Noname manuscript No.
(will be inserted by the editor)

Handling Constrained Multi-objective Optimization Problems With Heterogeneous Evaluation Times: Proof-of-Principle Results

Julian Blank · Kalyanmoy Deb

COIN Report Number 2021018

Received: date / Accepted: date

Abstract Most real-world optimization problems consist of multiple objectives to be optimized and multiple constraints to be satisfied. Moreover, the performance assessment of the objective and constraints often requires running different software packages separately along with evaluating mathematically defined functions with significantly different (heterogeneous) computing times. A single software package may compute a functional group of objectives and constraints as a block, thereby creating a complicated computing pattern in evaluating a single population member. While much effort has been made to handle computationally expensive functions in the literature, little attention has been paid to handle heterogeneously expensive optimization problems. This paper focuses on efficient and adaptive ways to schedule an evaluation of different functional groups of objectives and constraints by utilizing the potential of offspring population members in terms of their multi-objective ranks, the accuracy of low-fidelity models to predict their function values, and their computing times. Results on a number of unconstrained and constrained two and three-objective optimization problems show significantly better convergence than non-heterogeneous methods. This proof-of-principle study must now be extended by taking full advantage of surrogate-assisted optimization methods to propose more comprehensive and practical optimization algorithms.

Julian Blank
Michigan State University
E-mail: blankjul@egr.msu.edu

Kalyanmoy Deb
Michigan State University
E-mail: kdeb@egr.msu.edu

Keywords Heterogeneously Expensive Functions · Multi-objective Optimization · Constrained Optimization

1 Introduction

Many real-world optimization problems require the consideration of multiple conflicting objectives to reflect the complexity of the application [15]. Additionally, the satisfaction of constraints is necessary to guarantee the solution found by the optimizer is indeed a feasible solution [30]. Mathematically, an optimization problem can be defined by

$$\begin{aligned} \textbf{Minimize} \quad & f_m(\mathbf{x}), && \forall m \in 1, \ldots, M, \\ \textbf{subject to} \quad & g_j(\mathbf{x}) \leq 0, && \forall j \in 1, \ldots, J, \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, && \forall i \in 1, \ldots, N, \end{aligned} \tag{1}$$

where $\mathbf{x}$ are the variables to optimize, $x_i^{(L)}$ and $x_i^{(U)}$ are the lower and upper bounds for the i-th variable, f_m is the m-th objective and g_j the j-th constraint function. For brevity, no equality constraints are considered here. The above mathematical description makes it apparent that assessing the performance of one design (solution) $\mathbf{x}$ requires evaluating a set of functions: M objective and J constraint functions. This results in evaluating a total of $M + J$ functions, from here on, referred to as *target* functions.

Because of the multi-disciplinary nature real-world problems, some of the target functions (a functional group) may require calling a single third-party software, running a simulation [5,34], or other computing-intensive tasks [25,29]. Depending on the software being used, the performance assessment might become fairly time-consuming, for instance, a couple of hours or even

days [1]. A typical practical optimization problem may have two to five such functional groups which must be independently called and evaluated. In addition, there may be certain simplistic target functions, which are mathematically defined and are much quicker to compute than the software or simulation codes.

When optimizing computationally expensive functions, special attention needs to be paid to the limited solution evaluation budget. Over the last decades, researchers have predominantly used surrogate-assisted optimization methods, where an interpolation or approximation model of the computationally expensive target function is utilized during optimization [21]. Most existing surrogate-assisted algorithms assume that the values of *all* target functions (objectives and constraints) are evaluated within one computing job and become available only at the end of the most expensive target evaluation.

A point-based optimization method requires a single new solution to be evaluated at a time to complete an iteration. Thus, the possibilities of exploiting the heterogeneity in the evaluation time of different target functions are limited. However, for a population-based optimization algorithm, such as a generational evolutionary computation (EC) algorithm, a set of offspring population members must be evaluated to proceed to the next generation. It is intuitive to realize that if some target functions are relatively quick to compute in contrast to others, the partial evaluation of population members can be utilized to determine if a population member needs to go through with more expensive target function evaluations. Thus, there is a need for building an evaluation plan for handling heterogeneous target functions for saving computational time, specifically in a population-based optimization algorithm. This is the main crux of this paper.

The majority of the surrogate-assisted optimization methods create new infill solutions based on optimization of the surrogate models. This is useful in its own right in hopefully finding good solutions in a quick computational time. However, since infill solutions are to be evaluated fully for all target functions before they can be used to update surrogate models or proceed with the algorithm, the ideas must be modified for heterogeneous target functions to harness the full advantage of quick partial evaluation of them. Because neither independently computable nor heterogeneously expensive functions have been a major focus of research in the past, barring a few studies, [8,3,2], researchers and practitioners remain using existing optimization methods by letting the optimizer wait until the calculation of all targets has finished. In such a case, the most time-consuming target function determines the waiting time for a solution to be evaluated entirely [3]. The waiting is caused by the optimization method not being capable of processing partial information and results in unused time slowing down the convergence. Other challenges, such as managing computational resources, software licenses, or dealing with hardware or software failures, must be addressed when optimizing real-world problems.

Motivated by such practical optimization challenges, this paper investigates the optimization of independently computable and heterogeneously expensive target functions. The heterogeneity raises a few pragmatic questions about the evaluation procedure itself, taking a close look at first.

The main contributions of this paper can be summarized as follows.

(i) For the first time, we provide a systematic evaluation procedure of a solution set with multiple independent targets and target groups for heterogeneously expensive problems. This includes splitting jobs responsible for the granularity of information and the job scheduling, determining when a piece of information is available.

(ii) Most existing studies address functions with varying evaluation times only considered unconstrained bi-objective or constrained single-objective problems. We investigate a more general case of constrained multi-objective optimization problems and identify the challenges of exploiting independently computable and heterogeneously expensive functions. From this analysis, two essential questions emerge: In what order shall an algorithm evaluate the targets, and when shall a solution be eliminated despite being partially evaluated.

(iii) We propose a constrained multi-objective evolutionary algorithm that exploits the existence of heterogeneously expensive functions. The probability of a solution's survival is based on repeatedly performing an environment selection with different amounts of error noise. The amount of noise depends on the observed prediction error of surrogates in the past. The order of target evaluations is defined by the error that has been made by predicting the survival probability and the evaluation time of the target.

(iv) The proposed idea applies to all kinds of evolutionary algorithms incorporating an (environmental) survival or deciding if a solution will become a part of the population or be discarded, thereby avoiding expensive target function computations. Moreover, it considers groups of targets (for instance, the first objective and third constraint) to be always evaluated together, which is a requirement often occurring in practice.

In the remainder of this paper, we first discuss related work of heterogeneously expensive optimization and its challenges. Then, we discuss the evaluation procedure, including the computing jobs and their scheduling in Section 3. In Section 4, we propose an evolutionary algorithm exploiting heterogeneously expensive objectives and constraints that employs a survival under uncertainty to determine the probability of a solution being promising to be kept. A comparison of the proposed framework for unconstrained and constrained multi-objective algorithms is provided in Section 5. Finally, conclusions are drawn in Section 6.

2 Related Work

Some effort has been made in the past to investigate the heterogeneous expensiveness of target functions. Mostly, bi-objective problems with no constraints have been considered so far. This implies one objective being computationally inexpensive (cheap) and one being expensive. Since only two target functions are considered, authors also refer to the difference as a *delay* in evaluating the objective functions.

The first paper directly addressing heterogeneously expensive objectives was published by Allmendinger et al. [3] in 2013. The authors have proposed three different ways of dealing with missing an objective value caused by such a delay. First, the missing objective value can be filled by randomly drawing pseudo values in the boundaries of the objective space (random). Second, some Gaussian noise is added to the corresponding objective value of a randomly chosen individual (being evaluated on all objectives) and assigned (noise-based). Third, the missing value is replaced by the nearest neighbor's objective values – which can be interpreted as fitness approximation – in the design space being evaluated on all objectives (fitness inheritance). Moreover, for the evaluation selection, the authors have proposed to select always the most recently generated offsprings (sweep selection) or to select them based on a priority score obtained by full or partial non-dominated rank (priority selection). The results indicate that the fitness-inheritance-based pseudo value assignment combined with the sweep selection performs the best.

This initial study has been extended by a number of schemes for handling the delay of an objective function [2]. The authors proposed four different approaches: wait for all objectives to be finished (Waiting); optimize the cheap objective first and evaluate the expensive objective for the optima found (Fast-First); use the cheap objective to look ahead at possible promising offsprings each generation (Brood Interleaving); incorporate even more selection pressure for the expensive objective evaluations by running a single-objective optimization algorithm on the cheap objective (Speculative Interleaving); the experimental study revealed that the performance is affected by the amount of delay for the objective. Speculative Interleaving turned out to perform well when the termination criterion is based on a shorter time limit, and the delay of the objectives is rather significant. Unsurprisingly, the Fast-First strategy outperformed other methods when the objectives were highly positively correlated. The authors also found out that Waiting and Brood Interleaving became increasingly competitive with a longer running time of the algorithms.

