Effectiveness of Surrogate-Based Optimization Algorithms for System Architecture Optimization

Jasper Bussemaker, Nathalie Bartoli, Thierry Lefebvre, Pier Davide Ciampa, Björn Nagel

Effectiveness of Surrogate-Based Optimization Algorithms for System Architecture Optimization

J.H. Bussemaker*
DLR (German Aerospace Center), Institute of System Architectures in Aeronautics, Hamburg, Germany

N. Bartoli†, T. Lefebvre‡
ONERA, DTIS, Université de Toulouse, Toulouse, France

P.D. Ciampa§, B. Nagel¶
DLR (German Aerospace Center), Institute of System Architectures in Aeronautics, Hamburg, Germany

The design of complex system architectures brings with it a number of challenging issues, among others large combinatorial design spaces. Optimization can be applied to explore the design space, however gradient-based optimization algorithms cannot be applied due to the mixed-discrete nature of the design variables. It is investigated how effective surrogate-based optimization algorithms are for solving the black-box, hierarchical, mixed-discrete, multi-objective system architecture optimization problems. Performance is compared to the NSGA-II multi-objective evolutionary algorithm. An analytical benchmark problem that exhibits most important characteristics of architecture optimization is defined. First, an investigation into algorithm effectiveness is performed by measuring how accurately a known Pareto-front can be approximated for a fixed number of function evaluations. Then, algorithm efficiency is investigated by applying various multi-objective convergence criteria to the algorithms and establishing the possible trade-off between result quality and function evaluations needed. Finally, the impact of hidden constraints on algorithm performance is investigated. The code used for this paper has been published.

Nomenclature

CR	=	Consolidation Ratio	LHS	=	Latin Hypercube Sampling
DOE	=	Design of Experiments	MBSE	=	Model-Based Systems Engineering
EA	=	Evolutionary Algorithm	MCD	=	Maximal Crowding Distance
EEI	=	Euclidean Expected Improvement	MDO	=	Multidisciplinary Design Optimization
EI	=	Expected Improvement	MDR	=	Mutual Domination Rate
EGO	=	Efficient Global Optimization	MI	=	Mixed-Integer
EHVI	=	Expected Hypervolume Improvement	MO	=	Multi-Objective
EMA	=	Exponential Moving Average	MOE	=	Mixture of Experts
EMFI	=	Expected Maximin Fitness Improvement	MOEA	=	MO Evolutionary Algorithm
EPoI	=	Enhanced Probability of Improvement	M(E)PoI	=	Minimum (Euclidean) PoI
EV	=	Expected Violation	MVPF	=	Minimum Variance of Pareto Front
f	=	Objective function or values	PF	=	Pareto Front
FHI	=	Fitness Homogeneity Indicator	PoI	=	Probability of Improvement
g	=	Constraint function or values	PoF	=	Probability of Feasibility
GA	=	Genetic Algorithm	RBF	=	Radial Basis Function
GD	=	Generational Distance	SBO	=	Surrogate-Based Optimization
HV	=	Hypervolume	SPI	=	Steady Performance Indicator
KPLS	=	Kriging with Partial Least Squares	$\hat{\square}$	=	Surrogate model estimate
LCB	=	Lower Confidence Bound			

*Researcher, MDO group, Aircraft Design & System Integration, Hamburg, jasper.bussemaker@dlr.de
†Senior researcher, Information Processing and Systems Department, nathalie.bartoli@onera.fr, AIAA Member MDO TC
‡Research engineer, Information Processing and Systems Department, thierry.lefebvre@onera.fr
§Head of MDO Group, Institite of System Architectures in Aeronautics, Aircraft Design & System Integration, Hamburg, pier.ciampa@dlr.de, AIAA MDO TC member
¶Institute director, Institite of System Architectures in Aeronautics, Hamburg, bjoern.nagel@dlr.de

I. Introduction

THE increasing complexity of aerospace products brings several challenges to the design of new systems [1]. Many requirements and constraints on the transportation system, the manufacturing system, the aircraft itself, individual systems and components have to be taken into account. In addition, ever stricter regulations are applied to emission and noise levels. An important solution for overcoming such challenges lies in the ability to take important decisions earlier on in the design process. Such decisions have a high impact on the performance of the final design, and suffer from uncertainty owed to the lack of knowledge at early design stages. Additionally, due to the complexity of aerospace systems, many interacting decisions have to be taken into account at the same time, leading to extremely large combinatorial design spaces. To be better able to take such architecting decisions with an increased level of certainty, it is needed to apply physics-based systematic design space exploration techniques [2]. This also reduces the reliance on expert judgment, which, while quick, can also suffer from expert bias, subjectivity, conservatism, and overconfidence.

First, it should be made possible to quantitatively evaluate the performance of candidate architectures, so they can be objectively compared to each other in terms of design goals. This can be a complicated multi-disciplinary, multi-organizational process involving many communication and socio-technical challenges [1]. It requires the ability of quickly formulating and deploying design systems to support the quantitative multi-disciplinary analysis of candidate architectures and make sure such analysis is coherent, consistent, and meaningful [3].

Once it is possible to evaluate architecture candidates, it is then needed to model the system architecture design space in such a way as to explicitly identify all architecture design choices that need to be evaluated, and yield function-to-form mappings for the justification of the system architectures with respect to system stakeholders and their needs. In addition to enabling integration within the wider MBSE process, such a model offers the formalization needed to construct an optimization problem based on the identified design choices. An optimization algorithm can then explore the design space to find the most optimal system architecture, or the Pareto-optimal set of system architectures if multiple objectives are optimized for.

System architecture optimization problems are challenging to solve due to their black-box, hierarchical, mixed-integer and multi-objective nature (see Sec. II.B). Research is needed into what optimization algorithms can deal with this kind of problems, and to identify areas where further research might be necessary. This paper addresses this need, by comparing the performance of different optimization algorithms for solving a system architecture optimization problem. An overview is presented on system architecting, its place in the systems engineering process, and behavior of system architecture optimization problems. Then, a literature review of currently existing optimization algorithm components is given. Investigations are performed into the effectiveness (how well the "real" Pareto front is approximated) and the efficiency (how many function evaluations are needed for a certain approximation of the Pareto front). Finally, an investigation into architecture optimization subject to hidden constraints is presented.

II. System Architecting

A. Complex System Architecting

The development of modern complex systems needs to account for an ever increasing number of capabilities to be delivered, as well as for organizational boundaries, integration and communication challenges, and constraints stemming from all the stages of the product's life-cycle. Ideally every decision taken at each stage of the development should be evaluated along the entire life-cycle. The management of such development complexity requires a shift to a novel system development paradigm.

In this context, the DLR "Institute of System Architectures in Aeronautics" is developing a novel "*model based conceptual framework*" for architecting, designing and optimizing complex aeronautical systems. The expected impact is a drastic reduction in time and costs associated with the development, via an increased transparency, efficiency, and traceability of the design and decision making processes. The conceptual framework, introduced in [4] extends the scope of design and optimization methods to all the phases of the development life cycle of complex systems. The implementation of the concept is supported by the development of novel design methods and approaches, leveraging digital design engineering and modeling technologies. The work presented in this paper focuses on solving system architecture optimization problems, and is part of the European Commission funded project AGILE 4.0 (2019-2022) [5]. The project will act as test environment for architecture optimization, and several realistic system architecting problems will be defined and explored within the scope of the project.

System architecting can be seen as a step in the systems engineering process [6]. The system of interest consists of a set of interconnected components that together fulfill the functions that justify the existence of the system. A system

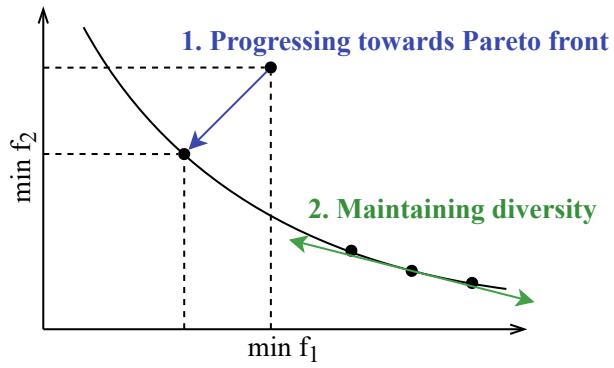


Figure 1 The goals of multi-objective optimization: progress towards the Pareto-front while maintaining sufficient diversity, adapted from [8].

architecture describes how these functions are fulfilled, by iteratively mapping elements of form (i.e. architecture components) to function, and defining connections among elements of form. This paper builds on previous work on modeling the design space of system architectures as presented in [2]. Modeling the system architecture design space enables the semantic representation of architecture choices involved in the construction of architectures. This enables integration in the MBSE process, and offers the formalization needed for formulating an optimization problem from the design space model.

B. System Architecture Optimization

Optimization is the formalization of a design task: an automated way of finding the best design to solve a given problem [7]. In optimization, an optimization algorithm systematically varies a set of design variables (a design vector), and observes the change in one or more merit functions (or objective functions) this change of design variables elicits. Its goal is then to find a design vector that minimizes (or maximizes) the values of these objective functions, while at the same time making sure that the values of any constraint functions remain within limits. The challenge of solving such a problem is increased by non-linear and a-priori unknown objective and constraint function behavior. Many different algorithms exist for solving optimization problem, with various degrees of effectiveness for different types of optimization problems. Here, effective means either to have a high probability of finding the optimal point at all, or finding (an approximation of) the optimal point in the least amount of steps [7]. This last point is important, as the evaluation of one design vector to find out its objective and constraint function values, usually takes several orders of magnitude longer than the internal calculations of the optimization algorithms. This is easy to confirm by thinking of aerodynamic wing shape optimization: one evaluation of a possible shape might involve running high-fidelity CFD (Computational Fluid Dynamics) analysis on a supercomputer.

In this paper we are dealing with the optimization of system architectures, which poses several difficulties for optimization algorithms [2]. In optimization, a distinction is made between design variables of continuous type and of discrete type. Discrete design variables can be further broken down into integer (ordering and distance have meaning) and categorical (no notion of order or distance) design variables. System architecture optimization problems might have all of these types of design variables: an example of an integer design variable might be the number of engines per wing, a categorical design variable might be whether to add a winglet or not, a continuous design variable might be the bypass ratio of the engine. This mixed-discrete nature of the design variables makes it more difficult to solve the optimization problem, as it excludes the use of gradient-based optimization algorithms: algorithms that use local derivative information to find the "direction" towards the optimum.

Another characteristic of system architecture optimization is that because of conflicting stakeholder needs, there are in general multiple objectives that a system architecture might be optimized for. This means that rather than finding one optimal architecture, the goal is to find a set of Pareto-optimal architectures: architectures that are better than others for some objectives, but worse for other objectives [9]. Within the Pareto set, no architecture can be said to be better than the other, and therefore an additional decision-making step that chooses the final architecture based on additional considerations needs to be performed after the optimization has been completed. Solving these so-called multi-objective optimization problems is hard, because instead of simply moving towards the optimal point, there is also a need to find

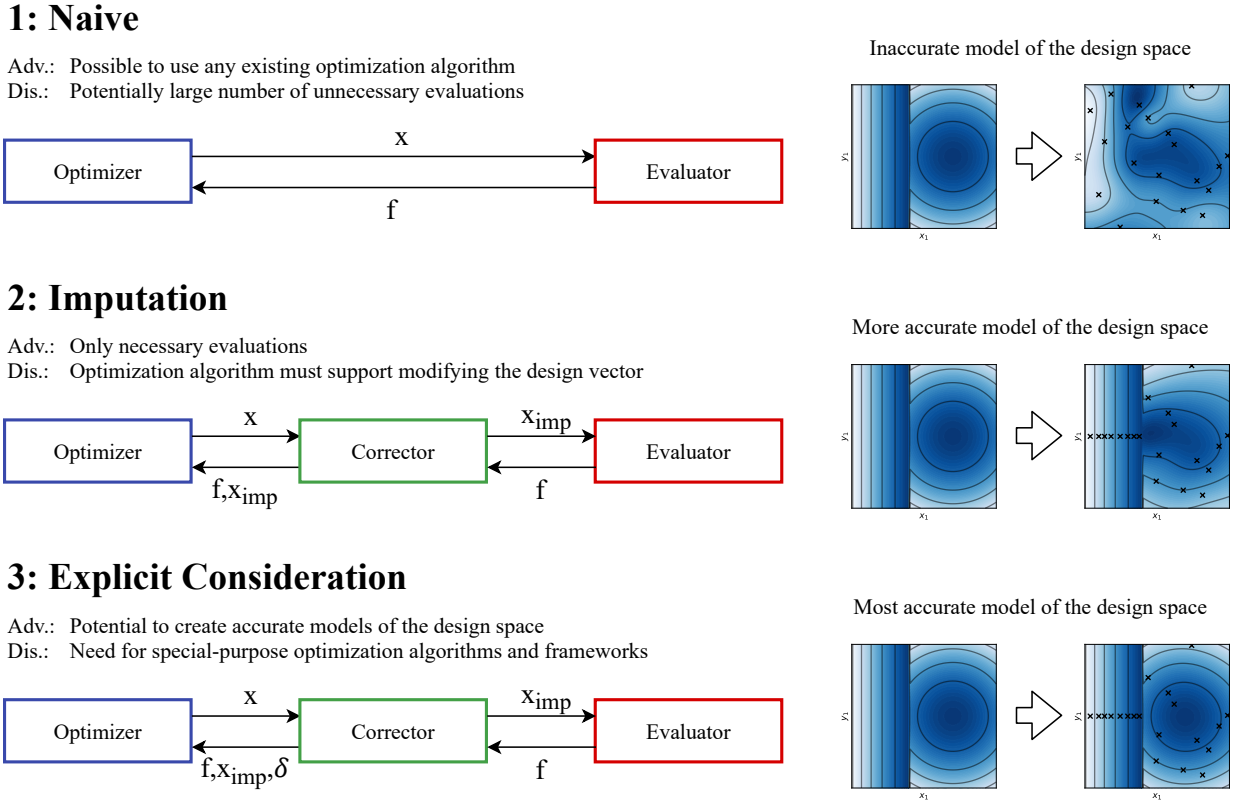


Figure 2 Three strategies for dealing with hierarchical optimization problems [10]: naive (ignore the effects), imputation, and explicit consideration. The corrector converts a design vector to its canonical form (x_{imp}) by imputing inactive design variables, and optionally returns which design variables are active (δ). This makes sure design points are only evaluated once. The contour plots on the right give a notional view of how accurate the optimizer can model the design space, based on the test problem from [10]: on the left side of the domain, only one design variable is active.

design points sufficiently spread-out along the Pareto-front (i.e. the set of Pareto points in the solution-space). This is called the balance between exploitation (i.e. finding the best design) and exploration (i.e. finding new regions in the design space of potential best designs), the principle is visualized in Fig. 1.

