Constrained Bi-objective Surrogate-Assisted Optimization of Problems with Heterogeneous Evaluation Times: Expensive Objectives and Inexpensive Constraints

Julian Blank[0000−0002−2227−6476] and Kalyanmoy Deb[0000−0001−7402−9939]

Computational Optimization and Innovation (COIN) Laboratory
Michigan State University, East Lansing, MI, USA
{blankjul,kdeb}@msu.edu
http://www.coin-lab.org

COIN Report 2020019

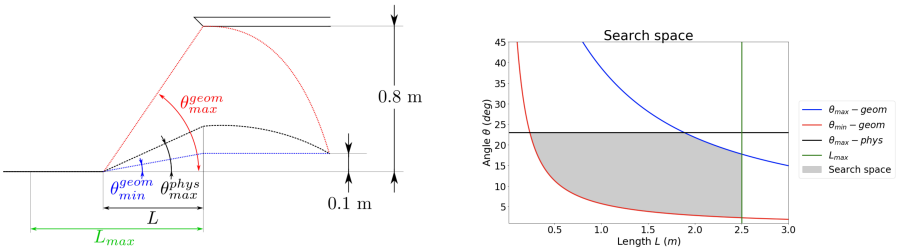
Abstract. In the past years, a significant amount of research has been done in optimizing computationally expensive and time-consuming objective functions using various surrogate modeling approaches. Constraints have often been neglected or assumed to be a by-product of the expensive objective computation and thereby being available after executing the expensive evaluation routines. However, many optimization problems in practice have separately evaluable computationally inexpensive geometrical or physical constraint functions, while the objectives may still be time-consuming. This scenario probably makes the simplest case of handling heterogeneous and multi-scale surrogate modeling in the presence of constraints. In this paper, we propose a method which makes use of the inexpensiveness of constraints to ensure all time-consuming objective evaluations are only executed for feasible solutions. Results on test and real-world problems indicate that the proposed approach finds a widely distributed set of near-Pareto-optimal solutions with a small budget of expensive evaluations.

Keywords: Multi-objective Optimization · Surrogate-assisted Optimization · Inexpensive Constraints · NSGA-II · Evolutionary Algorithm

1 Introduction

Optimization of simulation models is essential and especially important in many interdisciplinary research areas to find optimized solutions for real-world applications. For instance, many engineering design optimization problems [22] require a time-consuming simulation to evaluate the design's performance. A common technique to deal with long-running simulation is to use so-called *surrogate* models or metamodels [1, 23, 18] to represent the original objective and/or constraint functions. This then allows a computationally cheaper optimization procedure

to find potentially good solutions without executing expensive evaluations. However, in real-world problems, a design's feasibility must be checked during the optimization process, as an infeasible solution, no matter how good its objective values are, cannot be accepted. For instance, consider the diffuser inlet design problem [13, 24], shown in Figure 1a. In [20], the diffuser design is modeled by two variables, the length (L) and the angle (θ), to maximize the total pressure ratio and the Mach number at the outlet. The objectives are evaluated using an expensive mesh generation procedure bounding the diffuser's height between 0.1 to 0.8 meters. In turn, these bounds limit the minimum and maximum value of θ for each L. This dependency and some additional constraints result in non-linear constraints that make the shaded region in Figure 1b feasible. The feasible search space obeys geometric and physical laws, which are relatively easy to satisfy in this problem but can not be expressed simply by box constraints on variables. However, the objective of the problem is to maximize the entropy generation rate by the diffuser, which must start from solving Navier-Stokes questions involving turbulent flow and solving the equations numerically with an adaptive step-size procedure. This process is many times more expensive than finding if a solution is feasible. Thus, a generic surrogate-assisted optimization method that treats all objective and constraint functions as equally expensive may not be most efficient for solving such real-world problems.



(a) Geometry [20] (b) Search Space [20]

Fig. 1: Geometric constraints of a diffuser inlet design.

In [25], the optimization of geometrical design parameters in IPM motor topology has been discussed. Apart from box constraints, the ratio between variables is bounded. Also, the authors mention that the simulation software allows running a computationally quick feasibility check to identify overlapping edges, negative lengths, or non-conventional geometric relations that will inevitably produce errors after the solver's start. The availability of such quick feasibility checks – without running the time-consuming simulation – results in a relatively computationally inexpensive constraint evaluation. However, to compute the objective function given by the magnetic flux requires an expensive finite element methodology, thereby making the IPM motor design task as another case of heterogeneous surrogate-based optimization.