In 2018 Chugh et al. have proposed HK-RVEA [11] an extension of K-RVEA [12] which can handle two objective functions with different latencies. Moreover, in contrast to other existing methods, where the expensive objective is predicted by relatively simple approximation, the authors have used Kriging (also known as Gaussian process), a powerful approximation model frequently used in surrogate-assisted optimization. Significant changes compared to the original K-RVEA are related to the training and update mechanism of the surrogate, driven by a single-objective evolutionary algorithm. A comparison regarding bi-objective test problems with previously proposed approaches [3,2] showed that HK-KRVEA works especially well in cases with low latencies.

Thomann et al. have developed the trust region-based algorithm MHT that employs quadratic approximations for objectives not being evaluated yet [37]. The Tammer-Weidner-functional is used for finding descent directions to make use of the heterogeneity of the objective functions. The trust region limits the surrogate's underlying error and serves as a step size in each iteration. The authors have used the concept of local ideal points given by the minimum of the local quadratic model to calculate the search direction in each step. Because of the limited function evaluation budget and one computationally expensive function, the goal of this initial study was to obtain only a single Pareto-efficient solution. In [36] the same authors have proposed a method that starts from the Pareto-efficient solution found by MHT and attempts to explore the neighborhood of the solution further to cover the whole or at least parts of the Pareto-front.

A surrogate-assisted approach called Tr-SAEA has been proposed by Wang et al. for heterogeneously expensive bi-objective problems [38,39]. Inevitably, the existence of a computationally inexpensive and an expensive one quickly leads to knowledge asymmetry. Thus, the authors propose a transfer learning scheme within a surrogate-assisted evolutionary algorithm to transfer

knowledge from the fast objective to the slower one. The transfer is achieved by fitting models where knowledge about the variables and the fast objective serve as an input. The approach has shown to be more robust to varying levels of latency and correlation between the objectives.

Another informative resource about the state-of-the-art of heterogeneous objectives and future research can be found in [4]. The article focuses on unconstrained multi-objective optimization with heterogeneous objective functions. Heterogeneity is discussed in a general manner with a focus on the heterogeneity of the evaluation times. The authors give an overview of recent developments and possible gaps in this research direction.

In [31] independently computable functions in constrained single-objective optimization have been investigated. The authors have proposed eight different constraint handling techniques by combining the ranking of infeasible/feasible solutions, the evaluation type, and the constraint violation aggregation function. In this first study, the sequence in which the constraints are evaluated is determined randomly. Results have shown that this can already significantly improve the convergence of an optimization algorithm. Later on, the work has been extended by incorporating a feasibility relaxation mechanism to permit constraint evaluation for potentially important solutions close to the constraint boundary and by using the feasibility ratio to determine the sequence for constraint evaluation [32]. By ordering the constraints based on the likelihood of violation, computational resources can be saved by stopping to evaluate a solution further as soon as the first violating constraint is discovered. Instead of using only knowledge of the feasibility ratio of constraints gathered from the past, a surrogate for predicting a solution's likelihood to violate a specific constraint is used in [33]. Other novelties presented by the authors are a modified infeasibility-driven ranking for ordering the partially evaluated solutions and an adaptive switching between partial and complete evaluation. Whereas the ranking is essential to give the potentially infeasible solution a chance to survive, the switching guarantees a minimum amount of entirely evaluated solutions.

A recent study [8] has considered constrained bi-objective heterogeneous problems. The study assumed that the problem consists of two functional target groups: computationally expensive objectives and computationally inexpensive constraints. This is probably one of the simplest cases, as every solution can be tagged as feasible or infeasible quickly before deciding to compute the expensive objective values. Results have shown that using a surrogate for predicting objectives and exploiting the inexpensiveness of constraints can significantly improve the performance of an optimization method.

Besides publications directly addressing heterogeneously expensive objectives, the connection to related research directions shall be discussed. One way of addressing the expensive objective is approximating the value with a surrogate before doing the time-consuming evaluation. This introduces a low-fidelity evaluation (using the approximation model) with an underlying prediction error and a high-fidelity model (time-consuming but without any error) for the corresponding objective. Having an objective function with different fidelity levels is also known as multi-fidelity optimization [22]. However, in contrast to heterogeneously expensive problems addressed in this paper, in multi-fidelity optimization, not only one but multiple functions with often different expenses exist for the *same* target. Moreover, the existence of multiple independently computable target functions requires to think about the design of *distributed* and *asynchronous* algorithms [37]. Distributing the evaluation of a set of solutions and their objectives and constraints on different computing nodes causes asynchronicity. An implementation has to address the asynchronicity, for instance, by waiting for all information necessary to obtain and returning the results – the most common implementation in algorithms – or by processing asynchronous events and thus partial evaluations. Furthermore, the situation of having partially evaluated solutions is in a way related to not considering some objectives or targets temporarily. Some studies have addressed the more specific case of removing (redundant) objectives for the whole optimization run [10]. In the case of heterogeneous expensiveness, one usually knows beforehand what objective is expensive and, thus, might less frequently be available for all individuals early on. This changes the viewpoint from what objective is being removed to how to decide whether it is worthwhile to spend time evaluating the time-consuming evaluation of the expansive objective for some individuals or not. Moreover, accessing only partial information of objectives or constraints is related to analyzing a data set with missing values. The occurrence of missing values has been studied thoroughly in data science and machine learning and thus is worth having a look at [6].

To the best of our knowledge, the combination of heterogeneously expensive functions for constrained multi-objective optimization has not been explored yet. Thus, this work shall provide a starting proof-of-principle study for different evaluation times considering both multiple objectives and constraints and should encourage more attention in the near future.

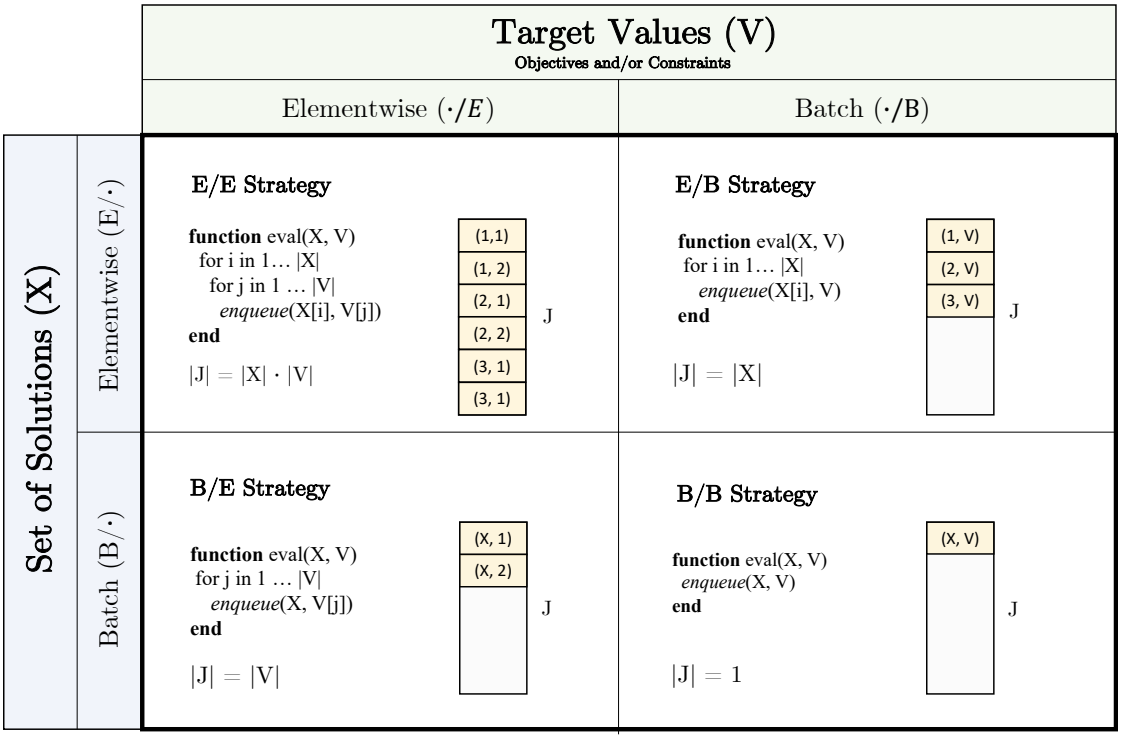


Fig. 1: Strategies for the evaluation procedure considering a set of solutions X and target values V to be calculated.

3 Background

Before different approaches for optimizing problems with heterogeneous target functions are discussed, the evaluation process must be looked at systematically. In general, the evaluation process itself can be split into two interdependent parts: i) the *jobs* being submitted by the algorithm, and ii) the *scheduling* of these jobs. We propose a scheme of different ways for an algorithm to submit jobs regarding a set of solutions and target functions for the former. This defines the *frequency* and *granularity* of information the algorithms retrieves. However, the schedules determine the point of time the algorithm is notified of the job to be finished. The purpose of the scheduler is to decide what job should be executed next. Since resources are commonly limited, a scheduler often uses a job queue or even more sophisticated load balancing techniques. Even though this may sound like a minor implementation detail, for practitioners running optimization methods in a distributed computing environment, this can become crucially important.

