# Surrogate-Based Optimization of System Architectures Subject to Hidden Constraints

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

Paul Saves[†], Nathalie Bartoli[‡], Thierry Lefebvre[§]
*DTIS, ONERA, Université de Toulouse, 31000 Toulouse, France*
*Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, 31000, Toulouse, France*

Björn Nagel[¶]
*DLR (German Aerospace Center), Institute of System Architectures in Aeronautics, Hamburg, Germany*

**System Architecture Optimization (SAO) can support the design of novel architectures by formulating the architecting process as an optimization problem. The exploration of novel architectures requires physics-based simulation due to a lack of prior experience to start from, which introduces two specific challenges for optimization algorithms: evaluations become more expensive (in time) and evaluations might fail. The former challenge is addressed by Surrogate-Based Optimization (SBO) algorithms, in particular Bayesian Optimization (BO) using Gaussian Process (GP) models. An overview is provided of how BO can deal with challenges specific to architecture optimization, such as design variable hierarchy and multiple objectives: specific measures include ensemble infills and a hierarchical sampling algorithm. Evaluations might fail due to non-convergence of underlying solvers or infeasible geometry in certain areas of the design space. Such failed evaluations, also known as hidden constraints, pose a particular challenge to SBO/BO, as the surrogate model cannot be trained on empty results. This work investigates various strategies for satisfying hidden constraints in BO algorithms. Three high-level strategies are identified: *rejection* of failed points from the training set, *replacing* failed points based on viable (non-failed) points, and *predicting* the failure region. Through investigations on a set of test problems including a jet engine architecture optimization problem, it is shown that best performance is achieved with a mixed-discrete GP to predict the Probability of Viability (PoV), and by ensuring selected infill points satisfy some minimum PoV threshold. This strategy is demonstrated by solving a jet engine architecture problem that features at 50% failure rate and could not previously be solved by a BO algorithm. The developed BO algorithm and used test problems are available in the open-source Python library SBArchOpt.**

## Nomenclature

| | | |
|---|---|---|
| BO | = | Bayesian Optimization |
| $c_h$ | = | Hidden constraint |
| $c_{v,l}$ | = | Value constraint $l$ |
| DoE | = | Design of Experiments |
| FR | = | Failure rate |
| $f_m$ | = | Objective function $m$ |
| GP | = | Gaussian Process |
| $g_k$ | = | Inequality constraint $k$ |
| HV | = | Hypervolume |

[*]Researcher, DDP Group, Department of Digital Methods for System Architecting, jasper.bussemaker@dlr.de
[†]Research engineer, Information Processing and Systems Department, paul.saves@onera.fr
[‡]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
[¶]Institute Director, Institute of System Architectures in Aeronautics, Hamburg, bjoern.nagel@dlr.de

| | | |
|---|---|---|
| IR | = | Imputation ratio (*d* subscript: discrete IR; *c* subscript: continuous IR) |
| $k_{\text{doe}}$ | = | DoE multiplier |
| MD | = | Mixed-Discrete |
| MRD | = | Maximum rate diversity |
| $n_{c_v}$ | = | Number of value constraints |
| $n_f$ | = | Number of objectives |
| $n_g$ | = | Number of inequality constraints |
| $n_{\text{batch}}$ | = | Batch size for infill points |
| $n_{\text{doe}}$ | = | Design of Experiments size |
| $n_{\text{infill}}$ | = | Total number of infill points generated |
| $n_{\text{parallel}}$ | = | Maximum number of parallel evaluations |
| $n_{\text{valid,discr}}$ | = | Number of valid discrete design vectors in an optimization problem |
| $n_x$ | = | Number of design variables |
| $n_{x_c}$ | = | Number of continuous design variables |
| $n_{x_d}$ | = | Number of discrete design variables |
| $N_j$ | = | Number of options for discrete variable $j$ |
| RFC | = | Random Forest Classifier |
| SAO | = | System Architecture Optimization |
| SBO | = | Surrogate-Based Optimization |
| $\hat{s}$ | = | Uncertainty estimate by a surrogate model |
| $\boldsymbol{x}$ | = | Design vector |
| $x_{c,i}$ | = | Continuous design variable $i$ |
| $x_{d,j}$ | = | Discrete design variable $j$ |
| $\hat{y}$ | = | Function value estimate by a surrogate model |
| $\delta_i$ | = | Activeness function for design variable $i$ |

## I. Introduction

System Architecture Optimization (SAO) is an emerging field with the goal of partial automation of designing system architectures by formulating the architecting process as a numerical optimization problem [1]. This way, a larger design space can be explored than is possible nowadays [2], and bias towards conventional solutions can be reduced [3]. This work deals with the aspect of developing efficient optimization algorithms for solving architecture optimization problems. For more background on architecture optimization, including how to model architecture design spaces and how to connect to evaluation code, the interested reader is referred to [4, 5].