Even though computationally expensive objective functions have been studied extensively [18, 9, 26], computationally inexpensive constraints have been

paid a little attention in the literature. It is also noteworthy that there can be other combinations of complexities in evaluating them. For instance, the constraint may be expensive to evaluate in other real-world problems, while the objectives may be relatively quick to compute. In this paper, we restrict ourselves to the heterogeneous problems involving inexpensive constraint evaluation and expensive objective evaluation procedures and belabor the consideration of other scenarios in later studies.

In the remainder of this paper, we first present the related work, which considers mostly the more general case of heterogeneous objective function evaluation. In Section 3, we propose our method for handling inexpensive constraint functions efficiently on an algorithmic level. The performance of the proposed method is evaluated on a variety of constrained bi-objective test problems in Section 4. Conclusions and future works are discussed in Section 5.

2 Related Work

In general, optimization problems with computationally expensive objectives and inexpensive constraints belong to the category of heterogeneously expensive target functions. The question being addressed is how an algorithm can use the heterogeneity to improve its convergence properties. Mostly, unconstrained bi-objective problems have been studied so far, which inevitably results in one objective being computationally inexpensive (cheap) and one being expensive or time-consuming. Since only two target values have mostly been considered, authors also refer to the difference as a *delay* in the evaluation between two objective functions, as a problem parameter.

In 2013, Allmendinger et al. [3] proposed three different ways of dealing with missing objective values caused by such a delay: (i) filling the missing objectives with random pseudo values within the boundaries of the objective space, (ii) adding Gaussian noise to a solution randomly drawn from the population, and (iii) replacing by the nearest neighbor's objective values in the design space being evaluated on all objectives. Authors also proposed different evaluation selection strategies, for instance, based on the creation time of an offspring or a priority score obtained by partial or full non-dominated rank. This study was one of the first addressing delayed objective functions and has laid the foundation for further investigations. In 2015, this study was extended by Allmendinger et al. [2] by proposing new ways of dealing with the heterogeneity based on using the cheap objectives to look at one or multiple generations ahead. The experimental study has revealed that the performance is affected by the amount of delay of the objectives and, thus, that some approaches are more suitable than others depending on the amount of delay.

In 2018, Chugh et al. have proposed HK-RVEA [7] as an extension of K-RVEA [6], which handles objective functions with different latencies. Compared to the original K-RVEA, significant changes are related to the surrogate's training and update mechanism, driven by a single-objective evolutionary algorithm. A comparison regarding bi-objective test problems with previously proposed ap-

proaches [2, 3] showed that this surrogate-assisted algorithm works especially well in cases with low latencies.

Moreover, a trust region-based algorithm that aims to find a single Pareto-optimal solution with a limited number of function evaluations was proposed by Thomann et al. [27]. The search direction for each step is given by the vector between the current best solution and the minimum of the locally fitted quadratic model. The trust region serves as step size and limits the surrogate's underlying error. Results on bi-objective optimization problems indicate the algorithm's capability to convergence to a Pareto critical point.

Wang et al. [28] proposed T-SAEA that uses a transfer learning approach to exploit knowledge gained about the inexpensive objective before evaluating the more expensive one. A filter-based feature selection finds essential variables of the computationally inexpensive objective. Then, these features serve as a carrier for knowledge transfer to the computationally expensive function and help to improve the metamodel's accuracy.

In general, most studies related to computationally expensive optimization problems do not consider constraints or assume them to be expensive as well. Also, existing literature about heterogeneous expensive target functions focuses on a delay of objective functions but *not* constraints. Because computationally inexpensive constraints are a practical issue that needs to be addressed efficiently, it shall be the focus of this paper.

3 Proposed Methodology

In the following, we propose IC-SA-NSGA-II, an inexpensive constraint handling method using a surrogate-assisted version of NSGA-II [10]. First, we describe three different methods to generate a feasible design of experiments, and, second, the outline of the algorithm. Our proposed method makes explicit use of the computationally inexpensive constraint functions and guarantees a solution's feasibility before running the time-consuming simulation of objective functions.

3.1 Design of Experiments (DOE) to Build Initial Model

In surrogate-assisted optimization [18], the so-called Design of Experiments (DOE) needs to be generated to build the initial model(s). The number of initial points $N^{\texttt{DOE}}$ depends on different factors, such as the number of variables or the complexity of the fitness landscape. A standard method frequently used to generate a well-spaced set of points is Latin Hypercube Sampling (LHS) [19]. However, with the availability of an efficient feasibility check, more sophisticated approaches shall be preferred. Therefore, three different methods returning a well-spaced set of *feasible* solutions using inexpensive constraint functions are described.