3.1 Evaluations Jobs

The frequency and granularity of information the algorithms retrieve opens up new possible ways of asynchronous calculations to efficiently use computing resources. The evaluation of a set of solutions X with multiple target values V can be achieved in several ways. For instance, the algorithm can evaluate each solution in X sequentially by submitting a job for each entry of X separately or a single job containing all solutions in X as a batch. Second, the algorithm can decide what target values each of the jobs should include: Should it be all targets in V that provide complete information about a solution, or just a subset of targets? These choices result in four different ways of packaging the computation jobs defining how the evaluation takes place (see Figure 1). We refer to strategy Y/Z where Y and Z are replaced by E (elementwise) if only a single value and by B (batch) if multiple solutions are chosen. The naming convention is applied analogously to Z with respect to the target values. The B/B strategy is most commonly used, where the algorithm schedules the calculation of all solutions X and target values V only once and retrieves the resulting values when the job has finished. This reduces the number of scheduled jobs to one; however, it does not allow the algorithm to retrieve any intermediate information during evaluation. Contrary to scheduling all jobs at once, the calculation can be split into many small jobs using the E/E strategy. For each solution X_i each target value V_j a separated job is submitted. The resulting number of

jobs $|J|$ is equal to $|X| \cdot |V|$, and the algorithm retrieves a notification whenever each of the jobs has finished. Thus, it has the highest frequency and granularity of information considered in this schema. However, possible calculations that might be shared across the calculation of target values are done repeatedly. The E/B strategy schedules a single solution at a time but multiple target values, which results in a job list of size $|X|$. Since the set of solutions evaluated at a time is usually larger than the target values, this is the strategy with the second most frequency of information. The B/E strategy submits multiple jobs for each target value but does not split up the set of solutions. This results in $|V|$ jobs to be processed. Analogously, the E/E strategy variables necessary for the calculations of multiple target values can not be shared. Nevertheless, if some target values are obtained significantly faster than others, the algorithm is notified without waiting for more computationally expensive target values. It is worth mentioning that a target $v \in V$ can also be a group of targets always being evaluated together. Each partitioning has benefits and drawbacks regarding their flow of information to the algorithm and the concrete implementation.

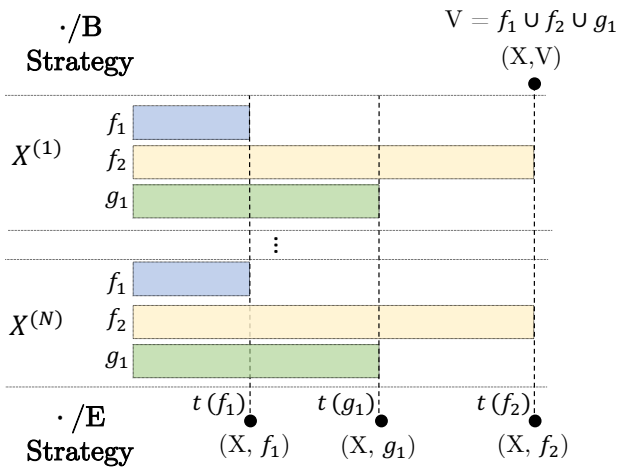


Fig. 2: A comparison of ·/B and ·/E strategies assuming parallel processing of all jobs.

3.2 Job Scheduling

Job partitioning defines the general frequency and granularity of information, but the time of retrieving pieces of information remains unknown without concrete timing. The most straightforward implementation of a scheduler is a FIFO or priority queue, allowing new jobs to be added and retrieving the next job to compute. For now, let us assume all jobs in the queue are processed in *parallel* which requires distributing jobs to at least $|X| \cdot |V|$ workers.

Further, the optimization problem shall consist of two objectives, f_1 and f_2, and one constraint, g_1. We assume different execution times for each target value, $t(f_1)$ for the first and $t(f_2)$ for the second objective, and $t(g_1)$ for the constraint. The evaluation times are defined such that $t(f_1) < t(g_1) < t(f_2)$. Figure 2 demonstrates a possible evaluation procedure for a solution set of size three. Moreover, it shows the information flow for the ·/B (E/B and B/B) and ·/E (B/E and E/E) strategy. The ·/B strategy returns the result given by the union of all target values $V = f_1 \cup f_2 \cup g_1$ exactly once. This implies the algorithm is waiting to obtain full information about all solutions and is idle meanwhile. The waiting time is given by $\max(t(f_1), t(f_2), t(g_1))$ or in general by $\max(t(V_1), \ldots, t(V_{|V|}))$. One single outlier (with a rather larger evaluation time) will increase the overall waiting time and greatly impact the algorithm's overall performance. In this example, the maximum evaluation time is given by $t(f_2)$, which is almost three times larger than $t(f_1)$ (see Figure 2). On the contrary, the ·/E strategy provides some information to the algorithm whenever the calculation of a target value has been finished. Thus, for the three target values the algorithm sends the first notification at $t(f_1)$ with f_1, the second at $t(g_1)$ with g_1 and the third at $t(f_2)$ with f_2. This implies that the optimization algorithm retrieves multiple partial function evaluations at different times, changing the evaluation schedule by step-wise eliminations, thereby making the optimization task more efficient. A batch-wise evaluation of targets would not allow any such advantage.

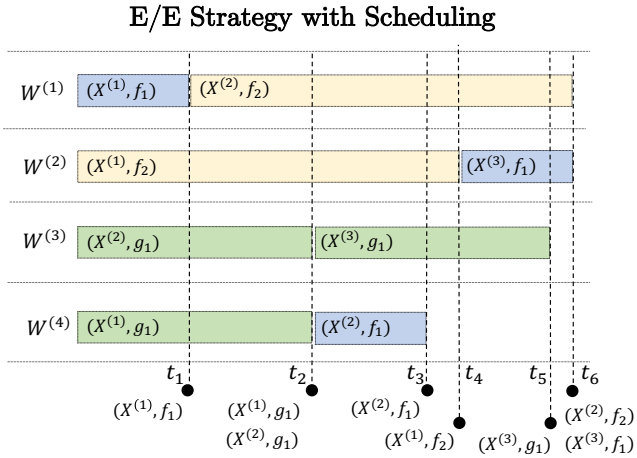


Fig. 3: Job schedule using a queue.

A different execution of jobs might occur, not assuming that a computing unit exists for all jobs simultaneously. Figure 3 illustrates a possible execution of jobs using four instances/worker W_1 to W_4 using the E/E strategy. Because the scheduler decides what jobs to execute next, no prediction about the availability of target values can be made. Moreover, the dispatcher will report different amounts of partial information; thus, an asynchronous algorithm design is desired.

The systematic analysis of the evaluation process shall help researchers think of possible implementations and their impact on the algorithm's design. Based on the above evaluation strategies for dealing with heterogeneously expensive objectives and constraints, we propose a population-based optimization method with B/E evaluation strategy, which makes use of partial evaluation of target functions to carefully eliminate evaluation of expensive targets for potentially inferior population members, thereby making the overall method computationally efficient.

4 Proposed Methodology

Most existing algorithms do not particularly exploit the practical fact that the objectives and constraints are often independently computable. The independent evaluation usually originates from different software packages being executed to determine the performance of a solution. The procedure of assessing the performance of a solution might be fundamentally different for each software package, and thus most of the time, the computing time will vary. This results in an optimization problem with independently heterogeneously expensive objectives and constraints that the optimization method must evaluate and use.

For most practical problems, at least one of the target functions involves a time-consuming evaluation process (we refer here as *high-fidelity* evaluations), thereby causing an optimization run to go on for hours or days. In order to optimize such problems, *surrogate-assisted* optimization methods are prevalent. Using a few high-fidelity solution evaluations, a *surrogate* model (an approximate mathematical function) of each target function is created. Instead of using high-fidelity expensive target functions, surrogate models are usually optimized to find a set of *infill* points. Evaluating a solution using the surrogate models is referred to as *low-fidelity* evaluation. The created infill points are evaluated using high-fidelity target functions, and the surrogate models are updated. Since the optimization task is performed on the surrogate models (low-fidelity evaluation), there is usually a substantial gain in computational time compared to optimization with high-fidelity evaluations. In general, there is a trade-off between gain in computational effort and the resulting accuracy of the obtained solutions in surrogate-assisted optimization methods. If a few high-fidelity solutions are used to build surrogate models, the computational time will be small, but the resulting surrogate model may be inaccurate. Thus, optimizing the surrogate models may result in inferior infill solutions in terms of the high-fidelity target functions. Surrogate-assisted optimization algorithms make a fine balance between the building cost of surrogate models and how extensively they are optimized.

However, it is important to note that this paper does not propose another surrogate-assisted optimization algorithm, nor does it plan to compare the proposed methodology with an existing surrogate-assisted optimization method. Here, we focus on handling heterogeneously expensive target functions within an optimization algorithm utilizing surrogate models. Thus, no effort is made to directly find new infill points using the surrogates. However, instead, models are used to evaluate new solutions to estimate their expected target function values without computing them with high-fidelity evaluation procedures. This is not to say that built surrogate models cannot be exploited further beyond the scope of our proposed methodology, and rather this is something we plan to execute in a subsequent study. Here, we address the presence of heterogeneity in practical target function evaluations and how a population-based algorithm can exploit it to come up with a computationally quick algorithm.

