6971***	0.0206	**49.7224***	0.0215	**49.6470**	0.0449	**49.7100**	0.0402	**49.7270***	0.0259	49.5006	0.0825	49.3136	0.1001
WP	$3.677 \cdot 10^{18}$	$5 \cdot 10^{15}$	$3.662 \cdot 10^{18}$	$3 \cdot 10^{15}$	$3.653 \cdot 10^{18}$	$9 \cdot 10^{15}$	$3.631 \cdot 10^{18}$	$1.4 \cdot 10^{16}$	$3.583 \cdot 10^{18}$	$1.4 \cdot 10^{16}$	$3.556 \cdot 10^{18}$	$1.3 \cdot 10^{16}$	$3.492 \cdot 10^{18}$	$2.1 \cdot 10^{16}$	$3.471 \cdot 10^{18}$	$1.5 \cdot 10^{16}$



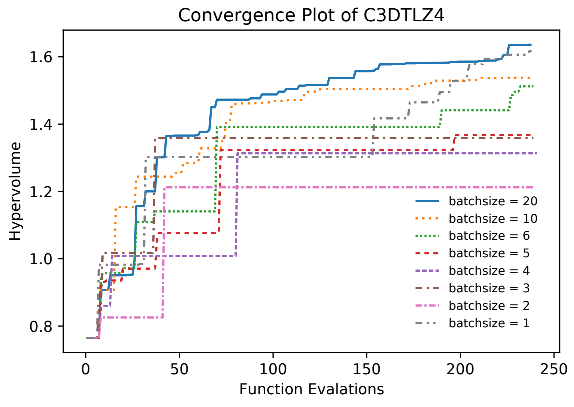
(a) Obtained feasible solutions in objective space for TNK test function. With a different color for different batch sizes used in the acquisition function.



(b) Obtained feasible solutions in objective space for C3DTLZ4 test function. With a different color for different batch sizes used in the acquisition function.



(c) Convergence plot for TNK function. With a different color for different batch sizes used in the acquisition function.



(d) Convergence plot for C3DTLZ4 function. With a different color for different batch sizes used in the acquisition function.

Figure 3: Obtained Pareto Frontiers and Convergence plots for TNK and C3DTLZ4 test function

Table 3: Rounded mean number evaluations (Eval), mean number of algorithm iterations (Itr) given the different batch sizes (1, 2, 3, 4, 5, 6, 10, 20) to achieve the hypervolume threshold. The hypervolume threshold is 90% of the total dominated area between the Nadir point and the Pareto frontier of all runs together. A dash (-) indicates that the threshold is not achieved within $40 \cdot d$ function evaluations for the 11 runs, an arrow down (⇃) indicates that not every run reached the threshold.

Function	Threshold	Batch size 1		Batch size 2		Batch size 3		Batch size 4		Batch size 5		Batch size 6		Batch size 10		Batch size 20	
		Eval	Itr	Eval	Itr	Eval	Itr	Eval	Itr	Eval	Itr	Eval	Itr	Eval	Itr	Eval	Itr
BNH	4496.2	9	9	10	5	10	4	8	2	8	2	9	2	13	2	21	2
CEXP	3.4380	9	9	11	6	11	4	12	3	13	3	11	2	17	2	31	2
SRN	22810	17	17	19	10	20	7	22	6	18	4	20	4	30	4	37	2
TNK	0.2775	44	44	47	24	48	16	48	12	46	10	44	8	49	5	68⇃	4⇃
CTP1	0.2717	13	13	15	8	15	5	15	4	19	4	16	3	19	2	34	2
C3DTLZ4	1.5788	197⇃	197⇃	219⇃	110⇃	-	-	-	-	-	-	-	-	232⇃	24⇃	187⇃	10⇃
OSY	11393	18	18	64	33	20	7	27	7	37	8	25	5	39	4	125⇃	7⇃
TBTD	7359.8	16	16	29	15	43	15	57⇃	15⇃	61⇃	13⇃	36⇃	6⇃	76⇃	8⇃	77⇃	4⇃
NBP	725935	16	16	15	8	14	5	19	5	19	4	20	4	26	3	46	3
DBD	54.133	14	14	15	8	15	6	14	4	15	3	20	4	27	3	32	2
SPD	$5.106 \cdot 10^9$	59	59	58	30	62	21	68	17	62	13	77	13	103	11	140	7
CSI	7.1691	120	120	124	62	119	40	118	30	119	24	123	21	179	18	-	-
SRD	2658080	14	14	14	7	16	6	17	5	18	4	20	4	23	3	68	4
WB	0.61745	94⇃	94⇃	88	45	106⇃	36⇃	65⇃	17⇃	81	17	133⇃	23⇃	132⇃	14⇃	-	-
BICOP1	0.59988	82	82	67	34	106	36	139	35	206⇃	42⇃	322⇃	54⇃	-	-	-	-
BICOP2	0.27667	353⇃	353⇃	338⇃	169⇃	-	-	356⇃	89⇃	-	-	-	-	-	-	-	-
TRICOP	45.3701	13	13	16	8	15	6	21	6	16	4	17	3	19	2	33	2
WP	$3.517 \cdot 10^{18}$	58	58	62	31	66	22	74	19	92	19	103	18	-	-	-	-