Next, system architecture optimization problems feature decision hierarchy: design variables can be conditionally active based on other design variables. This means that there are regions in the design space where one or more of the design variables have no influence of the performance of the architecture. There are several ways of dealing with this problem [10] (see Fig. 2 for a visual explanation): ignore the effects, imputation, and explicit consideration. Ignoring the effects potentially confuses the optimization algorithm, because multiple different design vectors might actually map to the same design point (because of inactive design variables). This is solved by imputation: inactive design variables are replaced by some predefined value, so that no duplicate design vectors are evaluated. Explicitly considering hierarchy effects can yield the most effective optimization algorithms, however this requires special-purpose optimization algorithms, and information about which design variables are active at locations in the design space (the δ-function of [11]). It should be noted that regardless of the strategy chosen, there will always be a difference between the apparent and feasible design space [2]: the apparent design space spans all combinations of all design variables (i.e. what the optimization algorithm sees), the feasible design space takes out combinations leading to infeasible or duplicate design points due to inactive design variables (i.e. what the architecture evaluator sees).

Finally, because quantitative performance evaluation of systems depends on numerical simulation techniques, architecture optimization problems may also be subject to hidden constraints. Hidden constraints are constraints that are not known to the optimizer, and only the status (i.e. satisfied or violated) is known qualitatively after completing an evaluation [12]. They are especially common in simulation-based optimization, because it can happen that simulations

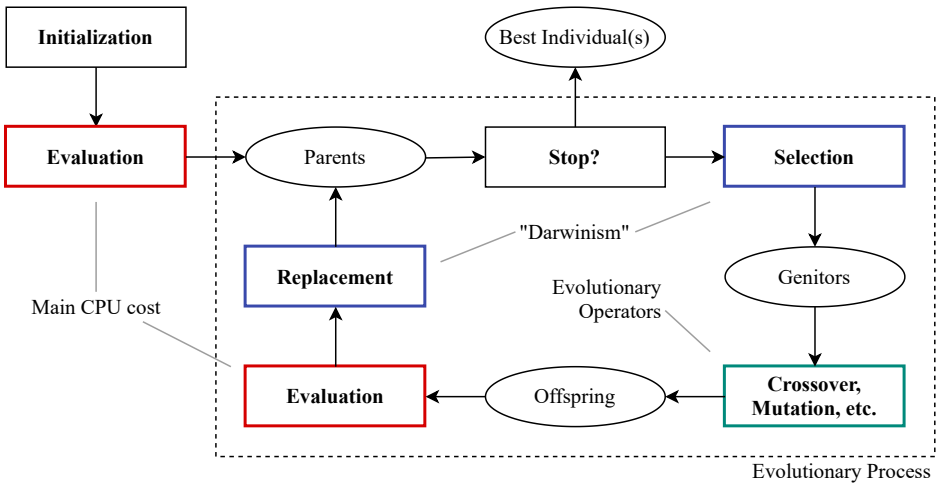


Figure 3 General procedure of an Evolutionary Algorithm, adapted from [8].

do not converge to a meaningful result [13]. This can also mean that if such constraint is violated, none of the other objective and constraint values has a valid result. Up to 60% of evaluations in a design of experiments may result in violated hidden constraints (i.e. infeasible results) [12].

In general, system architecture optimization problems can therefore be considered to have the following features:

- The design space is *non-linear* and its shape is *unknown a-priori* (the evaluation function is a "black-box"),
- Design variables are of *mixed-discrete* nature,
- A *hierarchy* between design variables may be present,
- The optimization problem may have *multiple objectives* to optimize for, and
- *Hidden constraints* may exist in the design space.

III. Optimization Algorithm Components

This section will provide an overview of existing types of optimization algorithms that in principle can tackle mixed-integer, multi-objective, hierarchical, black-box optimization problems: the type of problem involved in system architecture optimization. Additionally, multi-objective convergence criteria are discussed, as detecting the convergence of the Pareto front towards the real Pareto front is a non-trivial task compared to detecting single-objective convergence.

A. Evolutionary Algorithms

Generally a good choice for solving mixed-discrete optimization problems are Evolutionary Algorithms (EA's), originally known as the Genetic Algorithm (GA). EA's are based on the natural selection process as observed by Darwin [14]: an initial population of individuals (design points) is evolved (improved) by generating new generations (new design points) to replace parts of the current generation: parents are selected from the current population (current best design points) and offspring (new design points) is created by applying different kinds of genetic operators, like crossover and mutation. The offspring is then compared to the current population and used to replace all or parts of the current generation, based on comparing their fitness (objective) values and optionally retaining the current best individuals from the current population (elitism). This process is visualized in Fig. 3. Evolutionary algorithms are well suited for finding global optima, as long as the selection and configuration of the different genetic operators allow for sufficient exploitation (improving already-good points) and exploration (trying out new points in the design space).

Evolutionary Algorithms use crossover and mutation to generate offspring from selected parents. Different crossover and mutation operators are available. Additionally, different crossover and mutation operators can be used for continuous than for discrete design variables. Since in architecture optimization, both types of optimization variables will be present in the optimization problem, different types of evolutionary operators will be used on different design variable types. For the experiments in this study, the Simulated Binary Crossover and Uniform Crossover [15] are used for continuous and discrete design variables, respectively. For the mutation operation, Polynomial Mutation and Bitflip mutation [15] are used for continuous and discrete design variables, respectively.

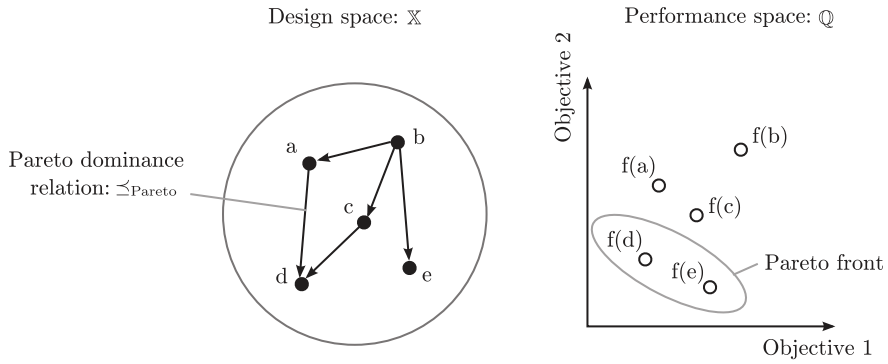


Figure 4 Pareto dominance relation, from [16]. In this example, point D and E are both in the Pareto set, since they dominate all other points, however not each other.

Multi-Objective Evolutionary Algorithms (MOEA's) are adaptations of the EA process where mainly the selection and replacement steps are modified to result in better exploration of the multi-objective search space [17]. Instead of optimizing for one objective directly, rather the population is improved by binary comparison according to some Pareto dominance relation (see Fig. 4), which results in an improvement of design points towards to Pareto-front, whereas at the same time a proper spread along the Pareto-front is ensured. Selecting individuals for creating offspring is done based on Pareto-rank as well. The most used MOEA is the Nondominated Sorting Genetic Algorithm II (NSGA-II) [18]. This algorithm is based on ranking the design points based on their distance from the Pareto-front (exploitation), closer is better, and the distance to other neighboring points (exploration), further is better.

Evolutionary Algorithms are effective at dealing with hierarchical optimization problems using the hidden genes approach [19]. The definition of the design space (the chromosomes) includes the maximum number of design variables (genes). Inactive design variables (hidden genes) are excluded from the cost function evaluation, however are still visible and actively take part in the design space exploration (e.g. evolutionary operators). There is therefore a difference between what the evolutionary algorithm operators on and what is used for cost function evaluation: this is the same phenomena of apparent vs feasible design spaces as discussed before.

Since Evolutionary Algorithms depend on comparison between results, for example through non-dominated sorting in the case of NSGA-II, the extreme barrier approach can be used to deal with hidden constraints [20]: objectives and constraints are assigned a value of $+\infty$ in case of violated hidden constraints, so that selection and replacement are guided towards feasible design points.

B. Surrogate-Based Optimization

Evolutionary algorithms are known for their robust effectiveness for finding global optima, but do require a relatively high number of function evaluations (however still many less than design space enumeration). This is a potentially problematic point, as it is expected that the main computation cost lies exactly there: at the function evaluations. Additionally, because simulation becomes more high-fidelity, this problem will only grow in the future. One class of optimization algorithms specifically designed to mitigate this problem are Surrogate-Based Optimization (SBO) algorithms [22]. Efficient Global Optimization (EGO) was one of the first SBO [23], based on Kriging model and the Expected Improvement (EI) infill criterion. The process, visualized in Fig. 5, is the same for all SBO algorithms: create an initial sample to build the surrogate model, sample new points based on an infill criterion and update the surrogate model, check if our convergence criterion has been reached, and sample new points if convergence has not been reached yet. The infill criterion is used to select new points to sample, and usually includes some trade-off between exploration and exploitation. For example, if Kriging surrogate models are used, model uncertainty information is used to aid in exploration (to explore areas to improve the current estimate of the surrogate model, in the hope that a new area of global optimum is found [24]). Exploitation is aided by simply selecting points to evaluate where the surrogate model predicts a global optimum. The great advantage of this approach is that because evaluating a point in the surrogate model is extremely cheap compared to evaluating the real function, an exhaustive search of the design space to find the best infill points can be performed. Training a surrogate model, however, can take a non-trivial amount of time and therefore should be taken into account.

The two most popular interpolating surrogate models used for SBO are Radial Basis Function (RBF) and Kriging

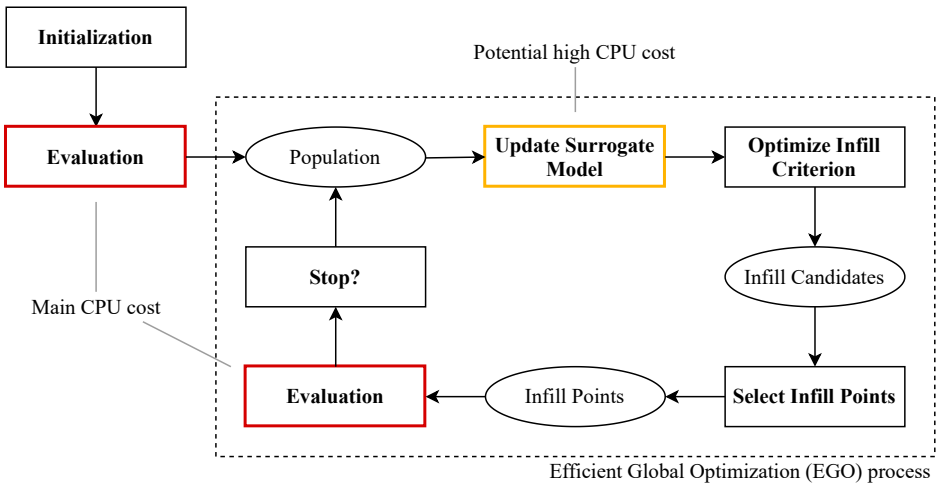


Figure 5 Process of the Efficient Global Optimization (EGO) algorithm, adapted from [21]. Surrogate-Based Optimization (SBO) algorithms are based on the same process, with variations in the initial sampling strategy, the underlying surrogate model, the infill criterion, and the stopping criterion.

surrogate models [22]. The main difference is that next to the function estimate itself, Kriging models (also known as Gaussian process models [25]) also calculate the uncertainty of the estimate. This allows the development of infill criteria that not only attempt to find the global minima (exploitation), but also attempt to improve the estimate of the surrogate model itself as to better be able to predict areas with global minima (exploration). Kriging models, however, have difficulty training for high-dimensional design spaces, which is less of a problem for RBF models [26]. Kriging modified with a Partial Least Squares estimation (KPLS) increases the training speed of Kriging models and might be a powerful mechanism for dealing with high-dimensional spaces [27].

SBO algorithms are very promising for efficiently finding the global optimum, however due to the nature of the used surrogate models are usually better suited for modeling continuous design spaces. A Kriging surrogate model uses kernels to calculate correlations between sample points, such that points that lie closer to each other are assumed to be more correlated to each other than points that lie further away. A common kernel is the exponential kernel, where correlation exponentially reduces with distance. The problem with mixed-integer optimization problems is that for the categorical variables, there is no notion of distance. Several different ways to solve this problem have been studied in the past, including using hybrid branch-and-bound algorithms [28], treating discrete variables as continuous variables [29], continuous relaxation or dummy coding [30], and developed special-purpose mixed-integer kernels [31, 32].

Infill Criteria Infill criteria determine the place in the design space where new points are sampled to increase the prediction quality of the model, and to confirm regions with global minima. The simplest infill criteria are based on directly finding either the design point where the surrogate model predicts the best objective function value $\hat{y}$ (bias towards exploitation, e.g. [34]), or where the model uncertainty (variance $\hat{s}$) is highest (bias towards exploration, e.g. [35]) [36]. Much effort has been invested in finding infill criteria with more natural balances between exploration and exploitation. The Expected Improvement (EI) [23] is a more balanced metric and represents the amount the function can be improved with respect to the current best point given that an improvement is achieved, and is calculated by:

$$EI(x) = (f_{min} - \hat{y}(x)) \cdot \Phi\left(\frac{f_{min} - \hat{y}(x)}{\hat{s}(x)}\right) + \hat{s}(x) \cdot \phi\left(\frac{f_{min} - \hat{y}(x)}{\hat{s}(x)}\right),$$

where f_{min} is the function value of the current best sample, $\hat{y}$ and $\hat{s}$ are the function and variance estimates, respectively, Φ and ϕ the standard normal density and distribution functions, respectively. The infill criterion then selects the point with the highest EI as new point to sample. Another criterion uses the Lower Confidence Bound (LCB) [37], which is the lowest expected value to be observed at a point given its normal distribution, and is calculated by:

$$LCB(x) = \hat{y}(x) - \alpha \cdot \sqrt{\hat{s}(x)},$$

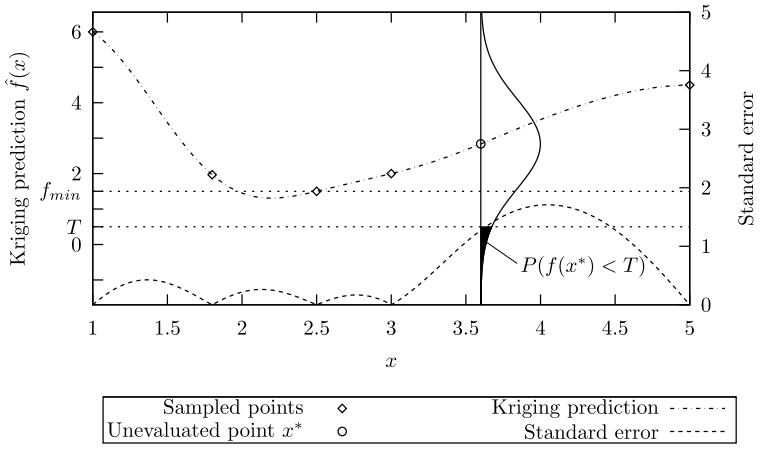


Figure 6 Illustration of the Probability of Improvement (PoI), from [33]: the probability that the function value will be better than some value $T < f_{min}$ given the surrogate model estimate $\hat{y}$ and variance $\hat{s}$.

where α is a scaling parameter (a typical value is 2 [37]). The LCB then replaces the objective function value and can be directly optimized for (minimized) to find a point with a potential new best function value. Another metric, the Probability of Improvement (PoI) [33], gives the probability that some value with an offset to the current best value $T < f_{min}$ is improved upon given the model estimate and its normal distribution (visualized in Fig. 6):

$$PoI(x) = \Phi\left(\frac{T - \hat{y}(x)}{\hat{s}(x)}\right)$$

The point with the maximum PoI can then be used for sampling the function.