4.1 How to Exploit Heterogeneity of an Optimization Problem?

Most existing population-based algorithms do not assume target functions are independently computable and thus wait to update the old population until all new population members are evaluated for all target functions. If all target functions must be computed simultaneously by a single evaluation procedure, or the evaluation time for all target functions is negligible compared to the desired time for executing an optimization run, no special treatment for any evaluation schedule is necessary.

However, let us consider that not all target functions can be evaluated as a single block of computation, rather independent groups of target functions must be evaluated using different evaluation schemes. Moreover, some groups of target functions require comparatively more time to get evaluated compared to other groups, so there is heterogeneity in the computing efforts. Such problems are predominant in most practical problems, including science and engineering. A practical design

must be evaluated from multi-physics considerations, such as aerodynamics, fluid mechanics, solid mechanics, aesthetics, and others. In such a scenario, if a new design is already evaluated to be worse for certain relatively inexpensive target functions, it can be avoided for further processing within the algorithm and thereby saving computational time by not evaluating expensive target functions. Importantly, such decisions can only be made relative to a set of solutions and cannot be made for a single solution. This is the reason why EC methods are ideal candidates for handling heterogeneous target functions.

For instance, let us assume a genetic algorithm after a few iterations solving a bi-objective optimization problem with one constraint. The algorithm has completed the mating process and created a set of new offspring solutions X to be evaluated for two objectives f_1 and f_2 and constraint g_1. Instead of evaluating all solutions at once for f_1, f_2 and g_1, only one or multiple solutions can be chosen from X to retrieve one target function value at a time, for instance, say, the constraint g_1. Depending on the values of g_1, some solutions can be discarded already because they are found to be infeasible. The more "promising" solutions are kept and continued to be evaluated on the next target f_1. Analogously, some solutions can be eliminated, and only a few are finally send to obtain the values f_2. By processing partial information, not all solutions will be evaluated for all targets, which speeds up the overall evaluation process. Such exploitation of partial information requires answering two elementary questions.

(Q1) Target Order Problem: A relevant question arises: "In what order should the targets be evaluated?". Intuitively, this should depend on the actual evaluation times of target functions and their predicted target values. For the example discussed above, if computational times are having the following relationships: $t(f_1) < t(g_1) < t(f_2)$, then an ordering of the targets by evaluation time should follow least to most expensive, or f_1 followed by g_1, which is then followed by f_2. But the values of the target functions must also play an essential role in deciding on the order of evaluation. Suppose g_1 is found to be positive. In that case, this indicates that the solution is infeasible, and a smart algorithm may decide not to evaluate f_1 and f_2 at all for this solution to save computational time. That is well and good, but there is a problem with the above method. In order to know their relative target function values, the solution has to be evaluated for all target functions. If all functions are already evaluated, there is no need to do any ordering. This rounding argument can be answered by making low-fidelity evaluations of offspring population members using the current surrogate models. But since surrogate models are approximate, each surrogate model may have different prediction accuracy. Thus, the order of their evaluation should depend on a solution's rank in the population-based on its predicted target values (for example, in multi-objective NSGA-II, a combination of non-dominated rank and crowding distance), their accuracy of prediction, and computational time for high-fidelity evaluation of target functions. Despite the importance of g_1 in determining the feasibility of a solution, it may turn out that computation of cheapest objective function (f_1) first is beneficial over the constraint evaluation to determine if the most expensive objective (f_2) needs to be evaluated at all. A combined metric for ordering target functions is provided in Subsection 4.4.

(Q2) Elimination Problem: The next question is as follows: "Under what circumstances should an offspring solution be eliminated and not continued to be evaluated for the remaining targets?". If all targets would be evaluated for all solutions, then the optimization method does not make any use of separately computing solutions. Therefore, some solutions need to be eliminated during the evaluation process. Whereas the order defines what partial information should be made available next, the elimination decides whether a solution is worth keeping and evaluated for more targets. The decision is based on partial information, where some targets are evaluated, their high-fidelity values are available, and a surrogate only predicts others. Another valuable piece of information is the surrogate accuracy derived from the past, which helps to judge how reliable the predictions are.

4.2 Survival Under Uncertainty

An environment selection or survival decides given a set of solutions which one are the fittest and shall survive. Under certainty, numerous survivals have been proposed in the literature, for instance, for unconstrained single-objective genetic algorithms simply a selection based on the objective values [23] or in NSGA-II, survival based on non-dominated sorting and crowding distance [19]. However, most environmental survivals proposed in the evolutionary computation literature assume that the exact values for objectives and constraints are known. The goal of the proposed *probabilistic* survival is to make existing survival procedures applicable under uncertainty. With uncertainty, we refer to the situation that some target values originate from a prediction with an underlying error. Despite the situation where either all targets are based on predictions or all are exact, the survival also needs to handle cases of

Algorithm 1: Probabilistic Survival: Subset Selection under Uncertainty: `prob_surv`(P, Q, V, e, γ)

Input: Population P, Offsprings Q, Uncertain Targets V, Predictions errors e, Iterations γ

```
   /* Repeat the experiment γ times                    */
1  foreach k ← 1 to γ do
2      α_i ← 0   ∀i ∈ (1, ..., |Q|)
3      M ← merge_and_copy(P, Q)
4      foreach i ← 1 to (1, ..., |M|) do
           /* Add noise for uncertain target           */
5          foreach v ∈ V do
6              M[v] = M[v] + N(0, e_v^2)
7          end
8      end
9      M' ← survival(M)
10     foreach i ← 1 to (1, ..., |Q|) do
           /* If survived, increase the counter        */
11         if Q_i ∈ M' then α_i ← α_i + 1 ;
12     end
13 end
   /* Convert survival counts to probabilities         */
14 foreach i ← 1 to (1, ..., |Q|) do  α_i ← α_i/γ ;
15 return α
```

Algorithm 2: Probabilistic Surrogate-Guided Mating: `prob_mating`$(P, Q, S, e, \gamma, \beta)$

Input: Population P, Surrogate S, Predictions errors e, Mating Iterations β, Prob. Surv. Iterations γ

```
   /* Regular mating used by EA                        */
1  Q ← mating()
2  foreach k ∈ (1, ..., β) do
       /* Double the number of offsprings              */
3      Q' ← Q ∪ mating()
       /* Surv. Prob. of each offspring                */
4      α ← prob_surv(P, Q', V, e, γ)
       /* Discard unpromising offsprings               */
5      Q ← top(Q', |Q|, α, 'descending')
6  end
7  return Q
```

mixed uncertainty, where some targets are exact, and some predicted.

The proposed probabilistic survival does not change an existing survival but calls it repeatedly with some introduced error noise for predicted targets. The procedure is illustrated in Algorithm 1. Given a parent population P, offsprings Q, a set of uncertain targets V, the average prediction error e_v of each target $v \in V$, the number of iterations the experiment is repeated γ, and a survival probability α_i for each offspring Q_i. In total, the survival under certainty calls the survival considering certainty exactly γ times. In each iteration, first, the population P and offsprings Q are merged and copied to M. Then for each solution, for each uncertain target $v \in V$ Gaussian noise is added $\mathcal{N}(0, e_v^2)$. The population, M with error noise, is sent to the survival selection, and the survivors are assigned to M'. If solution i has survived and thus is in M', its counter α_i is increased by one. Finally, the survival counters α_i are converted to probabilities by dividing by the number of experiments conducted γ.

With a mix of certain and uncertain targets, the proposed survival might look as follows. Assuming the objective space values f_1 already have been assessed using the high-fidelity evaluation, and $\hat{f}_2$ and $\hat{g}_1$ are predicted by a surrogate with a known prediction error of $e_{\hat{f}_2}$ and $e_{\hat{g}_1}$, respectively. In each iteration, $\hat{f}_2$ is provided with error noise $\mathcal{N}(0, e_{\hat{f}_2}^2)$, as well as the constraint $\hat{g}_1$ with $\mathcal{N}(0, e_{\hat{g}_1}^2)$. The outcome of multiple survival experiments with different amounts of introduced error noise for uncertain targets (f_2 and g_1) lets us derive the probability of a solution to survive in a mixed certain and uncertain environment.

4.3 Probabilistic Surrogate-Guided Mating

Given the prediction and the average error for each target, one can calculate the survival probability α for each offspring originating from mating. Since the mating only uses information about the parents and no predictions, many solutions may have a relatively low survival probability and might be directly discarded. In order to increase the survival probability, this information can be directly used during mating.

In Algorithm 2 the proposed modified mating is demonstrated. The overall idea is based on repeating the original mating procedure `mating`() multiple (β) times by only keeping the most promising offspring solutions. Initially, the offspring population Q is created. Then in each iteration, another offspring population is merged with it to create Q'. For each solution in Q', the survival probability is determined to keep only the solutions most likely to survive. This is achieved by taking the top $|Q|$ solutions from Q' based on a descending sorting by α. After having the process repeated β times, the solutions that have repeatedly survived are returned.