SAO problems are subject to various aspects that make them challenging to solve: the presence of *mixed-discrete* design variables stemming from architectural decisions, *multiple objectives* to optimize for stemming from conflicting stakeholder needs [6], and the fact that the evaluation functions have to be treated as *black-box* functions [7]. Another salient feature of SAO is design variable *hierarchy* [8]: the phenomenon that some design variables might decide whether other design variables are active or not, and might restrict the available options of other design variables. Consider for example a launch vehicle design problem where both the number of rocket stages and the fuel type of each stage are design variables [9]: it follows that the selection of the number of stages determines whether some of the fuel-type selection variables are active or not. Evolutionary algorithms such as NSGA-II are well-suited for solving such optimization algorithms and are currently available for use [1, 10–12]. When searching for novel system architectures to solve for example sustainability challenges, design expertise based on historical regression analysis may be not available, however. In general when designing new architectures, physics-based simulation is needed to accurately predict the performance of architectural candidates. This trend has two major consequences for optimization algorithms [12]:

1) Evaluating a design candidate is *expensive*, and therefore the optimizer should aim to find the optimum (or Pareto front) is as little function evaluations as possible;
2) Evaluation of a design candidate can *fail*, for example because of unstable underlying equations, infeasible physics, or infeasible geometry.

To deal with the challenges of *expensive* evaluation, Surrogate Based Optimization (SBO) algorithms have been developed. These algorithms train a regression or interpolation surrogate model of the objective $f(\mathbf{x})$ and constraint $g(\mathbf{x})$ functions with respect to the design variables $\mathbf{x}$. These models are then used to generate *infill* points for exploring the design space and search for the optimum or Pareto front. The principle of SBO is visualized in Figure 1. An especially powerful type of SBO is Bayesian Optimization (BO), where Gaussian Process (GP) models are used to provide not only an estimate of the function value $\hat{y}(\mathbf{x})$ but also an estimate of the associated uncertainty $\hat{s}(\mathbf{x})$. BO is more effective at solving optimization problems than SBO algorithms with other surrogate models [13], and has been extended to work for constrained [14, 15], mixed-discrete [13, 16], hierarchical [17–19] and multi-objective [20, 21] problems. The combination of all aspects has been demonstrated for SAO [12, 22, 23].