Rejection Based Sampling (RBS): A relatively simple approach is modifying LHS, which already provides a well-spaced set of solutions, to consider feasibility. Such a feasible and well-spaced set of points can be obtained by using LHS to produce a point set P and rejecting all infeasible solutions to obtain a set of

feasible solutions $P^{(\texttt{feas})}$. Because infeasible solutions have been discarded, the resulting point set does not have the desired number of points ($|P^{(\texttt{feas})}| < N^{\texttt{DOE}}$) and, thus, the process shall be repeated until enough feasible solutions have been found. If the size of the obtained point set exceeds $N^{\texttt{DOE}}$, a subset is selected randomly.

Niching Genetic Algorithm (NGA): The sampling process itself can be seen as an optimization problem where the objective is given by the constraint violation $f(\mathbf{x}) = \text{cv}(\mathbf{x})$, which takes a value zero for a feasible solution $\mathbf{x}$, and a positive value proportional to the sum of normalized constraint violation of all constraints if $\mathbf{x}$ is infeasible. Such an objective function results in a multi-modal optimization problem where a *diverse* solution set with objective values of zeros shall be found. One type of algorithm used for multi-modal problems is niching-based genetic algorithms (NGA) where the diversity is ensured by an ϵ-clearing based environmental survival [21]. For single-objective optimization problems, the ϵ-clearing occurs in the design space and guarantees a distance (usually Euclidean distance is used) from one solution to another. The survival always selects the best performing not already selected or cleared solution and then clears its neighborhood with less than ϵ distance. Thus, the spread of solutions is accomplished by disfavoring solutions in each other's vicinity. Having set the objective to be the constraint violation, a suitable ϵ has to be found. The suitability of a given ϵ depends on the size of the feasible region(s) of the corresponding optimization problem and is not known beforehand. On the one hand, if ϵ is too large, the number of optimal solutions found by the algorithm will not exceed $N^{\texttt{DOE}}$. On the other hand, if ϵ is too small, the solution set's spread has room for improvement. For tuning the hyper-parameter ϵ, we start with $\epsilon = \epsilon_0$ (a number close to 1.0) and execute NGA. If the size of the obtained solution set is less than $N^{\texttt{DOE}}$, we set $\epsilon = 0.9 \cdot \epsilon$ and repeat this procedure until a solution set of at least of size $N^{\texttt{DOE}}$ is found.

Riesz s-Energy Optimization (Energy): The Riesz s-Energy [16] is a generalization of potential energy concept and is defined for the point set $\mathbf{z}$ as

$$U(\mathbf{z}) = \frac{1}{2} \sum_{i=1}^{|\mathbf{z}|} \sum_{\substack{j=1 \\ j \neq i}}^{|\mathbf{z}|} \frac{1}{\left\| \mathbf{z}^{(i)} - \mathbf{z}^{(j)} \right\|^s}, \quad \mathbf{z} \in \mathbb{R}^{n \times M}. \tag{1}$$

where the inverse norm to the power s of each pair of points ($\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$) is summed up. It has already been shown in [5] that Riesz s-Energy can be used to achieve a well-spaced point set by executing a gradient-based algorithm. The restriction made there that all points have to lie on the unit simplex has been replaced by the feasibility check provided by the computationally inexpensive constraint. Thus, a point is only replaced by its successor obtained by the gradient update if the successor is *feasible*. The algorithm's initial point set, which is necessary to be provided, is first tried to be obtained by RBS, and if a sufficient number of feasible solutions could not be found by NGA.

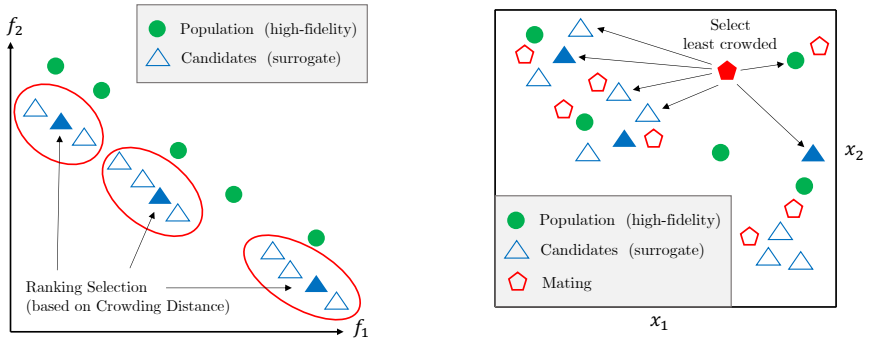
Algorithm 1: IC-SA-NSGA-II: **I**nexpensive **C**onstrained **S**urrogate-**A**ssisted NSGA-II.