4.4 Heterogeneously Expensive Evolutionary Algorithm (HE-EA)

The pseudo-code of the proposed Heterogeneously Expensive Surrogate-Assisted Evolutionary Algorithm (HE-EA) is shown in Algorithm 3. HE-EA assumes that an

approximation of evaluation times (ET) for each target exists beforehand. However, if this should not be the case, the evaluation time can be kept track of using a book-keeping approach after evaluating each target. Initially, a list of all targets V is created where first all objectives and then all constraints appear. Afterwards, HE-EA creates a space-filling set of designs P. Then, for each target $v \in V$, the initial population P is evaluated by calling the high-fidelity evaluation function `evaluate(P,v)`, the survival error ρ_v is set to one, the surrogate S_v is fit, and the mean absolute error e_v is estimated using cross-validation. Cross-validation is helpful to provide the algorithm an idea of the complexity of each target.

Until the time limit of running the optimization procedure has been met, the algorithm's main loop is repeated. It starts by performing the mating procedure to generate the offspring population Q and predicting the objective and constraints `predict(S,Q)` using the surrogate S. Analogously, the current population P is copied to P', and the surrogate is used to obtain approximations. Next, the order in which the targets are supposed to be evaluated needs to be determined. The order shall be based on the trade-off between evaluation time ET and surrogate error e. The function `order(ET,`α) first calculates an indicator value for each target. We propose a metric called information gain ($\texttt{IG}_k$) of the k-th target as the survival error (ρ_k) per unit evaluation time ($\texttt{ET}_k$), as follows:

$$\texttt{IG}_k = \frac{\rho_k}{\texttt{ET}_k}. \tag{2}$$

Targets with larger information gain are preferred to be evaluated first and thus, the target evaluation order τ is given by sorting `IG` in *descending* order. To illustrate the intuition behind this order, let us consider a few examples where two targets, v_1 and v_2, are compared with each other. Assuming both targets have the same evaluation time $\texttt{ET}_1 = \texttt{ET}_2$, but target one has a larger survival error $\rho_1 > \rho_2$. This results in $\texttt{IG}_1 > \texttt{IG}_2$ and the first target to be evaluated first. Intuitively this is the right decision because the target with a more significant estimation error can potentially eliminate more solutions already and thus save computation time. On the other hand, let two targets have the same survival error $\rho_1 = \rho_2$, but the first one have a larger evaluation time $\texttt{ET}_1 > \texttt{ET}_2$. This results in $\texttt{IG}_1 < \texttt{IG}_2$. In this case, the effort for evaluation is identical, and the target modeled less accurately is evaluated first. Having discussed two corner cases where one of the metrics is equally, the intuition behind information gain as a metric for the target order is that targets that are difficult to predict by the surrogates or are computationally less expensive are given preference during evaluation.

Algorithm 3: Heterogeneously Expensive Surrogate-Assisted Evolutionary Algorithm (HE-EA)

Input : Evaluation Times ET, Max. Survival Prob. $\alpha^{(\min)}$

```
/* Initialize the target vector                          */
V ← (f_1, ..., f_m, g_1, ..., g_J)
/* Sample design of experiments                          */
P ← doe()
foreach v ∈ V do
    evaluate(P, v)
    ρ_v ← 1.0
    S_v ← fit_surrogate(P, v)
    e_v ← estm_mae(S, P, v)
end
while time left do
    /* Create the offspring population                   */
    Q ← prob_mating(P, Q, S, e, γ, β)
    /* Prediction of P and Q                             */
    Q' ← predict(S, Q) P' ← predict(S, P);
    /* The order to eval. targets                        */
    τ ← order(ET, ρ)
    /* Targets with uncertainty                          */
    V^(U) ← V
    /* Survival prob. before evaluation                  */
    α^(0) ← prob_surv(P', Q', V^(U), e)
    foreach k ← 1 to |V| do
        v ← V[τ_k]
        /* Evaluate and copy targets                     */
        copy(P, P', v); evaluate(Q', v)
        V^(U) ← V^(U) \ {v}
        α^(k) ← prob_surv(P', Q', V^(U), e)
        /* Calculate the survival error                  */
        ρ_v ← Σ_{i=1}^{|Q'|} |α_i^(k) − α_i^(k−1)|
        /* Fit target surrogates and calc. e             */
        S_v ← fit_surrogate(P, v)
        e_v ← mae(S, P, v)
        /* Eliminate unpromising solutions               */
        Q ← eliminate(Q, α^(k), α^(min))
        /* If all solultions were eliminated             */
        if |Q| = 0 then break ;
    end
    P ← survival(P ∪ Q)
end
```

Before starting with the evaluation procedure, the uncertain targets $V^{(U)}$ are initialized to be all targets V, and the initial survival probabilities $\alpha^{(0)}$ are obtained by executing the probabilistic survival (see Algorithm 1). The evaluation process of the offsprings Q loops over the targets V in the order of τ using the counter variable k. In each iteration, the target $v \leftarrow V[\tau_k]$ is then evaluated for the offsprings and

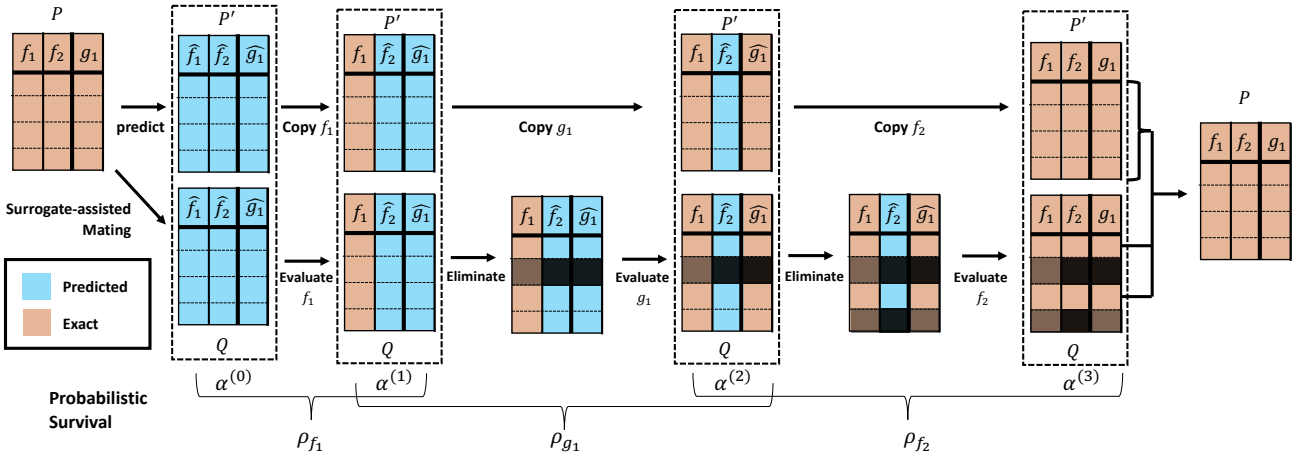


Fig. 4: One iteration of HE-EA consisting of an ordered target evaluation (f_1, g_1, f_2) and offspring eliminations.

copied over for the population. After removing the current target v from the remaining targets $V^{(U)}$, the new survival probability $\alpha^{(k)}$ is calculated, and the survival error ρ_v determined. The survival error is the mean absolute error between the two different α's represents the error introduced by the prediction of target v. Afterward, the surrogate used for the predictions of target v is updated and its prediction error set. In the end of each target iteration, offsprings with a survival probability $\alpha_i^{(k)} \leq \alpha^{(\mathtt{min})}$ are eliminated. The elimination of unpromising offsprings saves time because their evaluation of remaining targets is skipped over.

At the end of the evaluation process, the deterministic survival of the remaining fully evaluated offspring population and the current population is performed. In some iterations, no offspring might be left due to the iterative elimination, and one might wonder if the algorithm can be caught in a deadlock. However, in such iterations, the surrogate for each target has been updated using the already eliminated but partially evaluated offspring solutions. The procedure is then repeated until the time limit of running the algorithm has been reached. A time limit as a termination criterion instead of counting solution evaluation is recommended because it considers full and partial evaluations of individuals.

Let us have a close look at one iteration of HE-EA for a bi-objective optimization problem having two objectives (f_1, f_2) and one constraint (g_1). Let us assume that the target order has been determined to be (f_1, g_1, f_2). A flowchart diagram illustrates the process of evaluating the offspring population (see Figure 4). Initially, the current population P was copied and predicted by the surrogate to become P'. This step can be essential because comparing only predictions with predictions ensures comparing only apples with apples and oranges with oranges. However, if all surrogates fit through the data set exactly, this step is skipped. After generating the predicted population, the offspring population Q is created by surrogate-guided mating, and their predictions are available. In the flowchart, targets predicted by surrogates are shown in blue, and the ones with exact values are shown in orange. In the first iteration of the elimination-based evaluation, the parent and the offspring population are predicted by the surrogate, and the survival probability of $\alpha^{(0)}$ is calculated. After evaluating the first target $v \leftarrow f_1$, the survival probability $a^{(1)}$ having no error noise for f_1 is predicted. The survival error ρ_{f_1} is then determined by the mean absolute error of survival probabilities. Before moving on to the next target, all offspring members with $a^{(1)}$ less than $a^{(\mathtt{min})}$ are eliminated. Then, the target evaluation is continued by evaluating g_1, performing probabilistic survival result in $a^{(2)}$ and updating the surrogate error ρ_{g_1}. Analogously, in the last iteration, f_2 is evaluated. It is worth pointing out that $\alpha^{(3)}$ does not require any probabilities survival because after having evaluated all targets, no noise will be added, which results in a deterministic survival procedure. Thus, the survival probabilities are either zero or one. Finally, the population and the fully evaluated offspring population are sent to the survival operator to determine the population for the next iteration. Notice that the above algorithm also degenerates to the single-objective constraint problems with heterogeneous evaluation times among objective (f) and multiple constraints (g).