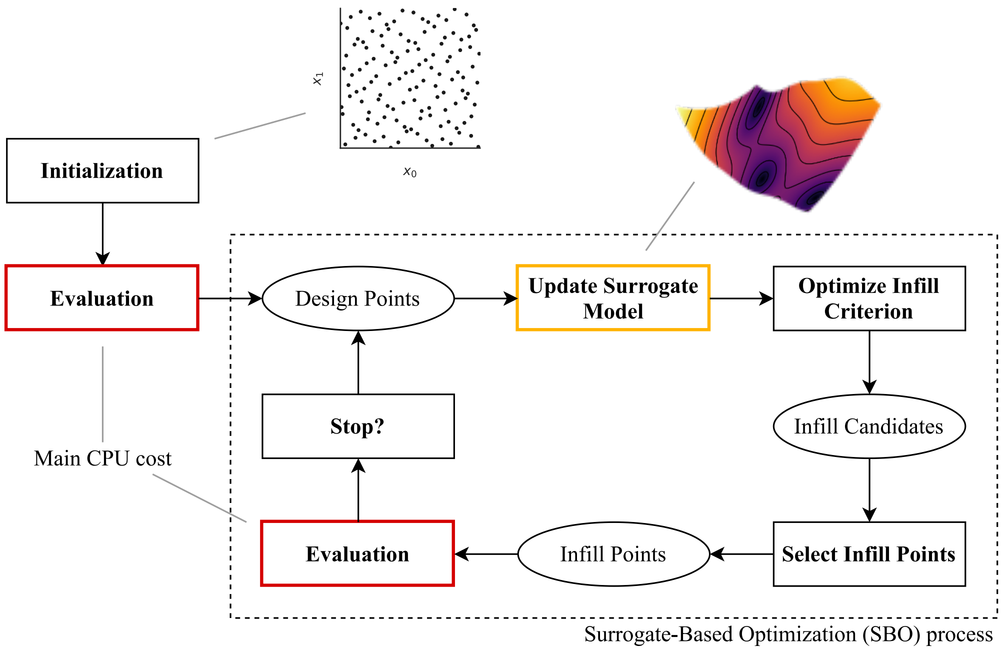


**Fig. 1 Principle of Surrogate-Based Optimization (SBO), reproduced from [12].**

Evaluation failures manifest themselves as so-called *hidden constraints* [24], which are also known as unknown, unspecified, forgotten, virtual, and crash constraints [24, 25]. When a hidden constraint is violated (i.e. an evaluation has failed), the objective $f$ and constraint $g$ functions are assigned NaN (Not-a-Number). In the taxonomy of Le Digabel & Wild [25] hidden constraints are classified as NUSH constraints: Non-quantifiable (the degree of constraint violation is not available, only whether it is violated or not), Unrelaxable (violating the constraint yields no meaningful information about the design space), Simulation (a simulation must be run in order to find out the status), and Hidden (the existence of the constraint is not known before solving the problem). Hidden constraints are assumed to be deterministic, mainly meaning that we do not consider evaluation failures due to temporary problems such as software license availability or network problems. Finding out whether the hidden constraint is violated may take as long, if not longer, than a non-failed evaluation [24]. The design space can be separated in regions with satisfied and violated hidden constraints, denoted as *viable* and *failed* regions, respectively. Depending on the design problem, the failed region can take up a significant part of the design space: Krengel & Hepperle report up to 60% for a wing design problem [26], and for an airfoil design problem Forrester et al. up to 82% failed points [27]. Hidden constraints pose a particular challenge to SBO algorithms: evaluation failures assign NaN (not-a-number) values to objective and constraints, and it is not possible to train a surrogate model on NaN's [10].

A review of dealing with hierarchical optimization challenges for BO will be given in Section II, which is extended to hidden constraints in Section III. The paper finishes with a demonstration of a jet engine optimization problem in Section IV and the conclusions in Section V. All test problems used in this work are available in the open-source SBArchOpt* library [28]. Code used to run the experiments is available open-source as well, at github.com/jbussemaker/ArchitectureOptimizationExperiments (hidden-constraints branch).

*https://sbarchopt.readthedocs.io/

## II. Bayesian Optimization in Hierarchical Design Spaces

Before diving into strategies for dealing with hidden constraints, this section reviews strategies for dealing with the other challenges of SAO using Bayesian Optimization (BO) algorithms as originally published in [12], and introduces the BO algorithm used in the rest of this work. The algorithm is implemented and available as ArchSBO in SBArchOpt.

**Gaussian Process Model Selection** For non-hierarchical problems, the mixed-discrete Gaussian Process (GP) models (also known as Kriging models) implemented in the Surrogate Model Toolbox (SMT)[†] are used. These models support continuous, integer and categorical variables through kernels developed by Saves et al. [19, 29]. GP models that support hierarchical variables have been developed in [19] and extended to also support categorical hierarchical variables in [12].

A disadvantage of GP models is that it becomes computationally expensive to train these models for high-dimensional spaces [30]. One way of dealing with this is to define a feature space of lower dimension to train the GP, and provide a conversion between input and feature spaces [31]. One specific method is Kriging with Partial Least Squares (KPLS) [32], which has recently been extended to work with discrete variables [33**?** , 34] and is implemented in SMT [19].

**Selecting Infill Points** Infill points are selected by optimizing infill criteria (also known as acquisition functions), and selecting the best one for a wide range of optimization problems influences BO performance significantly. The BO algorithm uses an enable of infills, with the size of the infill batch set to the maximum number of parallel equations: $n_{\text{batch}} = n_{\text{parallel}}$. An infill ensemble combines multiple underlying infills into a multi-objective infill problem and thereby provides two main advantages [35]:

1) The selected infill points are a compromise of the underlying infills, thereby mitigating problems with different infill criteria suggesting to explore wildly different parts of the design space [36];
2) Since the infill problem itself results in a Pareto front, it is easy to select multiple infill points for batch evaluation without needing to retrain surrogate models [37].

The infill ensemble is built-up of the following underlying infills:

- For single-objective optimization: Lower Confidence Bound (LCB) [38], Expected Improvement (EI) [7], and Probability of Improvement (PoI) [39];
- For multi-objective optimization: Minimum PoI (MPoI) [40] and Minimum Euclidean PoI (MEPoI) [22].

Selecting the best infill points is an optimization problem itself, and with the same design space as the SAO problem. To successfully deal with mixed-discrete hierarchical variables, the following sequential optimization procedure is applied:

1) Apply NSGA-II to search the mixed-discrete, hierarchical design space, solving the multi-objective infill problem:

$$\begin{aligned} \text{minimize} \quad & f_{\text{infill,m}}(\boldsymbol{x}_d, \boldsymbol{x}_c) \\ \text{w.r.t.} \quad & \boldsymbol{x}_d, \boldsymbol{x}_c \\ \text{subject to} \quad & \hat{g}_k(\boldsymbol{x}) \leq 0 \end{aligned}$$

   where $f_{\text{infill,m}}$ represents infill criterion $m$, $\boldsymbol{x}_d$ and $\boldsymbol{x}_c$ represent the discrete and continuous design variables, and $\hat{g}_k$ represents the predicted mean of constraint function $k$. Infill objectives $f_{\text{infill,m}}$ are normalized and inverted such that they become minimization objectives.
2) Select $n_{\text{batch}}$ points $\boldsymbol{x}_{\text{sel}}$ from the resulting Pareto front according to:
   - If $n_{\text{batch}} = 1$, select a random point;
   - If $n_{\text{batch}} > 1$, select the points with the lowest crowding distance as used in NSGA-II [41].
3) For each selected point $\boldsymbol{x}_{sel,i}$, improve the active continuous variables by solving following single-objective problem using SLSQP:

$$\begin{aligned} \text{minimize} \quad & f_{\text{impr}}(\boldsymbol{x}) \\ \text{w.r.t.} \quad & \boldsymbol{x}_c \cap \delta\left(\boldsymbol{x}_{\text{sel,i}}\right) \\ \text{subject to} \quad & \hat{g}_k(\boldsymbol{x}) \leq 0 \end{aligned}$$