7.3 One Shot Optimization Results

The Hypervolume of the one shot optimization algorithm configuration is given in Table 4. Inspection of this table tells us that the test functions with a high feasibility rate tend to give much better results compared to test functions with a low feasibility rate. This indicates that the constraints are not well fitted after the initial sample and that more adaptive sampling steps lead to better constraint boundary approximation and therefore to better Pareto frontier approximations.

8 CONCLUSION

In this paper a new acquisition function capable of multi-point multi-objective optimization is introduced and implemented together with a constraint handling mechanism. This new acquisition function is plugged in to the SAMO-COBRA algorithm, making it able to propose multiple solutions per iteration. Experiments on a benchmark test set show that with larger batch sizes, a significant number of iterations can be reduced, and therefore a lot of wall clock time can be saved. This is especially interesting in cases where the evaluation of solutions is very time consuming and when they can be evaluated in parallel. The new infill criteria not only gives the possibility to save wall clock time, but also the computational resources and the use of commercial licences can now be more effectively exploited.

Future work will be put into dealing with multi-fidelity optimization problems, and asynchronous function evaluations, to decrease wall clock time even more. Other extensions of the algorithm could be a discrete parameter handling mechanism so that the algorithm can also be used for a mixture of decision variable types.

Table 4: Mean and Standard deviation of hypervolume of the one shot optimization algorithm configuration between the Nadir point and the obtained Pareto frontiers over 11 runs after 80 function evaluations with an initial Halton sample of 40. The results are compared to the result of the original infill criteria with batch size 1 by computing the hypervolume differences in a percentage.

Function	hv	std	Percentage
BNH	4939.6	2	−0.60%
CEXP	3.6507	0.0240	−4.01%
SRN	23649	262	−5.79%
TNK	0.2044	0.0341	−46.18%
CTP1	0.2731	0.0091	−9.30%
C3DTLZ4	1.4308	0.0458	−7.95%
OSY	6144.9	1240.3	−105.52%
TBTD	6007.2	425.2	−34.05%
NBP	768803	4997	−4.00%
DBD	56.812	0.541	−5.60%
SPD	$2.9674 \cdot 10^9$	$3.058 \cdot 10^8$	−85.72%
CSI	5.9929	0.0472	−25.81%
SRD	2855825	61403	−3.37%
WB	0.5601	0.0126	−13.82%
BICOP1	0.4193	0.0482	−52.04%
BICOP2	0.0759	0.0296	−235.84%
TRICOP	47.750	0.798	−3.96%
WP	$3.198 \cdot 10^{18}$	$2.44 \cdot 10^{17}$	−14.98%

ACKNOWLEDGMENTS

Special thanks to C-Job Naval Architects for financing the research.

REFERENCES

[1] Taimoor Akhtar and Christine A Shoemaker. 2019. Efficient multi-objective optimization through population-based parallel surrogate search.

[2] Thomas Back. 1996. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford university press.