4.5 Surrogate Management

Even though surrogate modeling is not the focus of this study, a few words need to be said to complete the al-

gorithm description. In this study, each target value is modeled independently to avoid any error accumulation across targets. Thus, in total $|M| + |J|$ surrogate models are built [17]. For each target, different surrogate models are fitted, and then one with the least mean absolute error is chosen as a predictor. We have used two different types of models: Radial Basis Functions (RBFs) [24] and Kriging [27,28]. Each of them is instantiated with different hyper-parameters, resulting in a list of potential models from which one is selected. It is worth mentioning that the number of data points for each model may vary because of partial evaluations. In order to decrease the computational burden of surrogate fitting, in this study, only the previous 200 data points are considered. The mean absolute error is estimated in each iteration by considering the data points as the training set and the infill solutions evaluated as a validation set. This allows a realistic error estimation in each iteration.

5 Results and Discussions

The capability of the proposed method to exploit independently computable and heterogeneous expensive optimization problems shall be examined next.

We have chosen NSGA-II [19] with a population size of 100 for unconstrained and constrained bi-objective problems. For problems with three objectives, we have used NSGA-III [18,26,9] with 91 references directions originating from uniform weight sampling [14] with a partition number of 12. For both algorithms, the default parameter settings proposed in the papers and defined in their implementations available in the multi-objective optimization framework in Python called pymoo [7] is used. In each experiment, three different algorithms are compared with each other: First, the baseline algorithm without any modifications. This represents an optimization method waiting for all targets to be evaluated despite their heterogeneous evaluation times. Second, a modification of the baseline algorithm incorporating the Elimination-Based Evaluation (EBE) with the default mating procedure. The survival probabilities have been assessed by repeating the survival a hundred times ($\gamma = 100$). Thirdly, the complete heterogeneously expensive (HE) evolutionary algorithm with a surrogate-guided mating ($\beta = 30$) and elimination-based evaluation. The consideration of two different variations shall help to give credit to eliminating unpromising solutions during evaluation and creating solutions more likely to survive beforehand separately. Throughout the experiment, we have fixed the minimum survival probability to $\alpha^{(\texttt{min})} = 0.3$. Some numerical experiments that this seems to be a reasonable value for discarding unpromising individuals. When proposing a new method, the number of hyper-parameters shall be kept as small as possible. It is worth mentioning that in this study, the hyper-parameters γ, β, and $\alpha^{(\texttt{min})}$ have an intuitive meaning which helps to set them properly. Their values were determined through an empirical study, and the performance of configurations close to the suggested one has shown to be similar. Thus, no significant sensitivity could be observed.

Table 1: Average IGD values for unconstrained bi-objective problems from the ZDT test suite. NSGA-II does not use heterogeneous evaluation time information, hence produce identical IGD value for all different evaluation time combinations.

$t(f_1, f_2)$	NSGA-II	EBE-NSGA-II	HE-NSGA-II
ZDT1 ($N = 10$, $M = 2$, $J = 0$)			
(1,19)	0.3258 (−)	0.1053 (−)	**0.0166 (*)**
(5,15)		0.1078 (−)	**0.0169 (*)**
(10,10)		0.1162 (−)	**0.0120 (*)**
(15,5)		0.0968 (−)	**0.0119 (*)**
(19,1)		0.0882 (−)	**0.0099 (*)**
ZDT2 ($N = 10$, $M = 2$, $J = 0$)			
(1,19)	0.6457 (−)	0.1942 (−)	**0.0099 (*)**
(5,15)		0.2303 (−)	**0.0107 (*)**
(10,10)		0.2123 (−)	**0.0139 (*)**
(15,5)		0.2233 (−)	**0.0148 (*)**
(19,1)		0.2316 (−)	**0.0143 (*)**
ZDT3 ($N = 10$, $M = 2$, $J = 0$)			
(1,19)	0.2009 (−)	0.0854 (−)	**0.0376 (*)**
(5,15)		0.0930 (−)	**0.0474 (*)**
(10,10)		0.0777 (−)	**0.0348 (*)**
(15,5)		0.0597 (−)	**0.0324 (*)**
(19,1)		0.0540 (−)	**0.0194 (*)**
ZDT4 ($N = 5$, $M = 2$, $J = 0$)			
(1,19)	27.7984 (−)	**19.4416 (*)**	20.4623 (≈)
(5,15)		21.1689 (≈)	**17.5283 (*)**
(10,10)		17.7420 (≈)	**16.9622 (*)**
(15,5)		**16.2275 (*)**	16.2289 (≈)
(19,1)		**14.9382 (*)**	15.5179 (≈)
Total	0 (*)	3 (*)	17 (*)
	0 (≈)	2 (≈)	3 (≈)
	20 (−)	15 (−)	0 (−)

The performance of the proposed methods is assessed on unconstrained and constrained multi-objective test problems where the time for objective and constraint functions has been systematically varied. We have conducted 11 runs for each problem and algorithm to address the stochastic behavior of the underlying optimization method. All tables presented in this section

show the average IGD values [13]. The best-performing algorithm for each problem is marked as the winner (∗), and other algorithms performing significantly similar (Wilcoxon signed-rank test, $p = 0.05$) are labeled by (≈), and ones are performing significantly worse by (−). Moreover, we have denoted the number of variables by N, the number of objectives by M, and the number of constraints by J for each problem. Starting with bi-objective problems, we have used the ZDT [40] problem suite with the evaluation times for a solution are fixed to 20 time units. The overall evaluation budget is set to seven hours for ZDT1-3 and to 10 hours for ZDT4. This equals 13 and 18 generations of fully evaluated individuals, respectively. For the experiment, the evaluation time for the first objective is set to 1, 5, 15, or 19, and the second objective to make their sum be 20. The results are shown in Table 1. First, one can note that the different evaluation times always lead to identical results for the baseline algorithm, caused by the algorithm waiting for all targets to be evaluated before proceeding. Second, EBE and HE were both able to outperform the NSGA-II no matter what time variation of $t(f_1, f_2)$ has been chosen. By comparing EBE and HE with each other, one can conclude that increasing the probability of a solution surviving during mating is in general helpful. For ZDT1 and ZDT2, much better results, and for ZDT3, still significantly better results have been achieved. For ZDT4, none of the methods can converge close enough to the true Pareto front, given the limited evaluation budget. Thus, for ZDT4, even though both methods outperform NSGA-II, no clear winner can be declared. Altogether, we can conclude that for the considered unconstrained bi-objective problems, HE-NSGA-II shows the best results by winning 17/20 test instances and being significantly similar in the remaining ones.

One benefit of the proposed approach is considering groups of targets and the capability of extending the concept of heterogeneously expensive functions to multiple objectives and constraints. This shall become apparent when discussing the following constrained multi-objective problems. First, we investigate TNK [35, 15], which has two objectives and two constraints and a discontinuous Pareto-front. We have considered all objectives and all constraints, each as a group of targets. The imitates the real-world scenario of Software A returning both objective values and Software B the constraints.

The evaluation time variations have been set analogously to the unconstrained bi-objective problems, and for each run, the time limit is set to three hours. The results listed in Table 2 show the superiority of EBE and HE over the NSGA-II. Across all time variations, EBE and HE converge to the Pareto-optimal set. For four out of five problems, HE-NSGA-II turns out to be significantly the best performing method. Interestingly for $t(f, g) = (1, 19)$ representation, the case of objectives being much less computationally expensive than constraints, EBE performs marginally better.

Table 2: Average IGD values for the constrained bi-objective problem TNK.

TNK ($N = 2$, $M = 2$, $J = 2$)			
$t(f,g)$	NSGA-II	EBE-NSGA-II	HE-NSGA-II
(1,19)	0.0214 (−)	**0.0034 (*)**	0.0043 (≈)
(5,15)		0.0035 (−)	**0.0030 (*)**
(10,10)		0.0040 (−)	**0.0031 (*)**
(15,5)		0.0043 (−)	**0.0030 (*)**
(19,1)		0.0046 (−)	**0.0030 (*)**
Total	0 (*) 0 (≈) 5 (−)	1 (*) 0 (≈) 4 (−)	4 (*) 1 (≈) 0 (−)

For the welded-beam design problem [16], the first objective f_1 is the cost of fabricating the welded beam and can be written in a closed-form mathematical term. Thus, it is relatively quick to compute. The second objective f_2 and constraints g_1 and g_2 are the deflection of the beam-end and hence belong to the same target function group. Constraints g_4 is less time-consuming, but g_3 is the buckling load, requiring more computational effort. Following relative computational times are considered for the target functions: $t(f_1) = 1, t(\{f_2, g_1, g_2\}) = 12, t(g_3) = 12, t(g_4) = 1$. Again, the time limit has been set to three hours.