Input: Number of Variables n, Expensive Objective Function $f(\mathbf{x})$, Inexpensive Constraint Function $g(\mathbf{x})$, Maximum Number of Solution Evaluations $\texttt{SE}^{\max}$, Number of Design of Experiments $N^{\texttt{DOE}}$, Exploration Points $N^{(\texttt{explr})}$, Exploitation Points $N^{(\texttt{exploit})}$, Number of generations for exploitation k, Multiplier of offsprings for exploration s

/* initialize feas. solutions using the inexpensive function g */
1 $\mathbf{X} \leftarrow \texttt{constrained\_sampling}(\texttt{'energy'}, N^{\texttt{DOE}}, g)$
2 $\mathbf{F} \leftarrow f(\mathbf{X})$
3 **while** $|\boldsymbol{X}| < \mathit{SE}^{\max}$ **do**
  /* exploitation using the surrogate */
4 $\hat{f} \leftarrow \texttt{fit\_surrogate}(\mathbf{X}, \mathbf{F})$
5 $\left(\mathbf{X}^{(\texttt{cand})}, \mathbf{F}^{(\texttt{cand})}\right) \leftarrow \texttt{optimize}(\texttt{'nsga2'}, \hat{f}, g, \mathbf{X}, \mathbf{F}, k)$
6 $\left(\mathbf{X}^{(\texttt{cand})}, \mathbf{F}^{(\texttt{cand})}\right) \leftarrow \texttt{eliminate\_duplicates}(\mathbf{X}, \mathbf{X}^{(\texttt{cand})}, \mathbf{F}^{(\texttt{cand})})$
7 $C \leftarrow \texttt{cluster}(\texttt{'k\_means'}, N^{(\texttt{exploit})}, \mathbf{F}^{(\texttt{cand})})$
8 $\mathbf{X}^{(\texttt{exploit})} \leftarrow \texttt{ranking\_selection}(\mathbf{X}^{(\texttt{cand})}, C, \texttt{crowding}(\mathbf{F}^{(\texttt{cand})}))$
  /* exploration using mating and least crowded selection */
9 $\mathbf{X}', \mathbf{F}' \leftarrow \texttt{survival}(\mathbf{X}, \mathbf{F})$
10 $\mathbf{X}^{(\texttt{mat})} \leftarrow \texttt{mating}(\mathbf{X}', \mathbf{F}', s \cdot N^{(\texttt{explr})})$
11 $\mathbf{X}^{(\texttt{explr})} \leftarrow \texttt{feas\_and\_max\_distance\_selection}(\mathbf{X}^{(\texttt{mat})}, \mathbf{X}^{(\texttt{cand})}, X, g)$
  /* evaluate and merge to the archive */
12 $\mathbf{F}^{(\texttt{explr})} \leftarrow f(\mathbf{X}^{(\texttt{explr})})$; $\mathbf{F}^{(\texttt{exploit})} \leftarrow f(\mathbf{X}^{(\texttt{exploit})})$;
13 $\mathbf{X} \leftarrow \mathbf{X} \cup \mathbf{X}^{(\texttt{explr})} \cup \mathbf{X}^{(\texttt{exploit})}$
14 $\mathbf{F} \leftarrow \mathbf{F} \cup \mathbf{F}^{(\texttt{explr})} \cup \mathbf{F}^{(\texttt{exploit})}$
15 **end**

3.2 IC-SA-NSGA-II Algorithm

The outline of IC-SA-NSGA-II is shown in Algorithm 1. The initial design of experiments of size $N^{\texttt{DOE}}$ are obtained by the proposed initialization method based on Riesz s-Energy and then evaluated by executing the expensive simulation $f(\mathbf{x})$ (Lines 1 and 2). Afterwards, while the number of solution evaluations $\texttt{SE}^{\max}$ is not exceeded (Line 3) the algorithm continues to generate $N^{(\texttt{exploit})}$ solutions derived from the surrogate for exploitation and $N^{(\texttt{explr})}$ solutions obtained by mating and a distance-based selection for exploration. The exploitation starts with fitting surrogate model(s) which results in the approximation function $\hat{f}$ (Line 4). Using the surrogates $\hat{f}$ and the computationally inexpensive function g the optimization is continued or in other words simulated assuming $f = \hat{f}$ for k more generations (Line 5). From the last simulated generation, the candidates $\mathbf{X}^{(\texttt{cand})}$ and $\mathbf{F}^{(\texttt{cand})}$ are extracted from the optimum, and duplicates with respect to $\mathbf{F}$ are eliminated (Line 6). This ensures $\mathbf{F}^{(\texttt{cand})}$ to consist of only non-dominated solutions with respect to $\mathbf{F}$. From $\mathbf{X}^{(\texttt{cand})}$ only $N^{(\texttt{exploit})}$ solutions



(a) Exploitation: Select solutions from the candidates set obtained by optimizing on the surrogate.