[†]https://smt.readthedocs.io/

where $\boldsymbol{x}_c \cap \delta\left(\boldsymbol{x}_{\text{sel,i}}\right)$ represents the active continuous design variables at point $\boldsymbol{x}_{\text{sel,i}}$, and $f_{\text{impr}}$ the scalar improvement objective:

$$\begin{aligned}
\Delta f_{\text{infill,m}}(\boldsymbol{x}) &= f_{\text{infill,m}}(\boldsymbol{x}) - f_{\text{infill,m}}(\boldsymbol{x}_{sel,i}) \\
f_{\text{deviation}}(\boldsymbol{x}) &= 100\left(\max_{\text{m}} \Delta f_{\text{infill,m}}(\boldsymbol{x}) - \min_{\text{m}} \Delta f_{\text{infill,m}}(\boldsymbol{x})\right)^2 \\
f_{\text{impr}}(\boldsymbol{x}) &= \sum_{\text{m}} \left(\Delta f_{\text{infill,m}}(\boldsymbol{x})\right) + f_{\text{deviation}}(\boldsymbol{x})
\end{aligned}$$

This objective promotes improvement in direction of the negative unit vector, so that all underlying infill objectives are improved simultaneously while not deviating too much from the original $x_{\text{sel,i}}$ to maintain batch diversity.

**Sampling Hierarchical Design Spaces** When sampling hierarchical design space for creating the initial Design of Experiments (DoE), care must be taken that all sections of the design space are sufficiently represented because of a possible large discrepancy between how often individual discrete design variable values appear [6]. For example, the discrete design vectors representing pure jet engines (vs turbofan engines) in the jet engine design problem of Bussemaker et al. [10] only make up 0.02% of the design space. This discrepancy can be measured for each discrete design variable by the *rate diversity* $\text{RD}_j$ metric, or at the problem level by the *max rate diversity* MRD, defined as:

$$\text{Rates}_j = \left\{\text{Rate}(j, \delta_j = 0), \text{Rate}(j, 0), .., \text{Rate}(j, N_j - 1)\right\} \tag{1}$$

$$\text{RD}_j = \max \text{Rates}_j - \min \text{Rates}_j \tag{2}$$

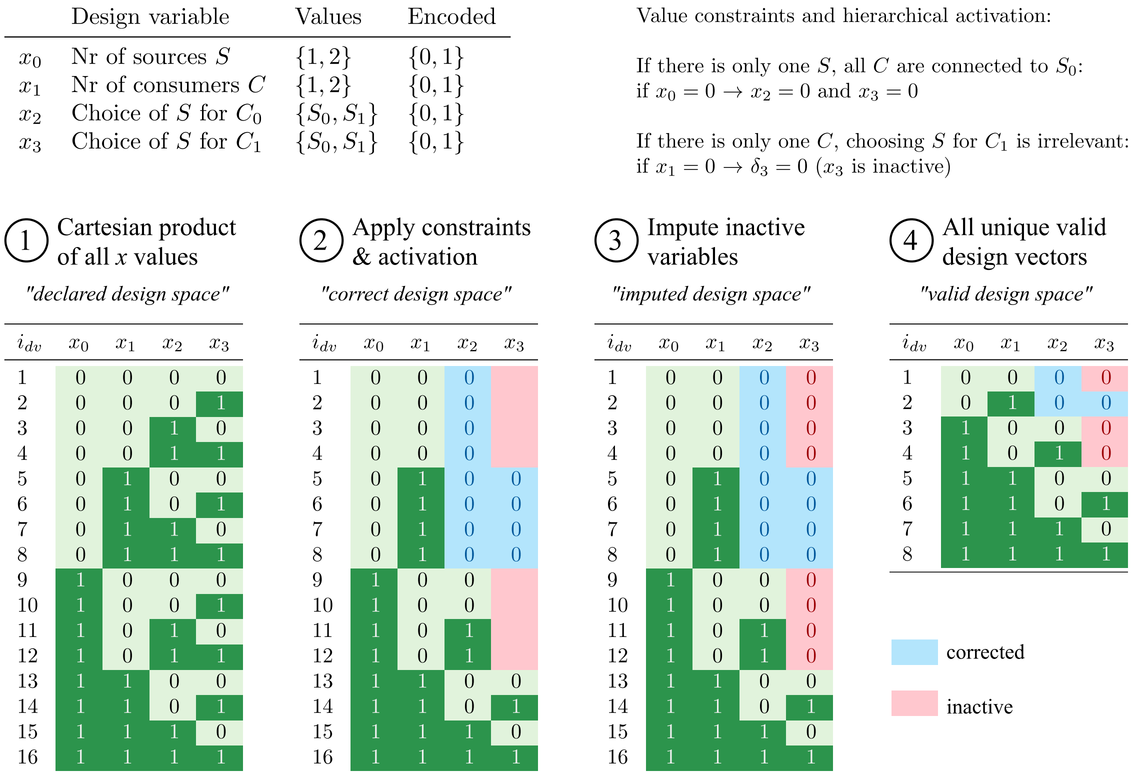
$$\text{MRD} = \max_{j \in 1,..,n_{x_d}} \text{RD}_j \tag{3}$$