If an RBF surrogate model is used, only $\hat{y}$ is available and therefore such balanced infill criteria as discussed before cannot be used. One strategy to still enable exploration of the design space is to use Coordinated Perturbation (CP) [38], a method similar to Simulated Annealing (SA), but where the probability of evaluating a new point is based on a trade-off between the distance to the current point (exploration) and the predicted function value (exploitation). CP modified for use in multi-objective optimization will be called the $\hat{y}$-Dist criterion, and implements the following differences with respect to the single-objective CP criterion:

1) Objective values are normalized and the euclidean distance is used to calculate the minimum distance (see [38]);
2) The RBF criterion and minimum distance are defined as separate objectives in the infill problem, to naturally balance between exploration and exploitation; this also removes the need for normalization between the RBF criterion and the minimum distance.

Constraint Handling Constraint handling in Surrogate-Based Optimization can be approached in several different ways [39]. One common way is to transform the constrained problem into an unconstrained problem by modifying the objective to be penalized due to constraint violation, thereby driving the optimizer towards the feasible region. For this approach it is difficult to find a balance between the objective value and the penalty multiplier. Another way is to drive the infill criterion to ignore points where the surrogate model for the constraint functions expect the constraint to be violated. This, however, can suffer from estimation errors and potentially ignore points lying in a feasible but unexplored area of the design space. One way to solve this is the Expected Violation (EV) metric, that includes information about the model variance to only ignore new infill points if this value (analogue to EI) is above some predefined threshold. The Probability of Feasibility (PoF) metric works according to the same principle [40].

Dealing with hidden constraints is more problematic, since violated hidden constraints result in all outputs (i.e. objectives and constraints) being invalid. The extreme barrier approach (replacing invalid values with $+\infty$) is not applicable, as infinite values cannot be modeled in surrogate models. One way of dealing with this is by replacing invalid values with the maximum of valid values, so that the surrogate model will predict unattractive objective and constraint values in the region of violated hidden constraints [12]. Another approach is by removing invalid design points from the surrogate model training set, and creating a second model that predicts whether a design point will be feasible or not [41]. The second predictor might then be based on some classification model, for example random trees.

Table 1 Approaches for applying surrogate models on mixed-integer (MI) and hierarchical (H) design spaces. n_c represents the number of combinations of the discrete variables. Note that the MI kernels imply that continuous kernels are used for continuous design variables.

Nr of Models	Input space	Input pre-processing	Kriging Kernel	Reference
n_c	Continuous	N/A	Continuous	e.g. Pelamatti et al. [45]
1	MI	Continuous relaxation	Continuous	e.g. Garrido-Merchán et al. [30]
		Dummy coding	Continuous	Herrera et al. [46]
		N/A	Hamming distance (Ham)	e.g. Zaefferer et al. [47]
			Gower distance (Gow)	Halstrup [48]
			Symbolic Covariance (SC)	McCane et al. [49]
			Compound Symmetry (CS)	Roustant et al. [50]
			Latent Variables (LV)	Zhang et al. [51]
	MI + H		Arc	Hutter et al. [11]
			Indefinite Conditional (Ico)	Zaefferer et al. [10]
			Imputation (Imp)	Zaefferer et al. [10]
			Wedge (Wed)	Horn et al. [52]
			SPW decomposition	Pelamatti et al. [53]
			DVW decomposition	Pelamatti et al. [53]

1. Multi-Objective, Mixed-Integer and Hierarchical Problems

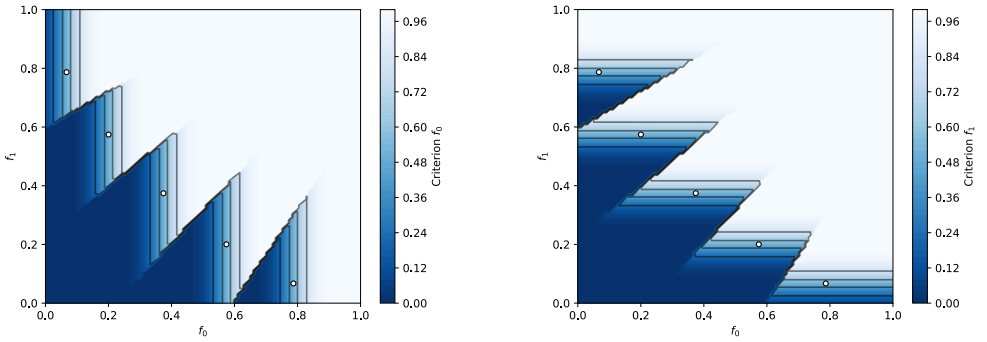
Dealing with multi-objective optimization problems has also been an important topic of research for SBO algorithms [42]. Many strategies have been studied, including multi-objective variations of the Expected Improvement [43] and Probability of Improvement (PoI) [33] criteria, the Expected Hypervolume Improvement (EHVI) [44], or simply optimizing for the estimated objective values themselves [24]. Infill criteria for multi-objective optimization are discussed in more details in a later section.

Extending Kriging surrogate models to support mixed-integer and hierarchical (i.e. variable-size) design spaces is an ongoing area of research, Tab. 1 provides an overview of existing approaches, and good reviews are provided by [10, 54]. One method is to simply train a surrogate model of the continuous design variables for each of the combinations of the discrete variables, known as Category-Wise (CW) Kriging [45]. The problem with this approach is that for the optimization of expensive black-box functions there are often not enough samples available to construct an accurate surrogate model for each of the discrete variable combinations [53]. An alternative approach is to model the design space including the continuous and discrete variables in one surrogate model. An obvious possibility is to treat the discrete variables as continuous variables, also called continuous relaxation; this method might be more appropriate for integer design variables, and less for categorical design variables. Kriging kernels capable of modeling mixed-integer spaces include the Hamming distance [47], Gower distance [48], and Symbolic Covariance (SC) [49], Compound Symmetry (CS) [50], and Latent Variables (LV) [51] kernels. Kernels that can also handle hierarchical design spaces include the Arc [11], Indefinite Conditional [10], Imputation [10], Wedge [52], Sub-Problem-Wise (SPW) decomposition [53], and Dimensional Variable-Wise (DVW) decomposition [53] kernels. It is noted that Baert et al. [55] mention that they have implemented mixed-integer Kernels inspired by Value Different Metrics (VDM) [56]. However, the VDM and derived mixed-integer metrics depend on the relative number of occurrences of output labels, and are therefore only applicable to classification problems (discrete output), and not to regression problems (continuous output) as with SBO.

An overview of SBO approaches including research into applying SBO for mixed-integer or multi-objective optimization is shown in Tab. 2. As can be seen, there has been no research on applying SBO for mixed-integer multi-objective problems simultaneously. Due to the modular nature of SBO algorithms, it should be possible to combine mixed-integer surrogate models with multi-objective infill criteria to enable this.

Table 2 Overview of Surrogate-Based Optimization (SBO) approaches, with a focus on Mixed-Integer (MI), Multi-Objective (MO), Hierarchical (H) and constraint handling (g) approaches. Abbreviations: Expected Improvement (EI), Expected Violation (EV), Coordinate Perturbation (CP), Kriging with Partial Least Squares (KPLS), Probability of Feasibiliy (PoF), Probability of Improvement (PoI), Expected Hyper-Volume Improvement (EHVI), Euclidean EI (EEI), Expected Maximin Fitness Improvement (EMFI), Minimum Variance of Pareto Front (MVPF), Lower Confidence Bound (LCB), Upper Trust Bound (UTB). $\hat{y}$ and $\hat{s}$ refer to the function and variance estimates, respectively.

Publication	Year	Name	Surrogate	Infill criteria	MI	H	MO	g
Jones et al. [23]	1998	EGO	Kriging	EI				
Gutmann [34]	2001		RBF	$\hat{y}$				
Sasena [21]	2002	SuperEGO	Kriging	EI				EV
Gramacy et al. [35]	2004		Kriging	$\hat{s}$				
Regis et al. [38]	2013	DYCORS	RBF	CP				✓
Yi et al. [57]	2014		Kriging	$\hat{y}$ vs $\hat{s}$				$\hat{g}$
Bartoli et al. [58]	2016	SEGOMOE	KPLS, MOE	$\hat{y}$ vs EI (WB2S [59])				✓
Hemker [60]	2008	SurOpt MINLP	Kriging	$\hat{s}$	✓			✓
Chen et al. [61]	2015		Kriging	$\hat{y}$ vs $\hat{s}$	✓			✓
Baert et al. [55]	2015	MV-SBO	Kriging w/ MI kernels		✓			✓
Müller [29]	2016	MISO	RBF, Kriging	CP, EI, $\hat{y}$	✓			
Roy et al. [28]	2017	AMIEGO	KPLS	EI	✓			EV
Zaefferer et al. [10]	2018		Kriging w/ MI kernels	EI	✓	✓		
Horn et al. [52]	2019		Kriging w/ MI kernels	EI	✓	✓		
Pelamatti et al. [45]	2019		Kriging w/ MI kernels	EI	✓	✓		PoF
Priem et al. [39]	2020		Kriging	WB2S	✓			UTB
Jeong et al. [62]	2005	Multi-EGO	Kriging	MO EI			✓	✓
Knowles [63]	2006	ParEGO	Kriging	Scalarized EI			✓	✓
Emmerich et al. [64]	2006		Kriging	MO EI, PoI			✓	✓
Keane [43]	2006		Kriging	EEI			✓	✓
Hawe et al. [33]	2007		Kriging	Enhanced PoI			✓	
Svenson et al. [65]	2016		Kriging	EMFI			✓	
Datta et al. [26]	2016	SMES-RBF	RBF	$\hat{y}$			✓	✓
Palar et al. [66]	2017		Kriging	EHVI, EEI			✓	
Rahat et al. [67]	2017		Kriging	Minimum PoI			✓	
Mueller [68]	2017	SOCEMO	RBF	$\hat{y}$			✓	✓
dos Passos et al. [44]	2018		Kriging	EHVI, MVPF			✓	
Tian et al. [24]	2019		Kriging	$\hat{y}$ vs $\hat{s}$			✓	
Tran et al. [69]	2020	srMO-BO-3GP	Multi-level Kriging	Scalarized EI			✓	✓



(a) First infill criterion objective.

(b) Second infill criterion objective.

Figure 7 Modification of the Probability of Improvement (PoI) for use in multi-objective optimization, visualized for a normalized bi-objective solution space with 5 hypothetical points on the Pareto front, and a constant variance. The shown criterion objectives are to be minimized for finding infill points.

2. Infill Criteria for Multi-Objective Optimization Problems

Two main approaches can be identified for creating infill criteria for use with multi-objective optimization problems: modification of single-objective (SO) infill criteria, and the formulation of infill criteria specifically made for multi-objective (MO) optimization.

Single-objective Infill Criteria in Multi-Objective Optimization Existing single-objective infill criteria, like the Lower Confidence Bounds (LCB) criterion, can be used in MO optimization by running a multi-objective optimization on infill criteria evaluated for each objective separately [24]. This would then result in a Pareto-set of points balancing the infill criteria values for the objectives, naturally resulting in a set of infill points that balance the exploration and exploitation over the difference objectives. This principle can be used for any SO infill criterion that does not depend on the notion of current best objective value.

If an infill criterion depends on the current best objective values, for example the Expected Improvement (EI) or Probability of Improvement (PoI), a modification is needed to make them usable in MO optimization. This modification is needed, because if these criteria would simply be evaluated for each objective separately, only points near the edges of the current Pareto front would be selected for infill, as these points represent the respective best values of the objectives. One way to solve this problem is by scalarizing the optimization problem for various weighting vectors and apply the SO infill criteria on the scalarized problem. This approach is for example used by ParEGO [63]: objective weightings are randomly selected from evenly distributed weightings, and scalarization is then done using the augmented Tchebycheff function.

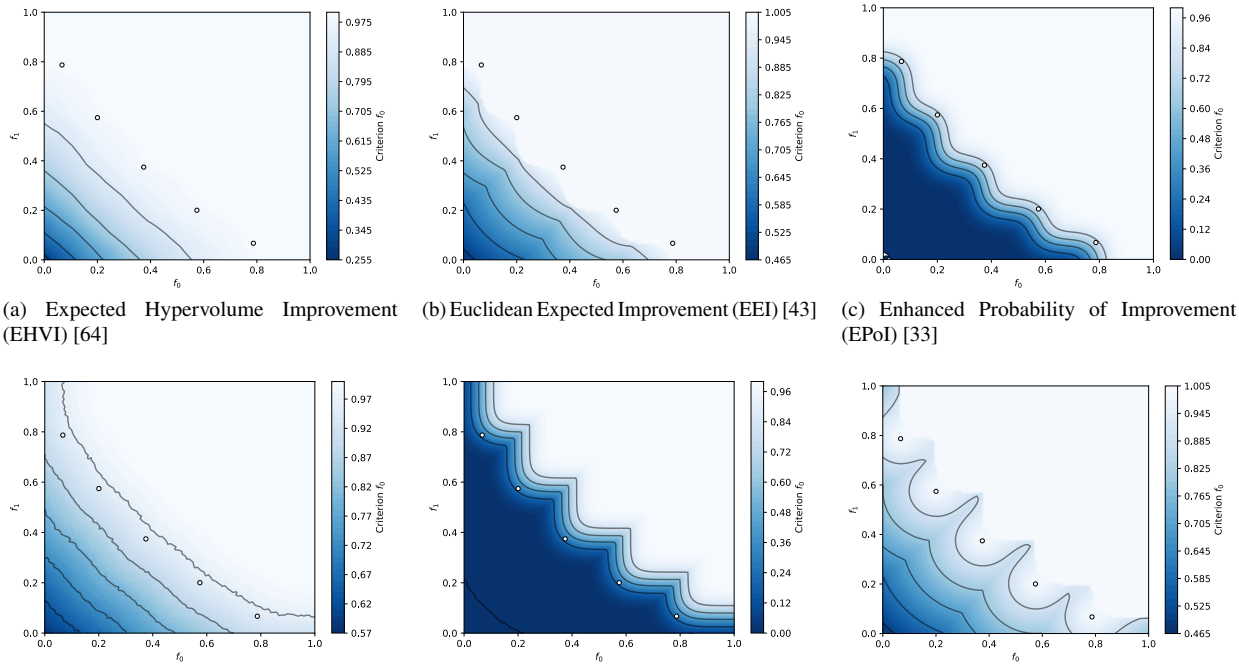
Another way to solve this problem is to instead of using the current best objective value, to use the objective values of the closest point on the Pareto front. This principle is used in the experiments in this paper, and it is shown in Fig. 7: it can be seen that around each point currently on the Pareto front, a slope towards the improvement of the Pareto front is constructed. For the shown hypothetical bi-objective problem, these two infill criterion objectives will therefore result in a well-diversified selection of infill points along the existing Pareto front. Compared to ParEGO, this approach prevents the scalarization of the multi-objective problem.