(b) Exploration: Select from a solution set obtained through evolutionary operators by maximizing the distance to existing solutions and candidates.

Fig. 2: The two steps in each iteration: exploitation and exploration.

are chosen for expensive evaluation by executing ranking selection [15] in each cluster where the ranking is based on the crowding distance in $\mathbf{F}^{(\texttt{cand})}$. Figure 2a illustrates the exploitation procedure of a population with five solutions (circles). The algorithm found ten candidate solutions (triangles) by optimizing the surrogate and the inexpensive constraint functions. In this example, $N^{(\texttt{exploit})} = 3$ and, thus, the K-means algorithm is instantiated to find three clusters. From each cluster, the solutions obtained by ranking selection based on the crowding distance are assigned to $\mathbf{X}^{(\texttt{exploit})}$.

Besides the exploitation, some exploration is essential to be incorporated into a surrogate-assisted algorithm. The exploration is based on the evolutionary recombination of NSGA-II with post-filtering based on the distance in the design space (Line 9 to 11). First, the environmental survival is executed because the mating should not be based on the archive $\mathbf{X}$ but instead on a subset of more promising solutions $\mathbf{X}'$. Second, mating takes place to produce $s \cdot N^{(\texttt{explr})}$ solutions $\mathbf{X}^{(\texttt{mat})}$ and, third, the set of *feasible* solutions from $\mathbf{X}^{(\texttt{mat})}$ being maximally away from $\mathbf{X}$ and $\mathbf{X}^{(\texttt{cand})}$ are assigned to $\mathbf{X}^{(\texttt{explr})}$. Figure 2b demonstrates this explorative step more in detail. All infeasible solutions generated through mating have already been eliminated. The solution with the maximum distance to others is selected, which represents the least crowded solution with respect to $\mathbf{X}$ and $\mathbf{X}^{(\texttt{cand})}$. The exploration step purposefully chooses solutions not suggested by the surrogate and helps to escape from local optima if necessary. After selecting the first solution with the maximum distance to others, the solution is marked as selected and now considered in the distance calculations for the second iteration in the selection procedure. Finally, the infill solutions $\mathbf{X}^{(\texttt{exploit})}$ and $\mathbf{X}^{(\texttt{explr})}$ are evaluated on the expensive objective functions f and merged with $\mathbf{X}$ and $\mathbf{F}$ (Line 12 to 14).

In our implementation, we set the number of the initial design of experiments to $N^{\texttt{DOE}} = 11n - 1$ where n is the number of variables. Moreover, we simulate

NSGA-II for $k = 20$ generations with 100 offsprings each generation on the surrogate model. We have used the NSGA-II implementation available in pymoo [4] for optimization and the Radial Basis Function (RBF) implementation with a cubic kernel and linear tail available in pySOT [14] for the surrogate model. In each iteration five new solutions are evaluated using the expensive objective function, where $N^{(\texttt{exploit})} = 3$ and $N^{(\texttt{explr})} = 2$. Furthermore, we set the multiplier of offsprings during exploration to $s = 100$. Moreover, it is worth pointing out that we compare our method with SA-NSGA-II, which does not assume the constraints are inexpensive but follows overall the same procedure. In contrast to IC-SA-NSGAII, it uses regular Latin Hypercube Sampling for the initial design of experiments and fits a surrogate of the constrained function(s) to evaluate feasibility.

4 Results

The proposed method uses the inexpensiveness of the constraint function(s), and, thus, the performance on *constrained* multi-objective optimization problems shall be evaluated. In this paper, we focus on bi-objective problems with up to ten constraint functions. In contrast to the constraint function, we treat all objectives to be computationally expensive.

To evaluate the algorithm's performance, we use the CTP test problems suite [12], which has been designed to address constraints of varying difficulty. Moreover, the performance on other bi-objective constrained optimization problems frequently used in the literature such as OSY [11], TNK [11], SRN [11], C2DTLZ2 [17], C3DTLZ4 [17], and Car Side Impact (CAR) [17] shall be evaluated. The number of solution evaluations $\texttt{SE}^{\max}$ is kept relatively small to mimic the evaluation budget of time-consuming simulations. In the following, first, the performance of methods proposed to generate a feasible solution set for the design of experiments are *visually* analyzed, and, second, the algorithm's performance on test problems is discussed.