with $j$ the discrete design variable index, $\delta_j = 0$ denoting the case where design variable $j$ is inactive, $N_j$ the number of possible options for discrete design variable $j$, and $\text{Rate}(j, \text{value})$ the relative occurrence rate of that value in all valid discrete design vectors. To ensure sufficient representation of design vectors subject to high rate diversities, a hierarchical sampling method is applied that works as follows:

1) Generate all possible valid discrete design vectors $x_{\text{valid,discr}}$ and obtain associated activeness information $\delta$;
2) Group design vectors by active variables $x_{\text{act}}$;
3) Uniformly sample discrete design vectors from each group;
4) Assign values to active continuous variables using Sobol' [42] sampling.

Investigations showed that this sampling method performs well for non-hierarchical problems, and performs better than random sampling for hierarchical problems. Refer to [12] for more details.

**Optimizing in Hierarchical Design Spaces** Finally, it was demonstrated that more integration of hierarchical information into the optimization problem yields better results. This mainly translates to the usage of the hierarchical sampling method described above, for which $x_{\text{valid,discr}}$ and activeness information $\delta$ is needed, and the application of correction and imputation as a repair operator. Correction and imputation ensure that all evaluated design vectors are valid, meaning that they satisfy value constraints and that all inactive design variables have canonical values. Value constraints occur when selecting a value for one design variable restricts the available values for another. For example, in the Apollo mission design problem [43], selecting a total crew size of 2 limits the number of crew members in the lunar module to 1 or 2, whereas a total crew size of 3 means there can be 1, 2, or 3 members assigned to the lunar module. Canonical values as applied to inactive design variables by imputation are 0 for discrete variables and center-domain for continuous variables. The principles of correction and imputation are visualized in Figure 2, showing how the declared design space (consisting of the Cartesian product of all discrete design variable values) is transformed to the valid design space (consisting of unique corrected and imputed design vectors).



| | Design variable | Values | Encoded |
|---|---|---|---|
| $x_0$ | Nr of sources $S$ | $\{1,2\}$ | $\{0,1\}$ |
| $x_1$ | Nr of consumers $C$ | $\{1,2\}$ | $\{0,1\}$ |
| $x_2$ | Choice of $S$ for $C_0$ | $\{S_0,S_1\}$ | $\{0,1\}$ |
| $x_3$ | Choice of $S$ for $C_1$ | $\{S_0,S_1\}$ | $\{0,1\}$ |

Value constraints and hierarchical activation:

If there is only one $S$, all $C$ are connected to $S_0$:
if $x_0 = 0 \rightarrow x_2 = 0$ and $x_3 = 0$

If there is only one $C$, choosing $S$ for $C_1$ is irrelevant:
if $x_1 = 0 \rightarrow \delta_3 = 0$ ($x_3$ is inactive)

1 Cartesian product of all $x$ values

*"declared design space"*

| $i_{dv}$ | $x_0$ | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 0 |
| 8 | 0 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 |
| 12 | 1 | 0 | 1 | 1 |
| 13 | 1 | 1 | 0 | 0 |
| 14 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 0 |
| 16 | 1 | 1 | 1 | 1 |

2 Apply constraints & activation

*"correct design space"*

| $i_{dv}$ | $x_0$ | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 🟥 |
| 2 | 0 | 0 | 0 | 🟥 |
| 3 | 0 | 0 | 0 | 🟥 |
| 4 | 0 | 0 | 0 | 🟥 |
| 5 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 1 | 0 | 0 |
| 8 | 0 | 1 | 0 | 0 |
| 9 | 1 | 0 | 0 | 🟥 |
| 10 | 1 | 0 | 0 | 🟥 |
| 11 | 1 | 0 | 1 | 🟥 |
| 12 | 1 | 0 | 1 | 🟥 |
| 13 | 1 | 1 | 0 | 0 |
| 14 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 0 |
| 16 | 1 | 1 | 1 | 1 |

3 Impute inactive variables

*"imputed design space"*

| $i_{dv}$ | $x_0$ | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 1 | 0 | 0 |
| 8 | 0 | 1 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 |
| 11 | 1 | 0 | 1 | 0 |
| 12 | 1 | 0 | 1 | 0 |
| 13 | 1 | 1 | 0 | 0 |
| 14 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 0 |
| 16 | 1 | 1 | 1 | 1 |