Special-Purpose Multi-Objective Infill Criteria Another strategy is by using infill criteria specifically designed for multi-objective (MO) problems. Most of these MO infill criteria attempt to transform the MO objective space into a single-objective space related to the distance to the current Pareto front. The goal is to find infill points that maximize some probability of finding an improvement over the current Pareto front. Table 3 compares the different MO infill criteria. Figure 8 visualizes the infill criteria objective spaces.

One of the earliest available criteria was the Expected Hypervolume Improvement (EHVI), as the hypervolume is a natural measure of the extent of the Pareto front. The problem with the EHVI is that there is no closed form for computing the expected improvement of the hypervolume, and that for dealing with the stochastic nature of the objective space (in terms of the mean predicted value and the variance), a Monte Carlo sampling approach is used. Computing the hypervolume itself is already expensive, and coupled with a Monte Carlo approach this leads to a factor

Table 3 Comparison of implemented multi-objective infill criteria. n_{pf} represents the number of points in the existing Pareto front.

Name	Publication	Year	Computation Cost	
			Relative	n_{pf} dependence
Expected Hypervolume Improvement (EHVI)	Emmerich et al. [64]	2006	~500	N/A
Euclidean Expected Improvement (EEI)	Keane [43]	2006	~10	n_{pf}^2
Enhanced Probability of Improvement (EPoI)	Hawe et al. [33]	2007	~10	n_{pf}^2
Expected Maximin Fitness Improvement (EMFI)	Svenson et al. [65]	2016	~10	$\sqrt{n_{pf}}$
Minimum Probability of Improvement (MPoI)	Rahat et al. [67]	2017	1	N/A
Minimum Variance of Pareto Front (MVPF)	dos Passos et al. [44]	2018	~0	N/A



(a) Expected Hypervolume Improvement (EHVI) [64]

(b) Euclidean Expected Improvement (EEI) [43]

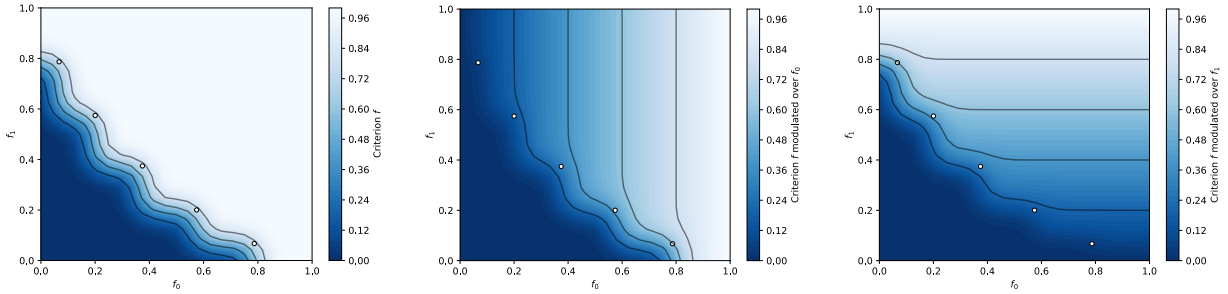
(c) Enhanced Probability of Improvement (EPoI) [33]

(d) Expected Maximin Fitness Improvement (EMFI) [65]

(e) Minimum Probability of Improvement (MPoI) [67]

(f) Minimum Euclidean Probability of Improvement (MEPoI)

Figure 8 Visualization of multi-objective infill criteria for a hypothetical normalized bi-objective solution space (f_0 and f_1) with 5 points on the Pareto front, and a constant variance.



(a) Infill criterion objective. (b) Modulated over f_0. (c) Modulated over f_1.

Figure 9 Modulation of a multi-objective infill criterion over the two underlying function objectives. This guides the selection of infill points along the currently existing Pareto front. Shown is the Enhanced Probability of Improvement infill criterion [33] for a hypothetical normalized bi-objective solution space (f_0 and f_1) with 5 points on the Pareto front, and a constant variance.

of approximately 500 between the computation of the EHVI and the MPoI. Another difficulty of the EHVI is that since the calculation of the HV does not extend beyond the edges of the currently known Pareto front, the EHVI criterion does not indicate improvement potential near these edges. This effect can be seen in Fig. 8a.

An additional consideration is the dependence of the computation time on the number of points in the Pareto front, as this might vary a lot between optimization runs, and the size of the Pareto front can grow very large if it is sufficiently diversified. Table 3 shows that over time, better MO infill criteria have been found that are quicker to compute, and have less dependence on the number of points in the existing Pareto front. The Minimum Variance of Pareto Front (MVPF) infill criterion derives its advantage from the selection of infill points after running a normal infill optimization based on the mean predicted objective values ($\hat{y}$). Therefore, the computation of the MVPF infill criterion actually costs no time other than querying the surrogate model.

MO extensions of the PoI and EI criteria are the Enhanced Probability of Improvement (EPoI) and Euclidean Expected Improvement (EEI) criteria, respectively. Similarly to their SO criteria, the EPoI and EEI are both based on the probability of improvement over the current Pareto front. The EEI then takes this probability and multiplies it with the centroid of the probability integral. The original derivation by Keane [43] shows a closed-form integral for the calculation of the EEI. However, due to computational cost concerns, the implementation used here replaces this integral with a Monte Carlo sampling approach, based on [33]. This also enables the use of these criteria for MO optimization problems with more than two objectives. Additionally, for the EEI, the closed-form centroid calculation has been replaced by the euclidean-based improvement measure from [70] to enable use in higher-dimensional MO problems. Both the EPoI and EEI criteria suffer from the same limitation as the EHVI criterion, in that near the edges of the Pareto front no improvement is indicated, as shown in Fig. 8b and 8c.

The implementation of the Expected Maximin Fitness Improvement EMFI) criterion is based on [65]. It deals with the predicted variance $\hat{s}$ by performing Monte Carlo sampling, leading to the non-smooth infill objective space as shown in Fig. 8d. In this case it is important to select an algorithm that can handle non-smooth and/or discontinuous objective spaces for solving the infill optimization problem.

The Minimum Probability of Improvement (MPoI) criteria has been [67] extended using the euclidean-based improvement measure from [70] as the Minimum Euclidean Probability of Improvement (MEPoI) metric. The objective spaces of the MPoI and MEPoI criteria are shown in Fig. 8e and 8f.

One issue with MO infill criteria, is that the MO infill search space is single-objective: there is only one point that maximizes a given MO infill criterion. It is possible to only search for one infill point per algorithm step, however then there is no guarantee that these new infill points are well-diversified along the existing Pareto front, especially since it is recognized that the infill criterion search landscape often is highly multi-model [67]. To enable the selection of multiple MO infill points per algorithm step, the single MO infill objective can be modulated by the predicted objective values to turn the single-objective MO infill criterion problem into a multi-objective infill problem. This principle is visualized in Fig. 9: an infill criterion (shown is the Enhanced Probability of Improvement [33]) is modulated over the two underlying problem objectives to turn the single-objective EPoI infill problem into a bi-objective problem, corresponding to the number of objectives of the underlying problem.

C. Multi-Objective Convergence Criteria

Normally, EA's and SBO's are executing with an evaluation budget [71]. This means they run for a given amount of function evaluations and stop once the budget has been exceeded, and then choose the best design point(s) from its search history to represent the best design points. Computational cost, however, is potentially wasted if the optimization algorithm converges before the budget has been exhausted: the algorithm then continues to search the design space, but no new better design points are discovered. Detecting convergence in single-objective optimization is usually done based on the value of the objective function itself: convergence is then assumed once objective function improvement per iteration falls below some threshold [8]. Applying the same principle to multi-objective optimization is not possible, as there the shape of the Pareto-front is the relevant aspect, not the current best values of the objectives. Note that convergence criteria can be independently applied to both Evolutionary Algorithms and Surrogate Based Optimization algorithms.

Common multi-objective convergence criteria include the Mutual Domination Rate (MDR) [71], Fitness Homogeneity Indicator (FHI) [72], Generational Distance (GD) [73], Spread [18], Hypervolume [74], Consolidation Ratio (CR) [75], Maximal Crowding Distance (MCD) and Steady Performance Indicator (SPI) [8].

IV. Investigation of Optimization Algorithm Effectiveness

In the previous section evolutionary algorithms, surrogate-based algorithms, and multi-objective convergence criteria that may be effective for solving system architecture optimization problems: black-box, mixed-discrete, hierarchical, multi-objective optimization problems. The first aspect to be investigated in this work is the effectiveness of different algorithms. Effectiveness is seen as distinct from efficiency, and relates to how well an optimization algorithm is able to find the global optimum (for single-objective optimization) or the Pareto-front (for multi-objective optimization), regardless of how quick it converges to the final result (i.e. efficiency) [7].

The optimization algorithms and convergence criteria discussed in the previous section have been implemented in Python using the multi-objective optimization framework pymoo* [15]. Surrogate-based optimization capabilities have been implemented using scikit-learn† [76] for Kriging models with mixed-integer and/or hierarchical kernels, and using SMT‡ [77] for RBF and continuous Kriging surrogate models. The optimization algorithm code and the code for performing the optimization experiments described in this section is published at [78]§. Readers should be able to reproduce the results presented in this paper.

All optimization concepts discussed in this section are implemented in Python using the *pymoo* multi-objective optimization framework [15]. The code for performing the optimization experiments that forms the basis for this paper is published online, to improve reproducibility.

A. Analytical Benchmark Problem

The analytical benchmark problem that will be used in this paper is based on the hierarchical Rosenbrock problem described by Pelamatti et al. [53]. This problem is based on the well-known high-dimensional Rosenbrock optimization problem, modified to have mixed-discrete and hierarchical design variables: two categorical variables determine the sub-problem the other variables act on (these are called *dimensional* variables in [53]). Based on which of the 4 sub-problems are selected, different continuous variables are active. In addition, several integer variables are added that map to continuous values for evaluation in by the original Rosenbrock function. The hierarchical Rosenbrock problem features 8 continuous, 3 integer, and 2 categorical design variables, 1 objective, and 2 inequality constraints.

Two modifications are applied to the hierarchical Rosenbrock problem of [53]: the problem is made multi-objective by adding a second objective, and another layer of categorical variables are added to increase the average amount of inactive variables. The second objective is calculated using

$$f_2(\boldsymbol{x}) = \left| 10 - \frac{f_1(\boldsymbol{x})}{40} \right|^2 + \sum_{i=1}^{4} 200\,(x_i + 1)^2\,, \tag{1}$$

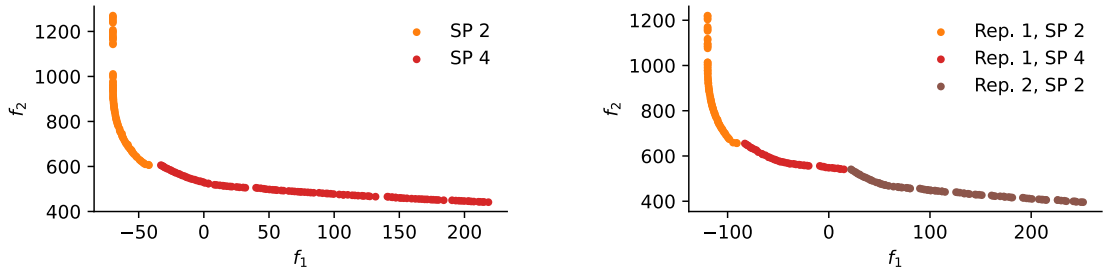
where f_1 is the first objective, $\boldsymbol{x}$ is the design vector, x_i are the first four continuous design variable as selected in the sub-problem. The Pareto front of the multi-objective hierarchical Rosenbrock problem is shown in Fig. 10a. As can be

*pymoo: https://pymoo.org/

†scikit-learn: https://scikit-learn.org/

‡SMT (Surrogate Modeling Toolbox): https://smt.readthedocs.io/

§jbussemaker/ArchitectureOptimizationExperiments



(a) Addition of a second objective f_2.

(b) Repetition of the problem.

Figure 10 Modifications of the hierarchical Rosenbrock problem [53]. "SP" refers to the sub-problem index (see [53]), "rep." to the replication index.

seen, it is composed of design points in two of the four original sub-problems.

The second modification is the repetition of the design variables, the creation of several random mappings from repeated variables to original variables, and the addition of a categorical variables selecting which mapping to use. The number of repetitions n_{rep} and number of mappings n_{map} are variable. Random mappings are created by randomly selecting which repetition an original design variable should map from for each of the design variables. The random state is fixed based on n_{rep} and n_{map}: this gives the impression of a random mapping, however it fixes the behavior of the problem between optimization runs, as needed for applying statistics on optimization algorithm performance. Additionally, each repetition modifies the objective values: they are moved along the existing Pareto front dimensions, so that a combined Pareto front is constructed that is constructed across repetitions. The result of this modification to the multi-objective hierarchical Rosenbrock problem is shown in Fig. 10b: it can be seen that the Pareto front consists of three parts, across two repetitions and two sub-problems.

The analytical architecture optimization benchmark problem used in this paper is therefore the repeated, multi-objective, hierarchical Rosenbrock problem, constructed with $n_{rep} = 2$ and $n_{map} = 2$. In total, this benchmark problem has 2 objectives, 2 inequality constraints, and 27 design variables, of which 16 continuous, 6 integer, and 5 categorical.