In Figure 3, the results of Rejection Based Sampling (RBS), Niching GA (NGA), and Riesz s-Energy (Energy) are shown. Compared to RBS and NGA, Energy obtains a very uniform and well-spaced point set in the inside of the feasible region across all problems. Also, it is worth noting that points on the constraint boundary are found, which can be very valuable to start with because, in practice, optima frequently lie on constraint boundaries. For the purpose of visualization, the CTP8 problem with two-variables (and nine feasible disconnected regions) has been investigated. All methods were able to obtain more than one feasible solution in all regions.

Table 1 lists the median values (obtained from 11 runs) of the Inverted Generational Distance (IGD) [8] indicator of 14 constrained bi-objective optimization problems. The obtained results have been normalized with respect to the ideal and nadir point of each problem's True front. The best performing method and other statistically similar methods (Wilcoxon rank test, $p = 0.05$) are marked in bold. Besides IC-SA-NSGA-II, we ran a more steady-state version of NSGA-II with five offsprings in each generation and an initial population of size $11n - 1$

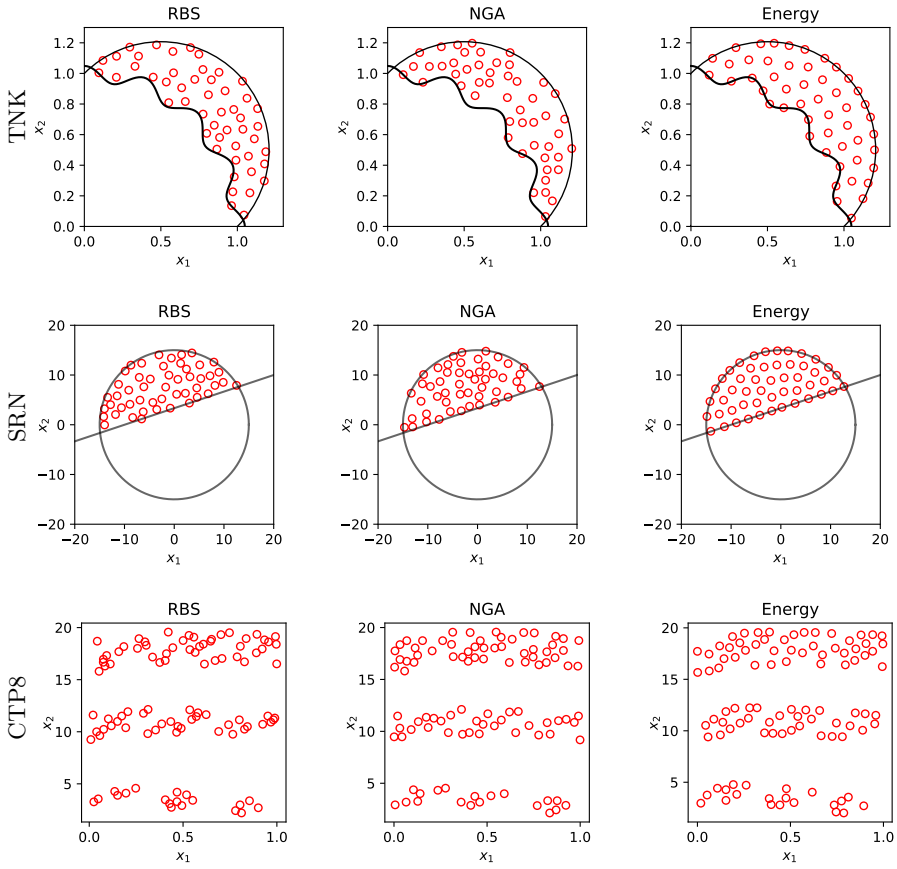


Fig. 3: Sampling the design of experiments only in the feasible space using Rejection-Based Sampling (RBS), Niching Genetic Algorithm (NGA) and Energy Method.

sampled by LHS. Moreover, SA-NSGA-II, which is our proposed method without the modifications made to exploit the availability of inexpensive constraints. Clearly, IC-SA-NSGA-II outperforms the other approaches for most of the problems. For 11 out of 14 optimization problems, our proposed method shows the best performance significantly; for two problems (CTP1 and SRN), SA-NSGA-II performs statistically similar; and for one problem (OSY), SA-NSGA-II shows slightly better results. NSGA-II is not able to find a near-optimal set of solutions with the limited $\texttt{SE}^{\max}$ for any of the selected problems.

Figure 4 shows the obtained solution set for each method of representative runs for CTP2, CTP4, CTP8, C3DTLZ4, TNK, and OSY, for which a well-converged and well-diversified non-dominated solutions are found with 200 to 500 solution evaluations. For the difficult problem CTP2, our proposed method converges near the true optima, which lies on the constraint boundary. Similarly, for CTP8, where nine feasible islands for which three contain the Pareto-

optimal set exist and, thus, a good exploration of the search space is needed. For C3DTLZ4, IC-SA-NSGA-II has obtained a better diversity in the solution set than SA-NSGA-II.