[3] Samineh Bagheri, Wolfgang Konen, Michael Emmerich, and Thomas Bäck. 2017. Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. *Applied Soft Computing* 61 (2017), 377–393. https://doi.org/10.1016/j.asoc.2017.07.060

[4] Slim Bechikh, Lamjed Ben Said, and Khaled Ghedira. 2010. Estimating nadir point in multi-objective optimization using mobile reference points. In *IEEE Congress on Evolutionary Computation.* IEEE, 1–9. https://doi.org/10.1109/CEC.2010.5586203

[5] Torsten Bosse, Nicolas R. Gauger, Andreas Griewank, Stefanie Günther, and Volker Schulz. 2014. One-Shot Approaches to Design Optimzation. In *Trends in PDE Constrained Optimization*, Günter Leugering, Peter Benner, Sebastian Engell, Andreas Griewank, Helmut Harbrecht, Michael Hinze, Rolf Rannacher, and Stefan Ulbrich (Eds.). Springer International Publishing, Cham, 43–66. https://doi.org/10.1007/978-3-319-05083-6_5

[6] Jakob Bossek, Carola Doerr, Pascal Kerschke, Aneta Neumann, and Frank Neumann. 2020. Evolving Sampling Strategies for One-Shot Optimization Tasks. In *Parallel Problem Solving from Nature – PPSN XVI*, Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann (Eds.), Vol. 12269. Springer International Publishing, Cham, 111–124. https://doi.org/10.1007/978-3-030-58112-1_8

[7] Martin D Buhmann. 2003. *Radial basis functions: theory and implementations.* Vol. 12. Cambridge university press.

[8] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. 2007. *Evolutionary algorithms for solving multi-objective problems.* Vol. 5. Springer. 5,10,188–189 pages.

[9] Rituparna Datta and Rommel G Regis. 2016. A surrogate-assisted evolution strategy for constrained multi-objective optimization. *Expert Systems with Applications* 57 (2016), 270–284. https://doi.org/10.1016/j.eswa.2016.03.044

[10] Roy de Winter. 2022. Roy de Winter/Multi-Point-SAMO-COBRA: Release1. https://doi.org/10.5281/zenodo.6461614

[11] Roy de Winter, Philip Bronkhorst, Bas van Stein, and Thomas Bäck. 2022. Constrained Multi-Objective Optimization with a Limited Budget of Function Evaluations. *Memetic Computing* (2022), 1–14.

[12] Roy de Winter, Bas van Stein, Matthys Dijkman, and Thomas Bäck. 2019. Designing Ships Using Constrained Multi-objective Efficient Global Optimization. In *Machine Learning, Optimization, and Data Science*, Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, Renato Umeton, and Vincenzo Sciacca (Eds.). Springer International Publishing, Cham, 191–203.

[13] Kalyanmoy Deb. 2001. *Multi-objective optimization using evolutionary algorithms.* Vol. 16. John Wiley & Sons.

[14] Kalyanmoy Deb, Amrit Pratap, and T Meyarivan. 2001. Constrained test problems for multi-objective evolutionary optimization. In *International conference on evolutionary multi-criterion optimization.* Springer, Springer, 284–298. https://doi.org/10.1007/3-540-44719-9_20

[15] Alexander Forrester, Andras Sobester, and Andy Keane. 2008. *Engineering design via surrogate modelling: a practical guide.* John Wiley & Sons. https://doi.org/10.2514/4.479557

[16] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. 2010. Kriging is well-suited to parallelize optimization. In *Computational intelligence in expensive optimization problems.* Springer, 131–162.

[17] Wenyin Gong, Zhihua Cai, and Li Zhu. 2009. An efficient multiobjective differential evolution algorithm for engineering design. *Structural and Multidisciplinary Optimization* 38, 2 (2009), 137–157. https://doi.org/10.1007/s00158-008-0269-9

[18] Raphael T. Haftka, Diane Villanueva, and Anirban Chaudhuri. 2016. Parallel surrogate-assisted global optimization with expensive functions–a survey. *Structural and Multidisciplinary Optimization* 54, 1 (2016), 3–13. https://doi.org/10.1007/s00158-016-1432-3

[19] John H Halton. 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* 2, 1 (1960), 84–90.

[20] Gideon Hanse, Roy de Winter, Bas van Stein, and Thomas Bäck. 2021. Optimally weighted ensembles for efficient multi-objective optimization. In *International Conference on Machine Learning, Optimization, and Data Science.* Springer, 144–156.