B. Experimental Setup

Different optimization algorithms will be compared with each other for their effectiveness at finding the Pareto front in the engine architecting problem. The different Kriging kernels and infill criteria for the Surrogate-Based Optimization (SBO) algorithm make for a large number of possible optimization algorithms. Therefore, before the actual algorithms will be compared with each other, the following two questions will be answered in order to down-select the SBO algorithms:

1) Which Kriging kernel performs best for system architecture optimization problems?
2) Should multi-objective infill criteria be used in their original form or their MO-modulated form?

The down-selected SBO algorithms will then be compared to each other and with the other algorithms for their effectiveness at finding the Pareto-front in the engine architecting design space. Each algorithm will be executed 8 times with the same parameter to obtain a feeling for the impact of stochastic elements in the algorithms (i.e. initial DOE, evolutionary operators, infill search). The initial Design of Experiments (DOE) for constructing the first surrogate models will be based on a Latin Hypercube Sampling (LHS) strategy. Algorithm effectiveness will be measured using the following metrics: ΔHV [66] to measure how closely the Pareto-front is approximated, and spread [18] to measure how evenly points are spread out along the Pareto-front. Constraints are handled by either rejecting newly found points with a constraint violation (for MOEA's), rejecting search points with predicted constraint violation (for RBF SBO), or by using the Probability of Feasibility (PoF) indicator (for Kriging SBO).

Table 4 presents the different algorithms that will be compared: an MOEA and multiple SBO algorithms. The algorithms are chosen such that the impact of different choices can be investigated: MOEA vs SBO, SBO surrogate model type, and infill criterion type. Note that the EHVI (Expected Hypervolume Improvement) infill criterion is not taken into account, due to its excessive computational cost (see also Table 3).

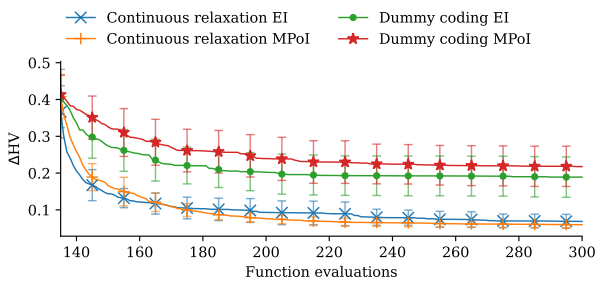
Table 4 Compared optimization algorithms.

Class	Name	Surrogate Model	Infill Criterion	Constraints
MOEA	NSGA2	N/A	N/A	Reject
RBF SBO	RBF $\hat{y}$	RBF	$\hat{y}$	Reject $\hat{g}$
	RBF $\hat{y}$-Dist		$\hat{y}$-Dist	
Kriging SBO with SO infill	Krg $\hat{y}$	Kriging	$\hat{y}$	PoF
	Krg $\hat{s}$		$\hat{s}$	
	Krg LCB		LCB	
	Krg EI		EI	
	Krg PoI		PoI	
Kriging SBO with MO infill	Krg EEI		EEI	
	Krg EPoI		EPoI	
	Krg MPoI		MPoI	
	Krg MEPoI		MEPoI	
	Krg EMFI		EMFI	
	Krg MVPF		MVPF	

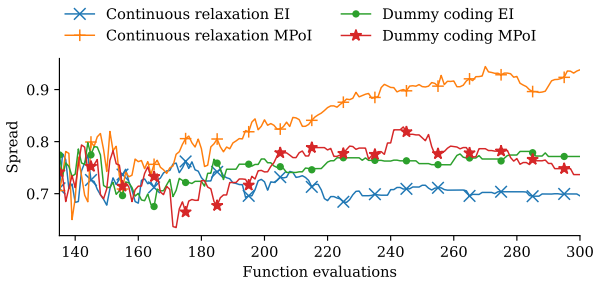
1. Kriging Surrogate Selection

The Kriging surrogate used in Surrogate Based Optimization (SBO) algorithms supports different kernels for representing similarity between two sampling points. These kernels are then used to construct the covariance matrix used in forming the prediction. As shown in Tab. 1, there are several ways of using Kriging for mixed-integer, hierarchical design spaces. This section aims to answer the questions of which Kriging kernel is the most appropriate for use for system architecting optimization problems. This will be done by comparing the different approaches shown in Tab. 1. Category-Wise Kriging is shown to be ineffective in SBO due to the requirement of enough samples for the remaining continuous design variables for each of the discrete combinations [53], and will therefore not be taken into account. Each of the Kriging surrogates with the different MI and hierarchical design space modeling approaches will be executed on the analytical benchmark problem with a budget of 300 function evaluations, with the Expected Improvement (EI) and Minimum Probability of Improvement (MPoI) infill criteria, and otherwise the same parameters as for the main comparison experiment. Kernels will be compared using the ΔHV and spread metrics. The DOE size is 135 points (5 times 27 design variables).

Figure 11 shows the result of this experiment, only for the continuous relaxation and dummy coding Kriging kernels.



(a) ΔHV metric (lower is better).



(b) Spread metric (lower is better).

Figure 11 Comparison of Kriging with continuous relaxation and dummy coding kernels on the analytical benchmark problem. Both Kriging models are compared using the EI (Expected Improvement) and MPoI (Minimum Probability of Improvement) infill criteria. Continuous relaxation provides faster convergence towards the Pareto front (ΔHV). However, the MPoI infill criterion with continuous relaxation results in a worse spread.

It shows that the continuous relaxation approach, i.e. treating discrete variables as continuous variables during training and infill optimization, results in the fastest convergence to the known Pareto front. The spread metrics show large differences between the selection of the infill criterion, which is investigated in a later experiment.

Other than the continuous relaxation and dummy coding approaches, all special-purpose mixed-integer and hierarchical Kriging kernels did not manage to meaningfully improve the ΔHV and spread as obtained by the initial DOE. Implementation of these kernels, as also available in the published code, has been validated using relevant analytical test problems used in the development of these kernels (e.g. from [10, 32, 53]. Unfortunately, all these test problems are single-objective problems, and of a relatively low dimensionality. An investigation into infill behavior has been performed, of which some results are shown in Fig. 18 (in the appendix). It shows that mixed-integer and hierarchical Kriging kernels result in less correlation between distance to the known Pareto front and the objective value of the infill criterion. Similar results have been demonstrated in [79]: they conclude that continuous SBO algorithms are well capable of, and often better at, solving discrete optimization problems.

Finally, it should be noted that compared to continuous relaxation, mixed-integer and hierarchical kernels take up to 100 times longer to train the model, and up to 20 times longer to evaluate an infill point. It is therefore concluded that the continuous relaxation Kriging approach is to be selected for high-dimensional architecture optimization problems, as this results in superior performance from a prediction, correlation between infill objective and distance to Pareto front, and training and infill time point of view.

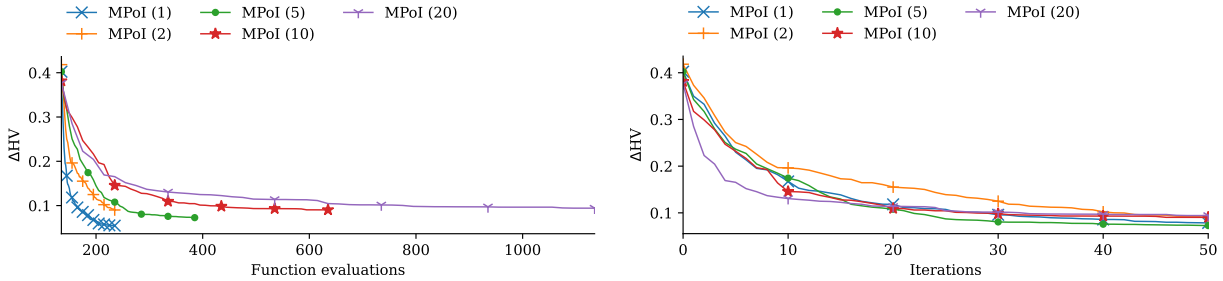
2. Infill Batch Size Investigation

The modulation of multi-objective (MO) infill criteria (see Fig. 9) offers potential improvements over directly using MO infill criteria due to the selection of multiple infill points per iteration. Selecting multiple infill points enables opportunities for a better-spread selection of infill points, requires less surrogate model training iterations, and allows time-savings in the expensive black-box evaluation if parallelization is possible. An experiment is performed to investigate the difference between one infill point per iteration and multiple infill points, and to give a suggestion on the amount of infill points per iteration. A comparison is made using the Expected Improvement (EI), Minimum Probability of Improvement (MPoI), and Minimum Variance of Pareto Front (MVPF) infill criteria with a Kriging surrogate model, and the RBF $\hat{y}$-Dist algorithm. EI is an infill criterion originally developed for use in single-objective optimization problems, and it naturally extended for use in multi-objective optimization by considering the EI criterion for each objective dimension separately (see Fig. 7). The MPoI infill criterion is an MO infill criterion, and therefore needs modulation if multiple infill points per iteration are needed (see Fig. 9). MVPF is an approach for selecting infill points from a population of candidate infill points after infill optimization, and naturally can be applied for selecting multiple infill points that are well-spread along the Pareto front. RBF $\hat{y}$-Dist attempts to trade-off exploration and exploitation by solving a multi-objective problem with exploration (finding the best expected value) and exploitation (finding the largest distance to existing design points) as objectives, and therefore also naturally lends itself to the selection of multiple infill points per iteration.

The main reason for using SBO is the reduction of the number of function evaluations to reduce optimization time. In this experiment it is assumed that perfect parallelization is possible, and that therefore the optimization time is only driven by the number of infill iterations, rather than the number of function evaluations. The continuous relaxation Kriging surrogate will be used, and otherwise all evaluation metrics are similar as for the experiments for the selection of the Kriging surrogate model.

The first result shows that if no parallel evaluation is possible, choosing 1 infill point per iteration converges to the Pareto front in the least amount of function evaluations needed, see Fig. 12. However, if the performance is compared by infill iterations, these differences disappear, and a larger number of infill points can lead to similar or faster convergence. An infill batch size of 5 is selected as the best size if multiple infill points are selected, due to the similar convergence speed (in iterations) as for 1 infill point, and a good final convergence to the Pareto front. For RBF $\hat{y}$-Dist, an infill batch size of 20 is selected, as there a larger infill size leads to a faster convergence (in terms of iterations), with less influence on the spread. The infill batch size has no influence on convergence if seen in terms of number of evaluations.

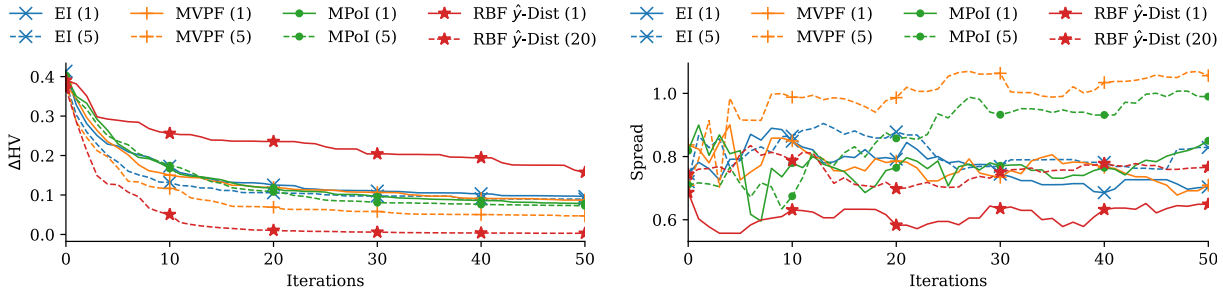
Figure 13 shows a comparison of the three infill criteria (EI, MPoI, and MVPF) for infill sizes of 1 and 5, and the RBF $\hat{y}$-Dist algorithm with infill sizes 1 and 20. It shows that in terms of ΔHV and spread, the larger infill sizes converge faster to the known Pareto front, however result in a worse spread of points along the Pareto front. Therefore there exists a trade-off between exploration and exploitation with regard to the infill batch size. An additional issue noted during the experiment was that Kriging surrogate models can have difficulty training when there are too many training points, and therefore it might be better to keep the amount of training points as low as possible. This effect does



(a) Comparison by function evaluations.

(b) Comparison by infill iterations.

Figure 12 Comparison of Kriging with continuous relaxation and MPoI (Minimum Probability of Improvement) infill criterion for different infill batch sizes. For an infill batch larger than 1, the modulated infill criterion is used (see Fig. 9). In terms of function evaluations, choosing only 1 infill point per iteration results in fastest convergence. If parallel evaluation is possible, infill sizes larger than 1 can result in faster convergence.



(a) ΔHV metric.

(b) Spread metric.

Figure 13 Comparison between 1 or 5 design points per infill iteration for different Kriging infill criteria, and the RBF $\hat{y}$-Dist algorithm. Larger infill batch sizes result in faster convergence (ΔHV) in terms of number of iterations, however with a worse spread along the Pareto front.

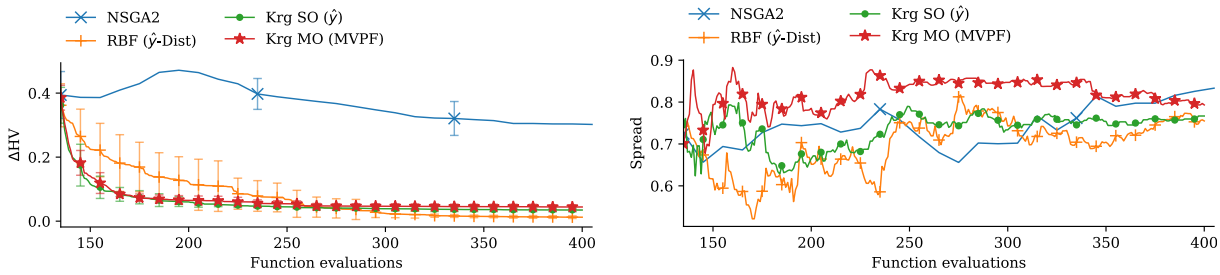
not apply to the RBF surrogate model. To conclude, selecting multiple infill points per iteration may result in a closer approximation of the Pareto front (ΔHV) in less number of iterations, however selecting 1 infill point per iteration:

- Converges faster in terms of function evaluations;
- Results in a better diversification (spread);
- Reduces the amount of training points, which improves training time and reduces convergence problems for Kriging surrogate models.

C. Results and Conclusions

Results on the effectiveness of the considered optimization algorithms (see Table 4) for system architecture optimization will be presented. NSGA2 is compared with various SBO algorithms. The Kriging SBO algorithms use the continuous relaxation kernel. One new point is selected per infill iteration for SBO (it is assumed no parallel evaluation of design points is possible), and an evaluation budget of 400 evaluations in total is used. NSGA2 is run with an offspring size of 10 per iteration. The DOE (or initial population) size is $n_{dv} \cdot 5$ (where n_{dv} is the number of design variables) individuals, which amounts to 135 points for the analytical benchmark problem.