Table 1: The median normalized Inverted Generational Distance (IGD) values out of 11 runs for NSGA-II, SA-NSGA-II and IC-SA-NSGA-II on constrained bi-objective optimization problems. The best performing method and other statistically similar methods are marked in bold.

Problem	Variables	Constraints	$\texttt{SE}^{\max}$	NSGA-II	SA-NSGA-II	IC-SA-NSGA-II
CTP1	10	2	200	3.6399	**0.0237**	**0.0196**
CTP2	10	1	200	1.4422	0.1721	**0.0173**
CTP3	10	1	200	1.2282	0.2752	**0.0357**
CTP4	10	1	400	0.8489	0.3969	**0.0736**
CTP5	10	1	400	0.7662	0.1145	**0.0139**
CTP6	10	1	400	7.7155	0.1909	**0.0117**
CTP7	10	1	400	1.5517	0.0164	**0.0032**
CTP8	10	2	400	11.6452	0.5963	**0.0074**
OSY	6	6	500	0.4539	**0.0273**	**0.0381**
SRN	2	2	200	0.0263	**0.0112**	**0.0108**
TNK	2	2	200	0.1281	0.0200	**0.0092**
C2DTLZ2	12	1	200	0.3787	0.1185	**0.0484**
C3DTLZ4	7	2	200	0.2622	0.1210	**0.0481**
CAR	7	10	200	0.2362	0.0168	**0.0147**

5 Conclusions

The optimization of computationally expensive optimization problems has become more important in practice. Often such problems have physical or geometrical constraints that are relatively computationally inexpensive and can be formulated in equations without running the simulation. To solve these kinds of problems, we have proposed IC-SA-NSGA-II, a surrogate-assisted NSGA-II, which efficiently handles inexpensive constraint functions in the initial design of experiments as well as in each iteration. We have tested our proposed method on 14 constrained bi-objective optimization problems, and our results indicate that efficiently handling the inexpensive constraints helps to converge faster.

This paper has focused on solving constrained optimization problems with inexpensive constraint and expensive objective functions with very limited function calls. However, some other heterogeneous problems with different time scales of objective and constraint evaluations must be considered next. For instance, the constraints can be even more time-consuming than the objective function, or the objectives and constraints can have different time scales within themselves. After addressing such cases, there is a need for developing a unified algorithm for generic heterogeneous optimization problems. Moreover, studies about heterogeneously expensive optimization problems have so far been limit to bi-objective

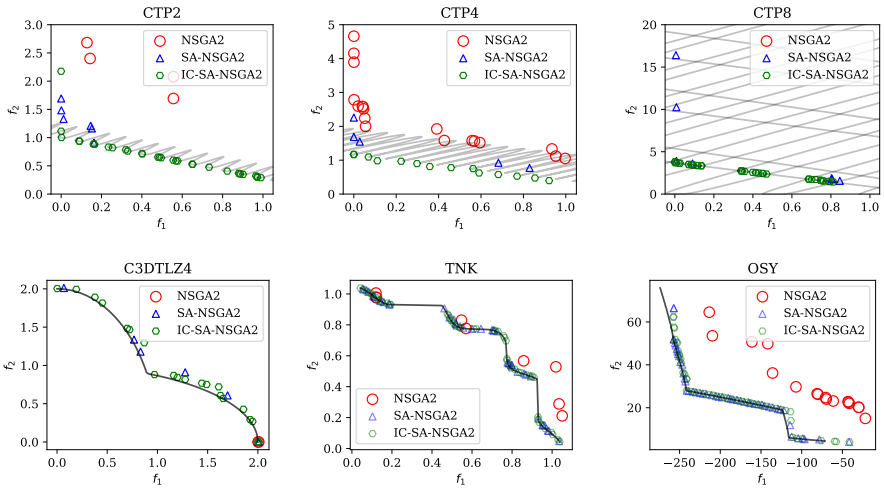


Fig. 4: Solutions in the objective space of representative runs for CTP2, CTP4, CTP8, C3DTLZ4, TNK, and OSY.

optimization problems. This paper has started to add some more complexity by adding constraint functions. However, the effect of heterogeneity for many-objective optimization problems shall provide more insides into exploiting the discrepancy of evaluation times and asynchronicity.