Table 3: Average IGD values for the constrained bi-objective problem Welded Beam.

Welded Beam ($N = 4$, $M = 2$,$J = 4$)			
	NSGA-II	EBE-NSGA-II	HE-NSGA-II
t	0.078 (−)	0.0577 (−)	**0.0168 (*)**

Overall, the results indicate that HE-NSGA-II performs significantly better than the two competitors. However, it is also worth mentioning that the difference between the average IGD values is relatively small. Some further analysis has shown that is caused by the $\{f_2, g_1, g_2\}$ groups of targets being responsible for a relatively high survival prediction error. Therefore, even though their evaluation is more time-consuming than others, they are scheduled first when the order of targets τ is determined. This clearly shows the challenge of

complex predict target groups being difficult to predict and time-consuming functions.

Table 4: Average IGD values for the three-objective problem DTLZ2.

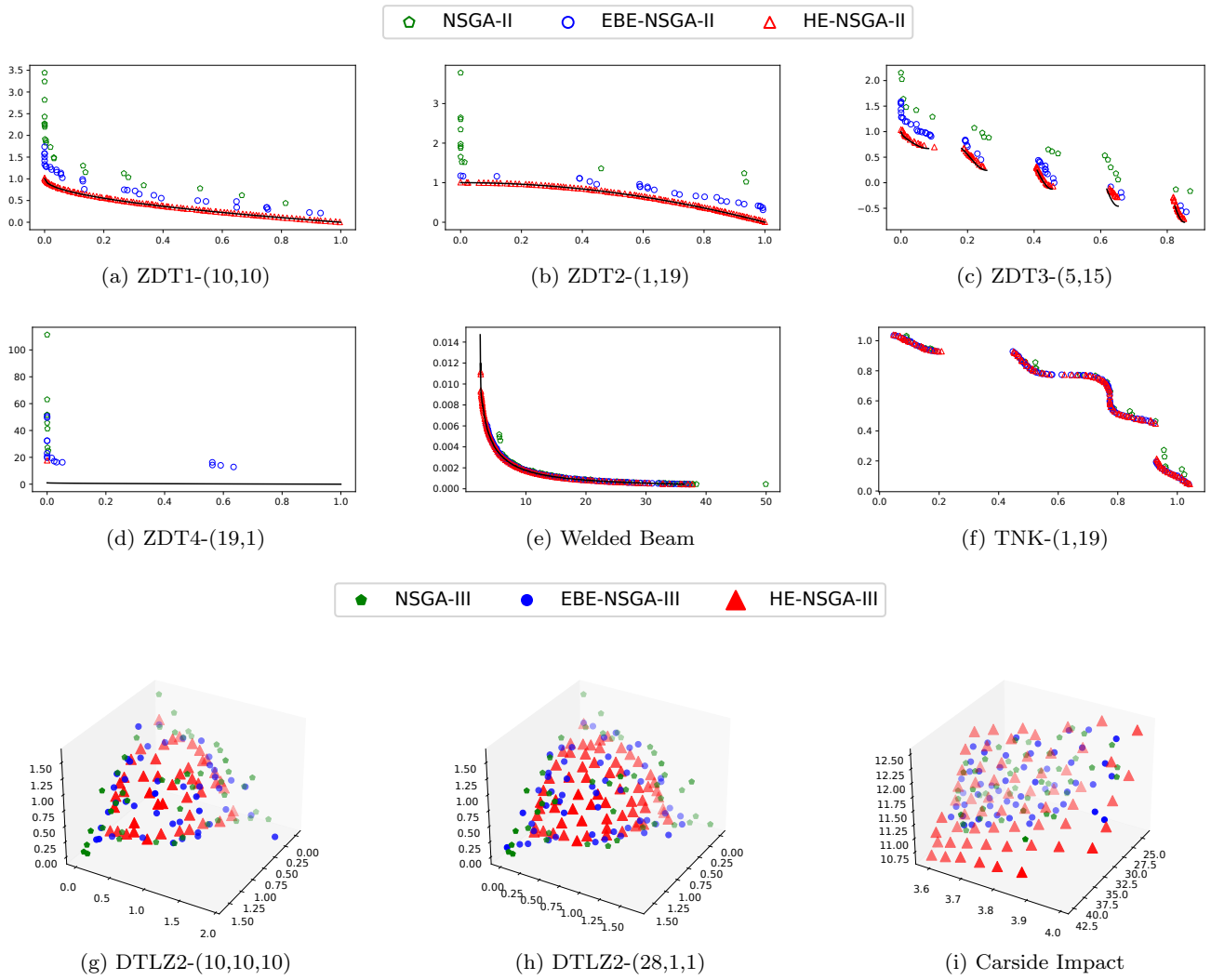
DTLZ2 ($N = 10$, $M = 3$, $J = 0$)			
$t(f_1, f_2, f_3)$	NSGA-III	EBE-NSGA-III	HE-NSGA-III
(28,1,1)	0.2824 (−)	0.1992 (−)	**0.0794 (*)**
(1,28,1)		0.2053 (−)	**0.0806 (*)**
(1,1,28)		0.1926 (−)	**0.0820 (*)**
(25,4,1)		0.2051 (−)	**0.0802 (*)**
(25,1,4)		0.2077 (−)	**0.0791 (*)**
(1,25,4)		0.2050 (−)	**0.0811 (*)**
(4,25,1)		0.2017 (−)	**0.0810 (*)**
(1,4,25)		0.2015 (−)	**0.0825 (*)**
(4,1,25)		0.2011 (−)	**0.0815 (*)**
(15,10,5)		0.2068 (−)	**0.0916 (*)**
(15,5,10)		0.2095 (−)	**0.0957 (*)**
(5,15,10)		0.2021 (−)	**0.0930 (*)**
(10,15,5)		0.2088 (−)	**0.0961 (*)**
(5,10,15)		0.2049 (−)	**0.0889 (*)**
(10,5,15)		0.2027 (−)	**0.0901 (*)**
(10,10,10)		0.2047 (−)	**0.1040 (*)**
Total	0 (*) 0 (≈) 16 (−)	0 (*) 0 (≈) 16 (−)	16 (*) 0 (≈) 0 (−)

Moreover, the proposed methods shall be analyzed for a DTLZ2 [20], an unconstrained three objective optimization problem. The experiment is set up that the evaluation times of all objectives f_1, f_2, and f_3 sum up to 30. In total, we have run the methods for 16 different combinations, varying the expensiveness from being completely homogeneous (10,10,10) to one objective being 28 times more computationally expensive to evaluate than the cheapest function. The time limit for each run has been set to four hours. The average results IGD values for all time variations are shown in Table 4. Whereas EBE improved the performance from NSGA-III, incorporating a more sophisticated mating in HE-NSGA-III outperforms the other competitors significantly across all problems. This implies that the prediction of values during the run was quite accurate and that putting some more bias into the offspring population has shown its effect. Another interesting fact is that the more heterogeneous the evaluations become, the better are the results. Whereas for homogeneous times (10,10,10), HE-NSGA-III achieved an average IGD value of 0.104, which is decreased to 0.0794 for (28,1,1).

Lastly, some visualizations of objective space from different optimization problems are discussed. Figure 5 shows the median performing runs for the baseline algorithms (NSGA-II or NSGA-III) and their EBS and HE variants. We have chosen some representative evaluation times for each problem. The scatter plots confirm the discussion of the results based on IGD values and give the reader an idea of the differences in convergence and diversity to expect by exploiting the heterogeneity. For ZDT1-3 (see Figure 5a to 5c) one can observe the significant difference between the baseline approach and the proposed variants. For ZDT4 (see Figure 5d), the limit evaluation budget was not sufficient to converge for any method. It is worth noting that the figure reveals that EBE achieves a better diversity than HE. For Welded Beam (see Figure 5e), HE-NSGA-II can find more solutions with a smaller value of f_1 whereas for TNK (see Figure 5f), visually, no significant difference can be observed. For DTLZ2, the Figures 5g and 5h show that a larger amount of heterogeneity in fact helps to convergence faster and find a more diverse set. Another problem where the diversity of solutions has been shown to be significantly better is the Carside Impact problem shown in Figure 5i. The problem consisting of three objectives and ten constraints has been set up so that the objectives are computationally inexpensive and the constraints computationally expensive. Altogether, the visual inspections of the median runs of the experiment show how the exploitation of heterogeneously expensive functions can improve the convergence of an existing algorithm.