Results of the algorithm comparison using the analytical benchmark problem are shown in Fig. 14. Best optimization algorithms in the four compared algorithm classes are shown: NSGA2, RBF $\hat{y}$-Dist, Kriging $\hat{y}$, and Kriging MVPF. More detailed views on all compared algorithms are shown in the appendix in Fig. 19. Results show that SBO converges faster than NSGA2. This is expected, as NSGA2 essentially is a randomized guided search algorithm [14]; these kinds of algorithms are powerful at finding a global optimum or Pareto front, however need many function evaluations for this. Within the SBO algorithms, the Kriging SBO algorithms ($\hat{y}$ and MVPF) initially converge the fastest, however are later



(a) ΔHV metric. (b) Spread metric.

Figure 14 Comparison between algorithm effectiveness on the analytical benchmark problem. Shown are the best algorithm in the four compared classes: MOEA, RBF SBO, Kriging SBO with SO-infill criteria, and Kriging SBO with MO-infill criteria. SBO converges faster than MOEA (NSGA2). Within the SBO algorithms, Kriging SBO with MO-infill criteria converges the fastest. Spread values of all algorithms are comparable, however RBF SBO and Kriging SBO with MO-infill result in the highest spreads.

overtaken by RBF SBO. All four best algorithms achieve low spreads, indicating a good distribution of points along the Pareto front. Finally, it should be noted that training and querying RBF surrogate models is much faster than Kriging: the infill time varies between 1 and 2 ms for Kriging, whereas it stays around 0.2 ms for RBF; the training time varies between 10 and 20 seconds for Kriging and is about 50 ms for RBF.

Table 5 Investigated convergence criteria. Δ in the convergence limit indicates that the rate of change of the (filtered) convergence criterion is checked, rather than the absolute value. EMA represents the Exponential Moving Average filter.

Name	Convergence criterion	Filter	Settings	Convergence limit
Budget	Evaluation budget			> 400
MDR	Mutual Domination Rate	EMA(n = 2)		< 10%
MGBM	Martí-García-Berlanga-Molina	Kalman(r = 0.1, q = 0.1)		< 10%
FHI	Fitness Homogeneity Indicator	EMA(n = 2)		Δ < 1e-4
GD	Generational Distance	EMA(n = 2)		Δ < 1e-3
IGD	Inverse Generational Distance	EMA(n = 2)		Δ < 1e-3
Spread	Spread	EMA(n = 5)		Δ < 1e-4
HV	Hypervolume	EMA(n = 2)		Δ < 1e-3
CR	Consolidation Ratio	EMA(n = 2)	n_{delta} = 1	> 80%
MCD	Maximal Crowding Distance	EMA(n = 2)		Δ < 5e-4
SPI	Steady Performance Indicator	EMA(n = 2)	n_{last} = 4	< 2%

V. Investigation of Optimization Algorithm Efficiency

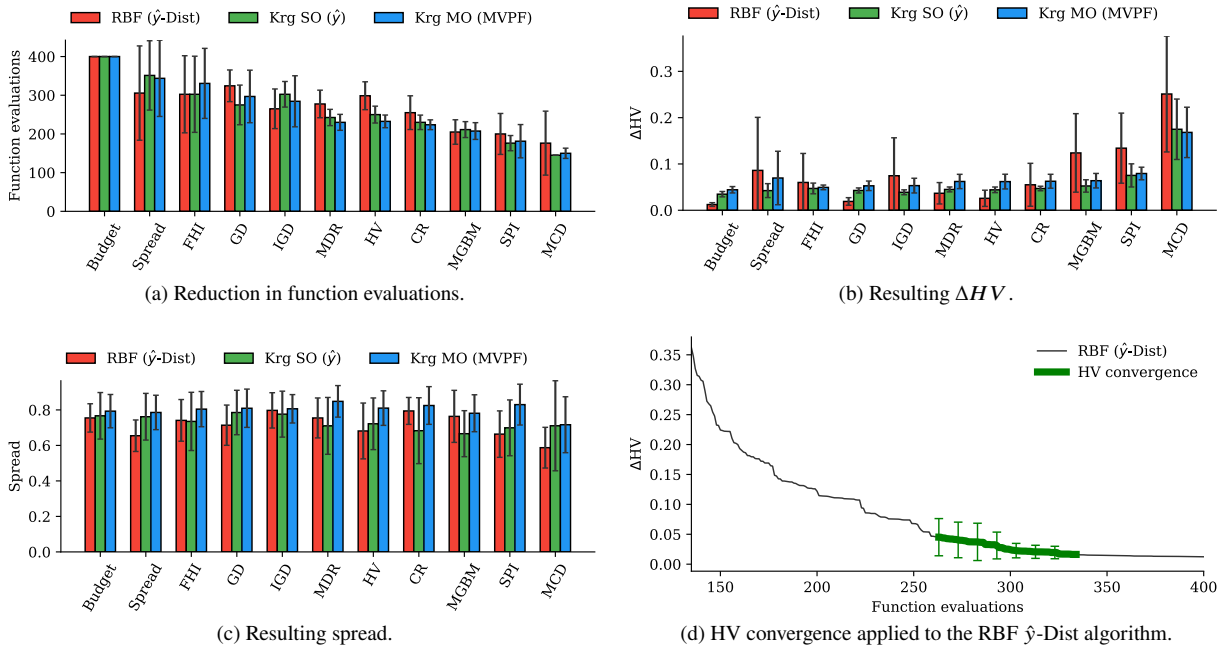
Next, optimization algorithm efficiency is investigated by comparing the amount of architecture evaluations needed for convergence towards the benchmark problem Pareto-front. Efficiency can also be defined in terms of other metrics, like memory usage or wall time, however since it is expected that one function evaluation takes significantly more time and resources than the optimization algorithm steps, this is seen as the most relevant parameter.

A. Experimental Setup

Optimization algorithm efficiency will be tested by applying several multi-objective convergence criteria, shown in Tab. 5. The results of the previous effectiveness investigation will be used to investigate the impact of the convergence criteria on the different optimization runs. Convergence criteria are evaluated every 10 function evaluations, to allow for iterations where no better points are found. The ΔHV and spread metrics are used to compare results. The efficiency experiment is applied to the best-performing algorithms from the effectiveness investigation: RBF $\hat{y}$-Dist, Kriging $\hat{y}$, and Kriging MVPF.

B. Results and Conclusions

Results on the efficiency experiment applied on the analytical benchmark problem are presented in Fig. 15. It shows that there is a clear trade-off between the number of function evaluations and the achieved quality of the multi-objective results in terms of distance to the Pareto front (ΔHV). The MDR (Mutual Domination Rate) and HV (Hypervolume) convergence metrics offer the largest reduction in function evaluations for the least reduction in ΔHV, most consistently. An example of the convergence zone in an optimization run is shown in Fig. 15d, which indeed seems to be shortly before the ΔHV remains steady. The spread is not affected significantly by a reduction in function evaluations. This can be explained by the lack of a clear trend in the spread metric throughout an optimization run, as can be seen in Fig. 14b.



(a) Reduction in function evaluations.

(b) Resulting ΔHV.

(c) Resulting spread.

(d) HV convergence applied to the RBF $\hat{y}$-Dist algorithm.

Figure 15 Comparison between algorithm efficiency on the analytical benchmark problem. Shown are average resulting number of evaluations, ΔHV, and spread for different algorithms and convergence criteria. There is a clear trade-off between remaining ΔHV and number of evaluations; the effect on the resulting spread is not significant.

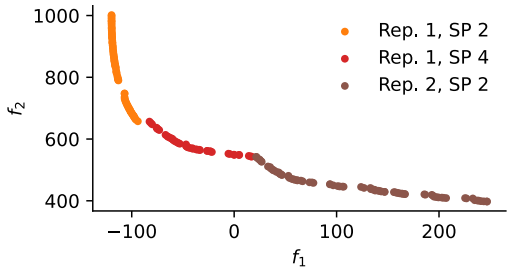
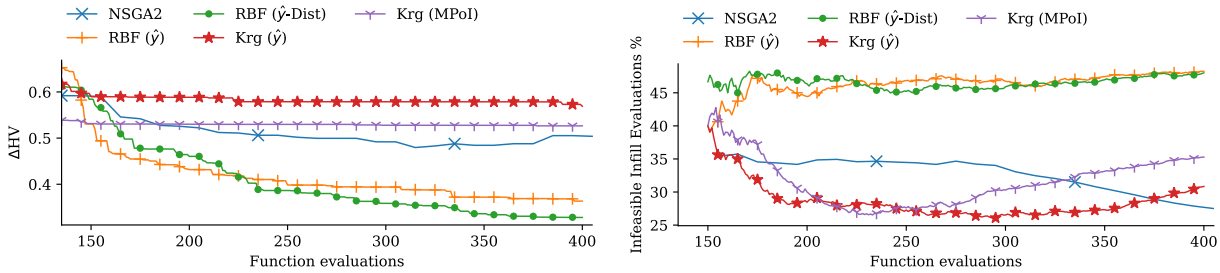


Figure 16 Modifications of the analytical benchmark problem presented in Section IV.A to include hidden constraints.



(a) ΔHV metric. (b) Infeasible infill evaluations.

Figure 17 Comparison between optimization algorithms on the analytical benchmark problem with hidden constraints. It shows that SBO algorithms are not able to find a significant improvement towards the Pareto front, if they are at all able to find an improvement.

VI. Surrogate-Based Optimization Subject to Hidden Constraints

Hidden constraints, also known as unknown, unspecified, and forgotten constraints, exhibit themselves when a design evaluations could not complete correctly and yields a meaningless results [12]. This is a common occurrence in simulation-based optimization due to the non-convergence of numerical solvers, for example because the solution is not physically possible [13]. Up to 60% of evaluations in an optimization run may result in violated hidden constraints. A realistic engine architecting problem published in [80] shows similar results: in the initial design of experiments of 205 design points, approximately 67% did not yield a meaningful result due to an unconverged design loop. The number of infeasible design points may be reduced by choosing a design space parameterization that reduces the chance for representing infeasible designs: for example, much effort has been invested in developing parametric geometry representations in the aircraft design discipline [81]. Despite these efforts, hidden constraints will still play a role in some simulation-based optimization problems.

Evolutionary algorithms like NSGA-II are able to solve problems where large parts of the design space are subject to hidden constraints, however need many function evaluations to do so. For example, in [80] a Pareto front of optimal engine architectures is found in 4000 function evaluations. Applying the SBO algorithms investigated in previous sections to the engine architecting problem shows that these algorithms are unable to produce a large improvement over the initial DOE. To support the claim that this is because of hidden constraints, the analytical benchmark problem used in previous sections is modified to also include hidden constraints. A hidden constraint is added that is violated if:

- g_2 is violated, or
- f_1 is within 3 of a multiple of 20 (i.e. $|0.5 - (f_1/20) \mod 1| > 0.35$), or
- f_2 is larger than 1000 and within 15 of a multiple of 100 (i.e. $|0.5 - (f_2/100) \mod 1| > 0.35$).

This results in a design space where in an initial design of experiments approximately 60% of design points yield an infeasible result, and a Pareto front with several gaps due to these hidden constraints, see Fig. 16. Running the same experiment as for determining algorithm effectiveness (i.e. 135 DOE points, 400 evaluations in total, 1 infill per iteration for SBO) shows that SBO algorithms are not significantly better than NSGA2 when exposed to hidden constraints, see Fig. 17. Kriging SBO algorithms are not able to improve the ΔHV metric at all. The RBF SBO algorithms are able to find an improvement, although one that stabilizes at $\Delta HV = 0.35$, whereas without hidden constraints the Pareto front is

approached much more closely. It is also shown that the number of infeasible infill evaluations is approximately 30% for NSGA2 and Kriging SBO algorithms, whereas it is up to 45% for RBF SBO algorithms.

At this point it can be concluded that the previously investigate SBO algorithms do not perform significantly better than NSGA2 for problems with hidden constraints. Hidden constraints need to be considered when developing and selecting algorithms for system architecture optimization.

VII. Conclusion and Outlook

System architecture optimization problems are hard to solve, because they in general suffer from a hierarchical design space, mixed-integer design variables, and multiple design objectives. A literature review discussing different types of optimization algorithms that are able to solve these kind of optimization problems is presented. Most suitable for this are Multi-Objective Evolutionary Algorithms (MOEA's). However, MOEA's suffer from needing many function evaluations for finding a Pareto-front. Surrogate Based Optimization (SBO) algorithms are specifically designed to reduce the number of function evaluations needed by constructing mathematical models of the design space. SBO algorithms use infill criteria to determine places in the design space to sample, with a trade-off between exploration and exploitation. In the past, research has been performed to enable SBO's to solve mixed-integer problems and multi-objective problems.

Optimization algorithms considered in the effectiveness (ability to find Pareto front) and efficiency (number of function evaluations needed to find Pareto front) experiments include NSGA2 (a MOEA) and various RBF- and Kriging-based SBO algorithms. RBF SBO does not expose information about estimation uncertainty, and therefore only directly sampling the surrogate ($\hat{y}$) or applying a distance-based exploration penalty ($\hat{y}$-Dist) can be used as infill. For Kriging SBO, many different single-objective and multi-objective infill criteria are investigated. Infill criteria are modified to be used in multi-objective optimization. The benchmark problem used in the experiments is based on a modification of a hierarchical Rosenbrock problem. All code implementing the various optimization algorithms, test problems, and experiments has been published at [78].

An investigation is performed into the best Kriging kernel for Kriging SBO. Next to continuous relaxation and dummy coding, several special-purpose mixed-integer and hierarchical Kriging kernels are investigated. Only the continuous relaxation and dummy coding kernels were able to approximate the known Pareto front, due to bad performance in modeling the high-dimensional hierarchical search space of the analytical benchmark problem. Continuous relaxation provided the best performance, both in optimization effectiveness and training and infill time. It should be investigated in more details why the special-purpose mixed-integer and hierarchical Kriging kernels were not effective for mixed-integer and hierarchical architecting problems. One area of focus should be the high-dimensionality of architecting problems in general. Additionally, it should be investigated whether machine learning models might be used as surrogate models for SBO [22], even though normally they might require a large number of samples for precise estimates [82].

Next, it is investigated how many infill points should be generated at every infill iteration. It is concluded that selecting multiple points per infill may result in a closer approximation of the Pareto front, however selecting 1 infill point per iteration: converges faster in terms of function evaluations, results in a better diversification along the Pareto front, and reduces the amount of training points in surrogate model.