4 All unique valid design vectors

*"valid design space"*

| $i_{dv}$ | $x_0$ | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 |
| 4 | 1 | 0 | 1 | 0 |
| 5 | 1 | 1 | 0 | 0 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 0 |
| 8 | 1 | 1 | 1 | 1 |

🟦 corrected

🟥 inactive

**Fig. 2 Illustration of correction and imputation in hierarchical design spaces, showing how the different sets of design vectors relate to each other, reproduced from [12].**

Correction and imputation are made available as a standalone repair operator so that it can be used during sampling. This is important, because there might be a large difference between the declared design space as defined by the Cartesian product of all design variable values, and the valid design space containing only valid design vectors. This difference is measured by the *imputation ratio* IR, defined as:

$$\mathrm{IR}_d = \frac{\prod_{j=1}^{n_{x_d}} N_j}{n_{\text{valid,discr}}} \tag{4}$$

$$\mathrm{IR}_c = \frac{n_{\text{valid,discr}} \cdot n_{x_c}}{\sum_{l=1}^{n_{\text{valid,discr}}} \sum_{i=1}^{n_{x_c}} \delta_i(\boldsymbol{x}_{d,l})} \tag{5}$$

$$\mathrm{IR} = \mathrm{IR}_d \cdot \mathrm{IR}_c \tag{6}$$

with $n_{x_d}$ and $n_{x_c}$ the number of discrete and continuous design variables, respectively, $n_{\text{valid,discr}}$ the number of valid discrete design vectors, $\delta_i$ the activeness function for design variable $i$, and $\mathrm{IR}_d$ and $\mathrm{IR}_c$ the discrete and continuous imputation ratio, respectively. An imputation ratio of 1 indicates no hierarchy. Higher values represent the ratio between the declared and valid design spaces sizes and thereby design space hierarchy. Imputation ratio can also be interpreted as the amount of random vectors that need to be generated in order to find one valid design vector. For example for the suborbital vehicle design problem in [44] there are $123e3$ possible architectures, however the Cartesian product of design variables yields $2.8e6$ design vectors: therefore, the imputation ratio IR = $2.8e6/123e3 = 22.8$. This means that on average for every 22.8 randomly generated design vectors, there will be only one that represents a valid design. Without correction and imputation it will be difficult for an optimization algorithm to generate new valid design vectors due to this effect. This applies to BO as well as to evolutionary optimization algorithms.

To conclude, the BO algorithm described in this section and implemented as ArchSBO in SBArchOpt features [12]:

- Hierarchical mixed-discrete Gaussian Process models, optionally using Kriging with Partial Least Squares (KPLS) to reduce training times for high-dimensional problems;
- Ensemble infill criteria with a sequential-optimization procedure for batch infill generation for single- and multi-objective optimization problems;
- Hierarchical sampling algorithm that groups valid discrete design vectors by active design variables $x_{\text{act}}$ to deal with rate diversity effects;
- Availability of correction and imputation as a repair operator to deal with imputation ratio effects.

## III. Hidden Constraint Strategies in Bayesian Optimization

This section identifies and investigates various strategies for dealing with hidden constraints in Bayesian Optimization (BO) algorithms. Although we only discuss the integration with BO, the following discussions apply to any Surrogate-Based Optimization (SBO) algorithm in principle.

### A. Literature Review and Strategy Classification

Hidden constraints are encountered in many engineering problems, Müller & Day [24] provide several examples. They also provide an overview of strategies tried in the past, including for non-BO algorithms. For example, for algorithms that depend on some ranking-based selection procedure rather than surrogate model building, such as evolutionary algorithms, it suffices to apply the extreme barrier approach: replace NaN by $+\infty$ to prevent these points being selected as parents for creating offspring for the next generation of design points. Local optimization algorithms can deal with hidden constraints by implementing some way to retrace part of their search path. For SAO, however, we only consider methods relevant for global optimization algorithms. For surrogate-based algorithms, one of the simplest methods is to train the surrogate models only on viable points, thereby in effect *rejecting* failed points from the training set. The disadvantage to this approach is that knowledge of the design space is ignored: the optimizer might get stuck suggesting the same infill point(s) over and over, because it cannot know that these infill points will fail to evaluate. A more advanced approach is to *replace* the failed points by some values derived from viable points, as initially suggested by Forrester et al. [27] and inspired by imputation in the sense of replacing missing data in statistical datasets. They reason that failed points actually represent missing data and can be replaced by values of close-by viable points. However, the replaced values should drive the optimizer towards the viable region of the design space, which leads them to formulate a method for finding replacement values from a Gaussian Process (GP) model trained on the viable points only:

$$y_{\text{replace}}\left(\mathbf{x}_{\text{failed}}\right) = \hat{y}\left(\mathbf{x}_{\text{failed}}\right) + \alpha \cdot \hat{s}\left(\mathbf{x}_{\text{failed}}\right) \tag{7}$$

where $y$ represents the output value to be replaced, $\hat{y}$ and $\hat{s}$ the output and uncertainty estimates of the GP trained on viable design points, and $\alpha$ some multiplier ($\alpha = 1$ in [27]). They show that their approach works well for a continuous single-objective airfoil optimization problem. Another strategy for replacing values of failed points is by simply selecting one or more neighbor points and applying some aggregation function to a scalar value to replace. For example, Huyer & Neumaier [45] replace failed values by the max of $n_{\text{nb}}$ neighbor points. This concept can be expanded by considering $n_{\text{nb}} = 1$ to use the closest point to select the replacement value, or $n_{\text{nb}} = n_{\text{viable}}$ to consider all viable points for the replacement value.