6 Conclusions

This study has started by investigating the evaluation of independently computable functions during optimization. Four different strategies for evaluating the objectives and constraints of a solution set have been proposed. The strategy of evaluating a set of solutions for a specific target (B/E) has been used to handle constrained multi-objective optimization problems with heterogeneous evaluation times. The proposed evolutionary algorithm has addressed the order of targets during evaluation by sorting the targets by the survival prediction error divided by evaluation time. The differently expensive target functions have been exploited by an elimination-based evaluation which discards partially evaluated solutions based on their likelihood of surviving. The concept can consider each target function separately but also handle target groups. This can especially be useful for practitioners where a software package returns more than one objective or constraint to be used in optimization. Moreover, the proposed approach is applicable to other evolutionary methods where an elitist environment survival is incorporated. This has been demonstrated by adding the support of



(a) ZDT1-(10,10) (b) ZDT2-(1,19) (c) ZDT3-(5,15)

(d) ZDT4-(19,1) (e) Welded Beam (f) TNK-(1,19)

(g) DTLZ2-(10,10,10) (h) DTLZ2-(28,1,1) (i) Carside Impact

Fig. 5: An illustration of the objective space for different types of unconstrained and constrained multi-objective problems. The results are based on representative run of the median performance for each problem. The different expensiveness of target functions and termination criteria are set analogously to the other experiments. The visibly better red-colored points are obtained using the proposed HE-NSGA-III procedure.

differently expensive targets to two well-known multi-objective algorithms. Results on unconstrained and constrained bi-objective and multi-objective problems indicate that the proposed method can efficiently exploit the fact of differently time-consuming target functions.

The proposed approach must also be extended in three directions. First, a further complication of the expensiveness of target functions is worth investigating. In this study, the evaluation time of each target function is kept constant, independent of the solution being evaluated. However, this does not need to be necessarily the case when solving real-world problems. Besides requiring a more sophisticated book-keeping approach of evaluation times, this also introduces another level of uncertainty to the target ordering problem which needs to be addressed.

Second, the generational evaluation approach proposed in this proof-of-principle study can be extended to develop a steady-state method for further gain in overall computational time. In such a method, each offspring member can be evaluated with respect to the parent population in an appropriate adaptively obtained order of the target functions. Its acceptance and elimination can be determined using surrogate models.

Third, this study has focused on how to take advantage of heterogeneous functions for objectives and constraints within a population-based optimization algorithm. Although surrogate models are created and used to determine the order of evaluating target functions

and eliminating evaluation of some population members depending on their multi-objective rank and evaluation time, the surrogate models themselves can be exploited further in arriving at better infill solutions. We defer such studies, which will eventually allow a complete surrogate-assisted heterogeneity-handling EC method that would be practically viable than the usual all-at-a-time evaluation-based algorithms. Nevertheless, our suggestion of a target function ranking scheme based on a member's worth in terms of non-domination and diversity in the population, accuracy of the prediction models, and actual computational times is unique and marks a start of the further future studies for solving heterogeneously expensive problems.

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 pp. 1–19 (2020). DOI 10.1080/0305215X.2020.1808972
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). DOI https://doi.org/10.1016/j.ejor.2014.09.033
3. Allmendinger, R., Knowles, J.: 'Hang on a minute': Investigations on the effects of delayed objective functions in multiobjective optimization. In: R.C. Purshouse, P.J. Fleming, C.M. Fonseca, S. Greco, J. Shaw (eds.) Evolutionary multi-criterion optimization, pp. 6–20. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
4. Allmendinger, R., Knowles, J.: Heterogeneous objectives: State-of-the-art and future research (2021)
5. Anderson, J.D., Wendt, J.: Computational fluid dynamics, vol. 206. Springer (1995)
6. Batista, G., Maria Carolina Monard: An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence **17**(5-6), 519–533 (2003). DOI 10.1080/713827181. URL https://doi.org/10.1080/713827181
7. Blank, J., Deb, K.: pymoo: Multi-objective Optimization in Python. IEEE Access **8**, 89497–89509 (2020)
8. Blank, J., Deb, K.: Constrained bi-objective surrogate-assisted optimization of problems with heterogeneous evaluation times: Expensive objectives and inexpensive constraints. In: H. Ishibuchi, Q. Zhang, R. Cheng, K. Li, H. Li, H. Wang, A. Zhou (eds.) Evolutionary multi-criterion optimization, pp. 257–269. Springer International Publishing, Cham (2021)
9. Blank, J., Deb, K., Roy, P.: Investigating the normalization procedure of NSGA-III. In: K. Deb, E. Goodman, C.A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, P. Reed (eds.) Evolutionary multi-criterion optimization, pp. 229–240. Springer International Publishing, Cham (2019)
10. Brockhoff, D., Zitzler, E.: Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In: T.P. Runarsson, H.G. Beyer, E. Burke, J.J. Merelo-Guervós, L.D. Whitley, X. Yao (eds.) Parallel problem solving from nature - PPSN IX, pp. 533–542. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
11. 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, GECCO '18, pp. 609–616. Association for Computing Machinery, New York, NY, USA (2018). DOI 10.1145/3205455.3205514. URL https://doi.org/10.1145/3205455.3205514
12. 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)
13. Coello Coello, C.A., Reyes Sierra, M.: A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In: R. Monroy, G. Arroyo-Figueroa, L.E. Sucar, H. Sossa (eds.) MICAI 2004: Advances in artificial intelligence, pp. 688–697. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
14. Das, I., Dennis, J.E.: Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. SIAM J. on Optimization **8**(3), 631–657 (1998). DOI 10.1137/S1052623496307510. URL http://dx.doi.org/10.1137/S1052623496307510
15. Deb, K.: Multi-objective optimization using evolutionary algorithms. John Wiley & Sons, Inc., USA (2001)
16. Deb, K., Goyal, M.: Optimizing engineering designs using a combined genetic search. In: PROCEEDINGS OF THE SIXTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, pp. 521–528. Morgan Kauffman Publishers (1995)
17. 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)
18. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation **18**(4), 577–601 (2014). DOI 10.1109/TEVC.2013.2281535
19. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. Trans. Evol. Comp **6**(2), 182–197 (2002). DOI 10.1109/4235.996017. URL https://doi.org/10.1109/4235.996017
20. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: A. Abraham, L. Jain, R. Goldberg (eds.) Evolutionary multiobjective optimization: Theoretical advances and applications, pp. 105–145. Springer London, London (2005)
21. Forrester, A.I., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**(1), 50 – 79 (2009)
22. Forrester, A.I., Sóbester, A., Keane, A.J.: Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **463**(2088), 3251–3269 (2007). DOI 10.1098/rspa.2007.1900
23. Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning, 1st edn. Addison-Wesley Longman Publishing Co., Inc., USA (1989)

24. Hardy, R.L.: Multiquadric equations of topography and other irregular surfaces. Journal of Geophysical Research (1896-1977) **76**(8), 1905–1915 (1971). DOI 10.1029/JB076i008p01905. URL https://doi.org/10.1029/JB076i008p01905
25. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: C.A.C. Coello (ed.) Learning and intelligent optimization, pp. 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
26. 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 (2014)
27. Krige, D.G.: 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 (1951)
28. Lophaven, S., Nielsen, H.B., Søndergaard, J.: DACE – a MATLAB kriging toolbox (2002)
29. Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., Banzhaf, W.: NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In: Proceedings of the genetic and evolutionary computation conference, GECCO '19, pp. 419–427. Association for Computing Machinery, New York, NY, USA (2019). DOI 10.1145/3321707.3321729. URL https://doi.org/10.1145/3321707.3321729
30. Mezura-Montes, E., Coello, C.A.C.: Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm and Evolutionary Computation **1**(4), 173–194 (2011)
31. Rahi, K.H., Singh, H.K., Ray, T.: Investigating the use of sequencing and infeasibility driven strategies for constrained optimization. In: 2019 IEEE congress on evolutionary computation (CEC), pp. 1642–1649 (2019). DOI 10.1109/CEC.2019.8790239
32. Rahi, K.H., Singh, H.K., Ray, T.: Feasibility-ratio based sequencing for computationally efficient constrained optimization. Swarm and Evolutionary Computation **62**, 100850 (2021). DOI https://doi.org/10.1016/j.swevo.2021.100850
33. Rahi, K.H., Singh, H.K., Ray, T.: Partial evaluation strategies for expensive evolutionary constrained optimization. IEEE Transactions on Evolutionary Computation pp. 1–1 (2021). DOI 10.1109/TEVC.2021.3078486
34. Szabó, B., Babuška, I.: Finite element analysis. John Wiley & Sons (1991)
35. Tanaka, M.: GA-based decision support system for multicriteria optimization. In: Proceedings of the international conference on systems, man and cybernetics, vol. 2, pp. 1556–1561 (1995)
36. Thomann, J., Eichfelder, G.: Representation of the Pareto Front for heterogeneous multi-objective optimization **1**, 293–323 (2019). DOI 10.23952/jano.1.2019.3.08
37. Thomann, J., Eichfelder, G.: A trust-region algorithm for heterogeneous multiobjective optimization. SIAM Journal on Optimization **29**(2), 1017–1047 (2019). DOI 10.1137/18M1173277. URL https://doi.org/10.1137/18M1173277
38. 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, GECCO '20, pp. 587–594. Association for Computing Machinery, New York, NY, USA (2020). DOI 10.1145/3377930.3390147. URL https://doi.org/10.1145/3377930.3390147
39. Wang, X., Jin, Y., Schmitt, S., Olhofer, M., Allmendinger, R.: Transfer learning based surrogate assisted evolutionary bi-objective optimization for objectives with different evaluation times. Knowledge-Based Systems **227**, 107190 (2021). DOI https://doi.org/10.1016/j.knosys.2021.107190
40. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8**(2), 173–195 (2000). DOI 10.1162/106365600568202. URL https://doi.org/10.1162/106365600568202