Algorithm effectiveness is investigated by comparing NSGA2 with RBF and Kriging SBO. In case of SBO, 1 infill point per iteration is used, and for Kriging SBO the continuous relaxation kernel is used. It is shown that for SBO performs significantly better than NSGA2 in terms of function evaluations needed to find the Pareto front. For RBF SBO, the $\hat{y}$-Dist infill provides the best performance; for Kriging $\hat{y}$ (direct evaluation) and MVPF (Minimum Variance of the Pareto Front) provide best performance. Algorithm efficiency is investigated by applying various convergence criteria on the effectiveness results. It is shown that the MDR (Mutual Domination Rate) and HV (Hypervolume) convergence metrics are best able to reduce the number of function evaluations without reducing distance to the Pareto front too much.

Even though it is shown that SBO algorithms perform well on the mixed-discrete, hierarchical, multi-objective analytical test problem, they do not offer any performance improvement over NSGA2 when hidden constraints are added. Future research will focus on hidden constraints in architecture optimization, to enable the application on realistic simulation-based optimization problems that may be subject to analysis convergence issues. The engine benchmark problem from [80] will be used to test specific approaches for dealing with hidden constraints in surrogate-based optimization. Surrogate-based algorithms that can deal with mixed-discrete, hierarchical and multi-objective design space will be applied in various industry-provided use-cases in the AGILE 4.0 project. These use-cases feature evaluation times of multiple minutes to hours per design point, and will therefore serve as a good example of why surrogate-based optimization that can deal with architecture design spaces is needed.

This work considered no multi-level optimization approach, where for example continuous variables are optimized at an inner-loop compared to an outer-loop optimizing the discrete design variables [83]. Scalarization of the multi-objective problem was also not considered (e.g. [84]), and although often weights involved in scalarization lead to an element of arbitrariness, it might speed up convergence for certain classes of optimization problems.

Acknowledgments

The research presented in this paper has been performed in the framework of the AGILE 4.0 project (Towards Cyber-physical Collaborative Aircraft Development) and has received funding from the European Union Horizon 2020 Programme under grant agreement n◦ 815122.

References

[1] Ciampa, P., La Rocca, G., and Nagel, B., "A MBSE Approach to MDAO Systems for the Development of Complex Products," *AIAA Aviation Forum*, American Institute of Aeronautics and Astronautics, Reno, Nevada, 2020. doi:10.2514/6.2020-3150.

[2] Bussemaker, J., Ciampa, P., and Nagel, B., "System Architecture Design Space Exploration: An Approach to Modeling and Optimization," *AIAA AVIATION 2020 FORUM*, American Institute of Aeronautics and Astronautics, 2020. doi: 10.2514/6.2020-3172.

[3] Ciampa, P. D., and Nagel, B., "AGILE Paradigm: The next generation of collaborative MDO for the development of aeronautical systems," *Progress in Aerospace Sciences*, Vol. 119, 2020. doi:10.1016/j.paerosci.2020.100643.

[4] Ciampa, P., and Nagel, B., "Accelerating the Development of Complex Systems in Aeronautics via MBSE and MDAO: a Roadmap to Agility," *AIAA AVIATION 2021 FORUM*, American Institute of Aeronautics and Astronautics, Virtual Event, 2021.

[5] AGILE4.0, "AGILE4.0 Portal," Available: www.agile4.eu, 2019. Accessed June 2021.

[6] Crawley, E., Cameron, B., and Selva, D., *System architecture: strategy and product development for complex systems*, Pearson Education, 2015. doi:10.1007/978-1-4020-4399-4.

[7] Eligius M. T. Hendrix, B. G. T., *Introduction to Nonlinear and Global Optimization*, Springer-Verlag GmbH, 2010.

[8] Rudenko, O., and Schoenauer, M., "A Steady Performance Stopping Criterion for Pareto-based Evolutionary Algorithms," *6th International Multi-Objective Programming and Goal Programming Conference*, Hammamet, Tunisia, 2004.

[9] Miettinen, K., *Nonlinear Multiobjective Optimization*, Springer US, 1998.

[10] Zaefferer, M., and Horn, D., "A First Analysis of Kernels for Kriging-Based Optimization in Hierarchical Search Spaces," *Parallel Problem Solving from Nature, PPSN XI*, Vol. 1, edited by R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, Springer Berlin Heidelberg, Berlin, Heidelberg, 2018, pp. 399–410. doi:10.1007/978-3-319-99259-4_32.

[11] Hutter, F., and Osborne, M., "A Kernel for Hierarchical Parameter Spaces," ????

[12] MÃijller, J., and Day, M., "Surrogate Optimization of Computationally Expensive Black-Box Problems with Hidden Constraints," *INFORMS Journal on Computing*, Vol. 31, No. 4, 2019, pp. 689–702. doi:10.1287/ijoc.2018.0864.

[13] Digabel, S. L., and Wild, S. M., "A Taxonomy of Constraints in Simulation-Based Optimization," ????

[14] Glover, F., and Kochenberger, G., *Handbook of Metaheuristics*, Vol. 57, 2003. doi:10.1007/b101874.

[15] Blank, J., and Deb, K., "Pymoo: Multi-Objective Optimization in Python," *IEEE Access*, Vol. 8, 2020, pp. 89497–89509. doi:10.1109/access.2020.2990567.

[16] Cook, L. W., Willcox, K. E., and Jarrett, J. P., "Design optimization using multiple dominance relations," *International Journal for Numerical Methods in Engineering*, Vol. 121, No. 11, 2020, pp. 2481–2502. doi:10.1002/nme.6316.

[17] Zitzler, E., Deb, K., and Thiele, L., "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary computation*, Vol. 8, No. 2, 2000, pp. 173–95. doi:10.1162/106365600568202.

[18] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182–197. doi:10.1109/4235.996017.

[19] Nyew, H. M., Abdelkhalik, O., and Onder, N., "Structured-Chromosome Evolutionary Algorithms for Variable-Size Autonomous Interplanetary Trajectory Planning Optimization," *Journal of Aerospace Information Systems*, Vol. 12, No. 3, 2015, pp. 314–328. doi:10.2514/1.i010272.

[20] Gratton, S., and Vicente, L. N., "A Merit Function Approach for Direct Search," *SIAM Journal on Optimization*, Vol. 24, No. 4, 2014, pp. 1980–1998. doi:10.1137/130917661.

[21] Sasena, M. J., "Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations," Ph.D. thesis, University of Michigan, 2002.

[22] Chugh, T., Sindhya, K., Hakanen, J., and Miettinen, K., "A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms," *Soft Computing*, Vol. 23, No. 9, 2019, pp. 3137–3166. doi:10.1007/s00500-017-2965-0.

[23] Jones, D. R., Schonlau, M., and Welch, W. J., "Efficient Global Optimization of Expensive Black - Box Functions," *Journal of Global Optimization*, Vol. 13, 1998, pp. 455–492. doi:10.1023/A:1008306431147.

[24] Tian, J., Tan, Y., Zeng, J., Sun, C., and Jin, Y., "Multiobjective Infill Criterion Driven Gaussian Process-Assisted Particle Swarm Optimization of High-Dimensional Expensive Problems," *IEEE Transactions on Evolutionary Computation*, Vol. 23, No. 3, 2019, pp. 459–472. doi:10.1109/TEVC.2018.2869247.

[25] Rasmussen, C., and Williams, C., *Gaussian Processes for Machine Learning*, Vol. 14, 2006.

[26] Datta, R., and Regis, R. G., "A surrogate-assisted evolution strategy for constrained multi-objective optimization," *Expert Systems with Applications*, Vol. 57, 2016, pp. 270–284. doi:10.1016/j.eswa.2016.03.044.

[27] Bouhlel, M. A., Bartoli, N., Regis, R. G., Otsmane, A., and Morlier, J., "Efficient global optimization for high-dimensional constrained problems by using the Kriging models combined with the partial least squares method," *Engineering Optimization*, Vol. 50, No. 12, 2018, pp. 2038–2053. doi:10.1080/0305215x.2017.1419344.

[28] Roy, S., Moore, K., Hwang, J. T., Gray, J. S., Crossley, W. A., and Martins, J., "A Mixed Integer Efficient Global Optimization Algorithm for the Simultaneous Aircraft Allocation-Mission-Design Problem," *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2017, pp. 1–19. doi:10.2514/6.2017-1305.

[29] Müller, J., "MISO: mixed-integer surrogate optimization framework," *Optimization and Engineering*, Vol. 17, No. 1, 2016, pp. 177–203. doi:10.1007/s11081-015-9281-2.

[30] Garrido-Merchán, E., and Hernández-Lobato, D., "Dealing with categorical and integer-valued variables in Bayesian Optimization with Gaussian processes," *Neurocomputing*, Vol. 380, 2020, pp. 20–35. doi:10.1016/j.neucom.2019.11.004.

[31] Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E.-G., and Guerin, Y., "Efficient global optimization of constrained mixed variable problems," *Journal of Global Optimization*, Vol. 73, No. 3, 2019, pp. 583–613. doi:10.1007/s10898-018-0715-1.

[32] Zuniga, M. M., and Sinoquet, D., "Global optimization for mixed categorical-continuous variables based on Gaussian process models with a randomized categorical space exploration step," *INFOR: Information Systems and Operational Research*, Vol. 58, No. 2, 2020, pp. 310–341. doi:10.1080/03155986.2020.1730677.

[33] Hawe, G. I., and Sykulski, J. K., "An Enhanced Probability of Improvement Utility Function for Locating Pareto Optimal Solutions," *16th Conference on the Computation of Electromagnetic Fields, COMPUMAG, Aachen, Germany*, Vol. 3, 2007, pp. 965–966.

[34] Gutmann, H. M., "A Radial Basis Function Method for Global Optimization," *Journal of Global Optimization*, Vol. 19, No. 3, 2001, pp. 201–227. doi:10.1023/A:1011255519438.

[35] Gramacy, R. B., Lee, H. K. H., and Macready, W. G., "Parameter space exploration with Gaussian process trees," *Twenty-first international conference on Machine learning - ICML '04*, ACM Press, New York, New York, USA, 2004, p. 45. doi:10.1145/1015330.1015367.

[36] Horn, D., Wagner, T., Biermann, D., Weihs, C., and Bischl, B., "Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9018, Springer International Publishing, 2015, pp. 64–78. doi:10.1007/978-3-319-15934-8_5.

[37] Cox, D., and John, S., "A statistical method for global optimization," *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 1992. doi:10.1109/icsmc.1992.271617.

[38] Regis, R. G., and Shoemaker, C. A., "Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization," *Engineering Optimization*, Vol. 45, No. 5, 2013, pp. 529–555. doi:10.1080/0305215x.2012.687731.

[39] Priem, R., Bartoli, N., Diouane, Y., and Sgueglia, A., "Upper trust bound feasibility criterion for mixed constrained Bayesian optimization with application to aircraft design," *Aerospace Science and Technology*, Vol. 105, 2020, p. 105980. doi:10.1016/j.ast.2020.105980.

[40] Schonlau, M., Welch, W. J., and Jones, D. R., "Global versus local search in constrained optimization of computer models," *Institute of Mathematical Statistics Lecture Notes - Monograph Series*, Institute of Mathematical Statistics, 1998, pp. 11–25. doi:10.1214/lnms/1215456182.

[41] Lee, H., Gramacy, R., Linkletter, C., and Gray, G., "Optimization Subject to Hidden Constraints via Statistical Emulation," techreport UCSC-SOE-10-10, UC Santa Cruz, Apr. 2010.

[42] Rojas-Gonzalez, S., and Van Nieuwenhuyse, I., "A Survey on Kriging-Based Infill Algorithms for Multiobjective Simulation Optimization," *Computers & Operations Research*, Vol. 116, 2019, p. 104869. doi:10.1016/j.cor.2019.104869.

[43] Keane, A. J., "Statistical Improvement Criteria for Use in Multiobjective Design Optimization," *AIAA Journal*, Vol. 44, No. 4, 2006, pp. 879–891. doi:10.2514/1.16875.

[44] dos Passos, A. G., and Luersen, M. A., "Multi-objective optimization with Kriging surrogates using "moko", an open source package," *Latin American Journal of Solids and Structures*, Vol. 15, No. 10, 2018, pp. 1–17. doi:10.1590/1679-78254324.

[45] Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E.-G., and Guerin, Y., "Surrogate model based optimization of constrained mixed variable problems: application to the design of a launch vehicle thrust frame," *AIAA Scitech 2019 Forum*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2019. doi:10.2514/6.2019-1971.

[46] Herrera, M., Guglielmetti, A., Xiao, M., and Filomeno Coelho, R., "Metamodel-assisted optimization based on multiple kernel regression for mixed variables," *Structural and Multidisciplinary Optimization*, Vol. 49, No. 6, 2014, pp. 979–991. doi:10.1007/s00158-013-1029-z, URL http://link.springer.com/10.1007/s00158-013-1029-z.

[47] Hutter, F., "Automated Configurations of Algorithms for Solving Hard Computational Problems," Ph.D. thesis, University of British Columbia, Vancouver, Canada, Oct. 2009.

[48] Halstrup, M., "Black-box optimization of mixed discrete-continuous optimization problems," 2016. doi:10.17877/DE290R-17800.

[49] McCane, B., and Albert, M., "Distance functions for categorical and mixed variables," *Pattern Recognition Letters*, Vol. 29, No. 7, 2008, pp. 986–993. doi:10.1016/j.patrec.2008.01.021.

[50] Roustant, O., Padonou, E., Deville, Y., ClÃĬment, A., Perrin, G., Giorla, J., and Wynn, H., "Group kernels for Gaussian process metamodels with categorical inputs," 2018.

[51] Zhang, Y., Tao, S., Chen, W., and Apley, D. W., "A Latent Variable Approach to Gaussian Process Modeling with Qualitative and Quantitative Factors," *Technometrics*, Vol. 62, No. 3, 2019, pp. 291–302. doi:10.1080/00401706.2019.1638834.

[52] Horn, D., Stork, J., SchÃijßler, N., and Zaefferer, M., "Surrogates for hierarchical search spaces," *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM, 2019. doi:10.1145/3321707.3321765.

[53] Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E., and Guerin, Y., "Bayesian optimization of variable-size design space problems," *Optimization and Engineering*, 2020. doi:10.1007/s11081-020-09520-z.

[54] Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E.-G., and Guerin, Y., "Overview and Comparison of Gaussian Process-Based Surrogate Models for Mixed Continuous and Discrete Variables: Application on Aerospace Design Problems," *High-Performance Simulation-Based Optimization*, Studies in Computational Intelligence, Vol. 833, edited by T. Bartz-Beielstein, B. Filipič, P. Korošec, and E.-G. Talbi, Springer International Publishing, Cham, 2020, pp. 189–224. doi:10.1007/978-3-030-18764-4_9.