The most advanced method for dealing with hidden constraints in SBO algorithms is to *predict* where the failed region lies with the help of another surrogate model [46]. Here the idea is to assign binary labels to all design point based on their failure status, for example 0 meaning failed and 1 meaning viable, train a model on this data, and then use that model to predict the so-called Probability of Viability (PoV). PoV lies between 0% and 100% and represents the probability that a newly-selected infill point will be viable (i.e. will not fail). During infill optimization, PoV can be used in two ways: either as a penalty multiplier to the infill criterion [46], or as a constraint to the infill algorithm that ensures that PoV of selected infill points is at or above some threshold [47]. Different types of surrogate models have been used to predict PoV, mainly selected due to their ability for modeling such binary classification problems. Used models include Random Forest Classifiers (RFC) [46], piecewise linear Radial Basis Functions (RBF) [24], Support Vector Machines (SVM) [48], SVM's with RBF kernel [47], Gaussian Process models [49, 50], and K-Nearest Neighbors (KNN) classifiers [51, 52].

**Table 1 Comparison of strategies for dealing with hidden constraints in Bayesian Optimization.**

| Strategy | Sub-strategy | Configuration | Parameter(s) |
|---|---|---|---|
| Rejection | | | |
| Replacement | Neighborhood [45] | Closest | |
| | | $n$-nearest, mean | $n$ |
| | | $n$-nearest, max | $n$ |
| | | Global, max | |
| | Predicted worst [27] | | $\alpha$ |
| Prediction | Random Forest Classifier (RFC) [46] | as $f_{\text{infill}}$ penalty | $\text{PoV}_{\text{min}}$ |
| | Gaussian Process (GP) [49] | as $g_{\text{infill}}$ | |
| | Support Vector Machine (SVM) [48] | | |
| | Piecewise linear RBF [24] | | |
| | K-Nearest Neighbor (KNN) classifier [51] | | |

To summarize the preceding discussion, there are three high-level strategies for dealing with hidden constraints in BO, also summarized in Table 1:

- Rejection: ignore failed points when training surrogate models;
- Replacement: replace values of failed points based on values of viable points, either from neighboring points or by predicted replacement;
- Prediction: train an additional surrogate model to predict the Probability of Validity (PoV), and using this either as infill constraint or as a penalty to the infill objectives when searching for infill points.

## B. Implementation into Bayesian Optimization Algorithms

The rejection and replacement strategies are implemented into the BO algorithm in a preprocessing step before training the GP models. The training set is separated into a viable and a failed set; for rejection the viable set is then simply discarded, whereas for replacement the points in the failed set are assigned some value for each output (i.e. each objective and constraint value) that is based on the viable points. Afterwards, the GP models are trained and the infill selection process continues as usual. For the prediction strategy, the viable set is used to train the GP models for infill search. The additional surrogate model for Probability of Viability (PoV) prediction is trained with a set containing all points and with binary labels assigned according to the viability status of the points: 0 for failed points and 1 for viable points. The infill optimization problem can then be modified by adding an inequality constraint that ensures that $\text{PoV}(\mathbf{x}) \geq \text{PoV}_{\text{min}}$:

$$g_{\text{PoV}}(\mathbf{x}) = \text{PoV}_{\text{min}} - \text{PoV}(\mathbf{x}) \leq 0 \tag{8}$$

where $\text{PoV}_{\text{min}}$ represents a user-defined minimum PoV to be reached, and $\text{PoV}(\mathbf{x})$ represents the predicted PoV for a given design point. PoV can also be integrated by modifying the infill objectives:

$$f_{\text{m,infill,mod}}(\mathbf{x}) = 1 - \left(\left(1 - f_{\text{m,infill}}(\mathbf{x})\right) \cdot \text{PoV}(\mathbf{x})\right) \tag{9}$$

where $f_{\text{m,infill}}$ represents the $m^{\text{th}}$ infill objective, and assuming infill objectives are normalized and to be minimized. Figure 3 presents several steps of running BO on a test problem, with an RBF model as PoV predictor used as an infill constraint at $\text{PoV}_{\text{min}} = 50\%$. The test problem is the single-objective problem from Alimo et al. [47], modified to have its optimum at the edge to the failed region near the top left corner of the 2D design space, implemented as AlimoEdge in SBArchOpt. As can be seen, all infill points suggested satisfy the $g_{\text{PoV}}$ constraint, however, in the earlier iterations this constraint can be inaccurate. Several infills are generated that violate the hidden constraint, and after each iteration the model gets more accurate at the edges to the failed regions for three locations in the design space where the optimizer expects the optimum to lie. This visualization additionally demonstrates that the predictor model should be able to handle one or more regions in the design space with closely-spaced failed and viable points.