[21] Himanshu Jain and Kalyanmoy Deb. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Trans. Evolutionary Computation* 18, 4 (2014), 602–622. https://doi.org/10.1109/tevc.2013.2281534

[22] Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13, 4 (1998), 455–492.

[23] Jian-Yu Li, Zhi-Hui Zhan, and Jun Zhang. 2021. Evolutionary Computation for Expensive Optimization: A Survey. *International Journal of Automation and Computing* 18 (Oct. 2021), 1–21. http://ijac.xml-journal.net/en/article/id/3b0787b0-c3fd-41bd-9593-da155838a209

[24] Charles A Micchelli. 1986. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive approximation* 2, 1 (1986), 11–22.

[25] Seyedali Mirjalili, Pradeep Jangir, and Shahrzad Saremi. 2017. Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. *Applied Intelligence* 46, 1 (2017), 79–95. https://doi.org/10.1007/s10489-016-0825-8

[26] Giacomo Nannicini. 2019. Performance of hybrid quantum-classical variational heuristics for combinatorial optimization. *Physical Review E* 99, 1 (2019), 013304.

[27] Emre Özkaya and Nicolas R Gauger. 2009. Single-step one-shot aerodynamic shape optimization. In *Optimal control of coupled systems of partial differential equations.* Springer, 191–204.

[28] Michael G Parsons and Randall L Scott. 2004. Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. *Journal of Ship Research* 48, 1 (2004), 61–76. https://doi.org/10.1007/s10489-016-0825-8

[29] Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. 2008. Multiobjective Optimization on a Limited Budget of Evaluations Using Model-Assisted $\mathcal{S}$-Metric Selection. In *International Conference on Parallel Problem Solving from Nature.* Springer, Springer, 784–794. https://doi.org/10.1007/978-3-540-87700-4_78

[30] 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 *International Conference on Parallel Problem Solving from Nature.* Springer, 784–794.

[31] Michael JD Powell. 1992. The theory of radial basis function approximation in 1990. *Advances in numerical analysis* (1992), 105–210.

[32] M. J. D. Powell. 1994. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In *Advances in Optimization and Numerical Analysis.* Springer Netherlands, 51–67. https://doi.org/10.1007/978-94-015-8330-5_4

[33] Frederik Rehbach, Martin Zaefferer, Boris Naujoks, and Thomas Bartz-Beielstein. 2020. Expected Improvement versus Predicted Value in Surrogate-Based Optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20).* Association for Computing Machinery, New York, NY, USA, 868–876. https://doi.org/10.1145/3377930.3389816

[34] Thomas P Scholcz, Tomasz Gornicz, and Christian Veldhuis. 2015. Multi-objective hull-form optimization using Kriging on noisy computer experiments. In *MARINE VI: proceedings of the VI International Conference on Computational Methods in Marine Engineering.* CIMNE, CIMNE, 1064–1077.

[35] Lei Shi, RJ Yang, and Ping Zhu. 2012. A method for selecting surrogate models in crashworthiness optimization. *Structural and Multidisciplinary Optimization* 46, 2 (2012), 159–170.

[36] Bas van Stein, Hao Wang, and Thomas Bäck. 2019. Automatic Configuration of Deep Neural Networks with Parallel Efficient Global Optimization. In *2019 International Joint Conference on Neural Networks (IJCNN).* IEEE, 1–7. https://doi.org/10.1109/IJCNN.2019.8851720

[37] Ryoji Tanabe and Akira Oyama. 2017. A note on constrained multi-objective optimization benchmark problems. In *2017 IEEE Congress on Evolutionary Computation (CEC).* IEEE, IEEE, 1127–1134.

[38] Takashi Wada and Hideitsu Hino. 2019. Bayesian Optimization for Multi-objective Optimization and Multi-point Search. arXiv:1905.02370 [stat.ML]

[39] Hao Wang, Bas van Stein, Michael Emmerich, and Thomas Back. 2017. A new acquisition function for Bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* IEEE, IEEE, 507–512.

[40] Jolan Wauters, Andy Keane, and Joris Degroote. 2020. Development of an adaptive infill criterion for constrained multi-objective asynchronous surrogate-based optimization. *Journal of Global Optimization* 78, 1 (2020), 137–160.