[55] Baert, L., Beauthier, C., Leborgne, M., and Lepot, I., "Surrogate-Based Optimisation for a Mixed-Variable Design Space: Proof of Concept and Opportunities for Turbomachinery Applications," *Volume 2C: Turbomachinery*, American Society of Mechanical Engineers, 2015, pp. 1–12. doi:10.1115/GT2015-43254.

[56] Wilson, D. R., and Martinez, T. R., "Improved Heterogeneous Distance Functions," *Journal of Artificial Intelligence Research*, Vol. 6, No. 1, 1997, pp. 1–34. doi:10.1613/jair.346.

[57] YI, S., Kwon, H. I., and Choi, S., "Efficient Global Optimization using a Multi-point and Multi-objective Infill Sampling Criteria," *52nd Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2014, pp. 1–19. doi:10.2514/6.2014-0898.

[58] Bartoli, N., Bouhlel, M.-A., Kurek, I., Lafage, R., Lefebvre, T., Morlier, J., Priem, R., Stilz, V., and Regis, R., "Improvement of efficient global optimization with application to aircraft wing design," *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, , No. June, 2016, pp. 1–28. doi:10.2514/6.2016-4001.

[59] Bartoli, N., Lefebvre, T., Dubreuil, S., Olivanti, R., Priem, R., Bons, N., Martins, J., and Morlier, J., "Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design," *Aerospace Science and Technology*, Vol. 90, 2019, pp. 85–102. doi:10.1016/j.ast.2019.03.041.

[60] Hemker, T., "Derivative Free Surrogate Optimization for Mixed-Integer Nonlinear Black Box Problems in Engineering," Ph.D. thesis, Technical University of Darmstadt, 2008.

[61] Chen, X. M., Zhu, Z., He, X., and Zhang, L., "Surrogate-Based Optimization for Solving a Mixed Integer Network Design Problem," *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 2497, No. 1, 2015, pp. 124–136. doi:10.3141/2497-13.

[62] Jeong, S., and Obayashi, S., "Efficient Global Optimization (EGO) for Multi-Objective Problem and Data Mining," *2005 IEEE Congress on Evolutionary Computation*, Vol. 3, IEEE, 2005, pp. 2138–2145. doi:10.1109/CEC.2005.1554959.

[63] Knowles, J., "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 1, 2006, pp. 50–66. doi:10.1109/TEVC.2005.851274.

[64] Emmerich, M., Giannakoglou, K., and Naujoks, B., "Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels," *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 4, 2006, pp. 421–439. doi:10.1109/TEVC.2005.859463.

[65] Svenson, J., and Santner, T., "Multiobjective optimization of expensive-to-evaluate deterministic computer simulator models," *Computational Statistics & Data Analysis*, Vol. 94, 2016, pp. 250–264. doi:10.1016/j.csda.2015.08.011.

[66] Palar, P. S., and Shimoyama, K., "On multi-objective efficient global optimization via universal Kriging surrogate model," *2017 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2017, pp. 621–628. doi:10.1109/CEC.2017.7969368.

[67] Rahat, A. A. M., Everson, R. M., and Fieldsend, J. E., "Alternative infill strategies for expensive multi-objective optimisation," *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '17*, ACM Press, New York, New York, USA, 2017, pp. 873–880. doi:10.1145/3071178.3071276.

[68] Müller, J., "SOCEMO: Surrogate Optimization of Computationally Expensive Multiobjective Problems," *INFORMS Journal on Computing*, Vol. 29, No. 4, 2017, pp. 581–596. doi:10.1287/ijoc.2017.0749.

[69] Tran, A., Eldred, M., McCann, S., and Wang, Y., "srMO-BO-3GP: A sequential regularized multi-objective constrained Bayesian optimization for design applications," 2020. URL `http://arxiv.org/pdf/2007.03502v2`.

[70] Parr, J. M., "Improvement Criteria for Constraint Handling and Multiobjective Optimization," phdthesis, University of Southampton, Feb. 2013. URL `https://eprints.soton.ac.uk/349978/1/JPARR-Thesis.pdf`.

[71] Martí, L., García, J., Berlanga, A., and Molina, J. M., "A stopping criterion for multi-objective optimization evolutionary algorithms," *Information Sciences*, Vol. 367-368, 2016, pp. 700–718. doi:10.1016/j.ins.2016.07.025.

[72] Marti, L., Garcia, J., Berlanga, A., and Molina, J. M., "A progress indicator for detecting success and failure in evolutionary multi-objective optimization," *IEEE Congress on Evolutionary Computation*, IEEE, 2010, pp. 1–8. doi:10.1109/CEC.2010.5586352.

[73] Van Veldhuizen, D., and Lamont, G., "Evolutionary Computation and Convergence to a Pareto Front," *Late Breaking Papers at the Genetic Programming 1998 Conference*, edited by J. R. Koza, Stanford University Bookstore, University of Wisconsin, Wisconsin, USA, 1998.

[74] Zitzler, E., and Thiele, L., "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, 1999, pp. 257–271.

[75] Goel, T., and Stander, N., "A non-dominance-based online stopping criterion for multi-objective evolutionary algorithms," *International Journal for Numerical Methods in Engineering*, Vol. 84, No. 6, 2010, pp. 661–684. doi:10.1002/nme.2909.

[76] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

[77] Bouhlel, M. A., Hwang, J. T., Bartoli, N., Lafage, R., Morlier, J., and Martins, J. R., "A Python surrogate modeling framework with derivatives," *Advances in Engineering Software*, Vol. 135, 2019, p. 102662. doi:10.1016/j.advengsoft.2019.03.005.

[78] Bussemaker, J., "jbussemaker/ArchitectureOptimizationExperiments: Release for AIAA paper," , 2021. doi:10.5281/ZENODO.4964341.

[79] Karlsson, R., Bliek, L., Verwer, S., and de Weerdt, M., "Continuous surrogate-based optimization algorithms are well-suited for expensive discrete problems," 2020.

[80] Bussemaker, J., Rocca, G. L., Ciampa, P., and Nagel, B., "System Architecture Optimization: An Open Source Multidisciplinary Turbofan Architecting Problem," *AIAA AVIATION 2021 FORUM*, American Institute of Aeronautics and Astronautics, Virtual Event, 2021.

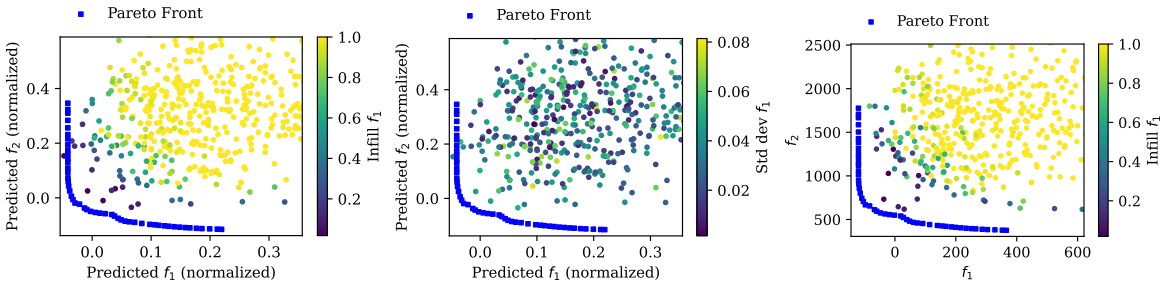
[81] Kulfan, B., and Bussoletti, J., ""Fundamental" Parameteric Geometry Representations for Aircraft Component Shapes," *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, 2006. doi:10.2514/6.2006-6948.

[82] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, MIT Press, 2016.

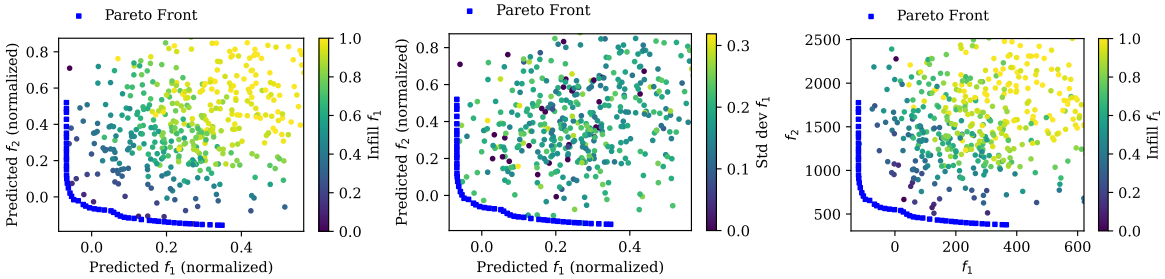
[83] Barjhoux, P.-J., Diouane, Y., Grihon, S., Bettebghor, D., and Morlier, J., "A bi-level methodology for solving large-scale mixed categorical structural optimization," *Structural and Multidisciplinary Optimization*, Vol. 62, No. 1, 2020, pp. 337–351. doi:10.1007/s00158-020-02491-w.

[84] Désidéri, J.-A., "Multiple-Gradient Descent Algorithm (MGDA)," Tech. rep., INRIA, 2009.

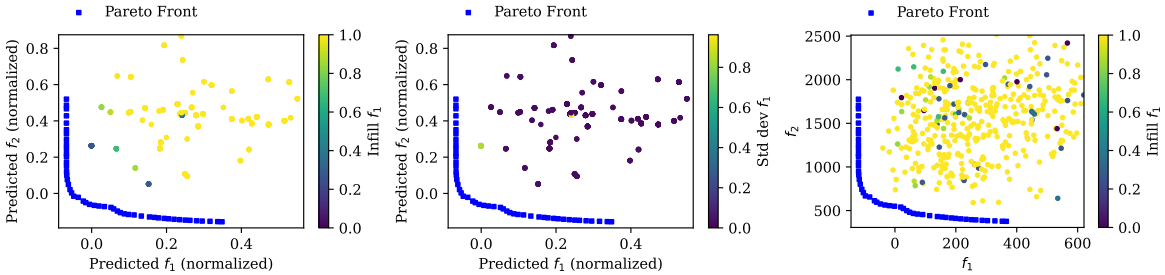
Appendix: Extended Results



(a) Kriging with continuous relaxation approach. It shows a good prediction of objective values f (compare left and right), and a good correlation between distance to the Pareto front and infill objective value (right figure). The standard deviation values are relatively low (up to 8%).

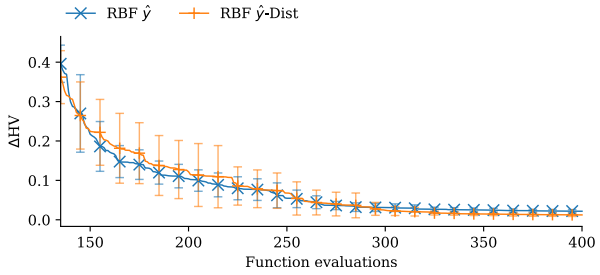


(b) Kriging with Indefinite Conditional (Ico) kernel. Prediction of objective values is less accurate than Kriging with continuous relaxation, due to higher standard deviation values (up to 40%). Higher standard deviation also leads to more spread of infill objective values further away from the Pareto front.

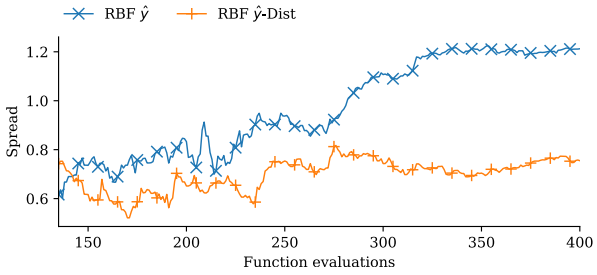


(c) Kriging with Sub-Problem-Wise Compound Symmetry kernel (SPW+CS). Only a small number of different objective values f are predicted: the left and middle figures show many overlaid points. The right figure shows low correlation between distance to Pareto front and infill objective value.

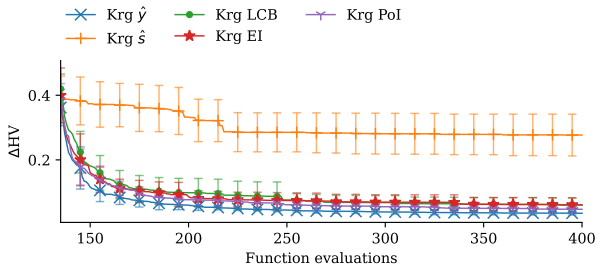
Figure 18 Comparison of infill performance for different Kriging kernels on the analytical benchmark problem using the MPoI infill criterion. Shown are from left to right, the infill objective value (to be minimized) for the predicted f values, the predicted standard deviations for the predicted f values, and the infill objective value for the real f values. The surrogate model has been trained using a DOE of 135 points, and 500 infill points are generated.



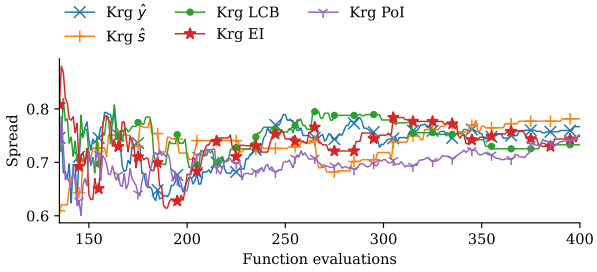
(a) RBF SBO (ΔHV).



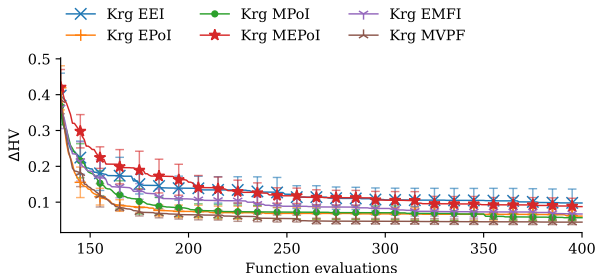
(b) RBF SBO (spread).



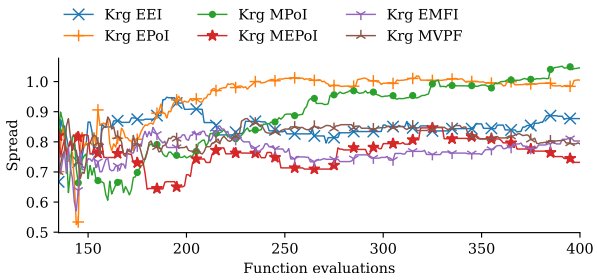
(c) Kriging SBO with SO-infill (ΔHV).



(d) Kriging SBO with SO-infill (spread).



(e) Kriging SBO with MO-infill (ΔHV).



(f) Kriging SBO with MO-infill (spread).

Figure 19 Comparison between algorithm effectiveness on the analytical benchmark problem. Focus on different SBO algorithm classes. The Kriging SBO algorithm are all based on continuous relaxation.