References

1. Ahrari, A., Blank, J., Deb, K., Li, X.: A proximity-based surrogate-assisted method for simulation-based design optimization of a cylinder head water jacket. Engineering Optimization **0**(0), 1–19 (2020)
2. Allmendinger, R., Handl, J., Knowles, J.: Multiobjective optimization: When objectives exhibit non-uniform latencies. European Journal of Operational Research **243**(2), 497 – 513 (2015)
3. Allmendinger, R., Knowles, J.: 'Hang on a minute': Investigations on the effects of delayed objective functions in multiobjective optimization. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) Evolutionary multi-criterion optimization. pp. 6–20. Springer Berlin Heidelberg (2013)
4. Blank, J., Deb, K.: Pymoo: Multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020)
5. Blank, J., Deb, K., Dhebar, Y., Bandaru, S., Seada, H.: Generating well-spaced points on a unit simplex for evolutionary many-objective optimization. IEEE Transactions on Evolutionary Computation
6. Chugh, T., Jin, Y., Miettinen, K., Hakanen, J., Sindhya, K.: A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. IEEE Transactions on Evolutionary Computation **22**(1), 129–142 (2018)
7. Chugh, T., Allmendinger, R., Ojalehto, V., Miettinen, K.: Surrogate-assisted evolutionary biobjective optimization for objectives with non-uniform latencies. In:

Proceedings of the genetic and evolutionary computation conference. pp. 609–616. GECCO '18, Association for Computing Machinery, New York, NY, USA (2018)
8. Coello Coello, C.A., Reyes Sierra, M.: A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004: Advances in artificial intelligence. pp. 688–697. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
9. Deb, K., Hussein, R., Roy, P.C., Toscano-Pulido, G.: A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation **23**(1), 104–116 (2019)
10. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. Trans. Evol. Comp **6**(2), 182–197 (Apr 2002)
11. Deb, K., Kalyanmoy, D.: Multi-objective optimization using evolutionary algorithms. John Wiley & Sons, Inc., USA (2001)
12. Deb, K., Pratap, A., Meyarivan, T.: Constrained test problems for multi-objective evolutionary optimization. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) Evolutionary multi-criterion optimization. pp. 284–298. Springer Berlin Heidelberg (2001)
13. Djebedjian, B.: Two-dimensional Diffuser Shape Optimization (Dec 2004)
14. Eriksson, D., Bindel, D., Shoemaker, C.A.: pySOT and POAP: An event-driven asynchronous framework for surrogate optimization. arXiv:1908.00420 (2019)
15. Goldberg, D.E.: Genetic Algorithms for Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley (1989)
16. Hardin, D., Saff, E.: Minimal Riesz energy point configurations for rectifiable d-dimensional manifolds. Advances in Mathematics **193**(1), 174 – 204 (2005)
17. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. IEEE Transactions on Evolutionary Computation **18**(4), 602–622 (Aug 2014)
18. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1**(2), 61 – 70 (2011)
19. McKay, M.D., Beckman, R.J., Conover, W.J.: Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979)
20. Perez, J.L.: Genetic algorithms applied in Computer Fluid Dynamics for multiobjective optimization. Bachelor senior thesis, University of Vermont (2018)
21. Pétrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: IEEE 3rd ICEC'96. pp. 798–803 (1996)
22. Rao, S.S.: Engineering optimization: Theory and practice. Wiley (2019)
23. Roy, P.C., Hussein, R., Blank, J., Deb, K.: Trust-region based multi-objective optimization for low budget scenarios. In: EMO 2019 - 10th International Conference, East Lansing, MI, USA, March 10-13, 2019, Proceedings. pp. 373–385 (2019)
24. Schmandt, B., Herwig, H.: Diffuser and Nozzle Design Optimization by Entropy Generation Minimization. Entropy **13**, 1380–1402 (Jul 2011)
25. Stipetic, S., Miebach, W., Zarko, D.: Optimization in design of electric machines: Methodology and workflow. In: 2015 Intl. Aegean Conference on Electrical Machines Power Electronics (ACEMP). pp. 441–448 (2015)
26. Stork, J., Friese, M., Zaefferer, M., Bartz-Beielstein, T., Fischbach, A., Breiderhoff, B., Naujoks, B., Tušar, T.: Open issues in surrogate-assisted optimization. In: Bartz-Beielstein, T., Filipič, B., Korošec, P., Talbi, E.G. (eds.) High-performance simulation-based optimization, pp. 225–244. Springer (2020)

27. Thomann, J., Eichfelder, G.: A trust-region algorithm for heterogeneous multiobjective optimization. SIAM Journal on Optimization **29**(2), 1017–1047 (2019)
28. Wang, X., Jin, Y., Schmitt, S., Olhofer, M.: Transfer learning for gaussian process assisted evolutionary bi-objective optimization for objectives with different evaluation times. In: Proceedings of the 2020 genetic and evolutionary computation conference. pp. 587–594. GECCO '20, ACM, New York, NY, USA (2020)