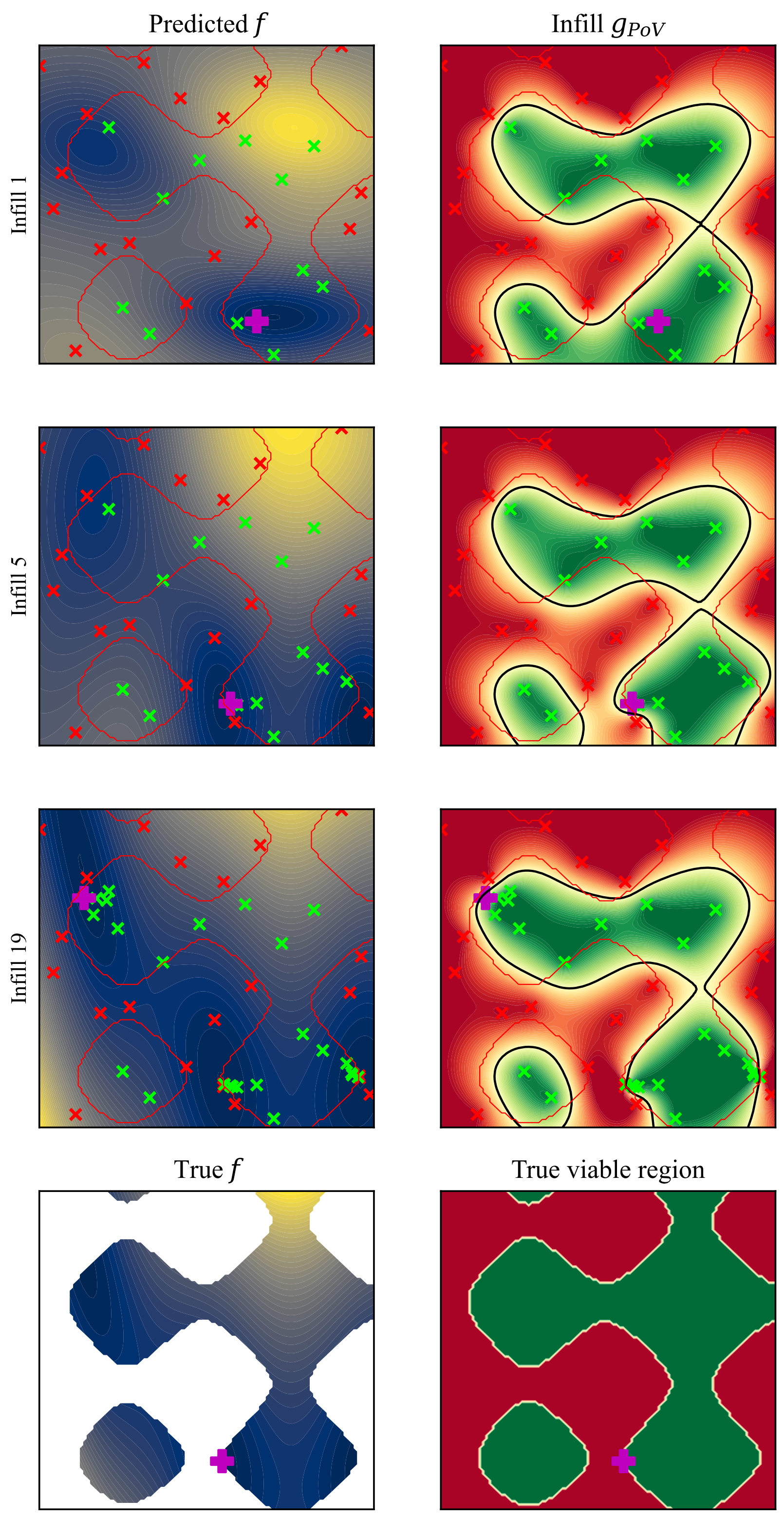


**Fig. 3 Several optimization steps between iteration 1 and 20 of BO executed on a test problem with its optimum lying at the edge to the failed region, as shown in the bottom row by the magenta cross. The main GP is shown on the left (darker means a lower, more optimal value), and the RBF model for predicting PoV is shown on the right. The RBF model is used as an infill constraint with $\mathrm{PoV_{min}} = 50\%$, showing green and red contours for satisfied and violated constraint values, respectively. Green, red, and magenta points represent viable, failed, and selected infill points, respectively.**

**Table 2 Test problems for comparing hidden constraint strategies. All problems are available in the SBArchOpt library [28]. Abbreviations and symbols: IR = imputation ratio, HC = hidden constraints, MD = mixed-discrete, H = hierarchical, MO = multi-objective, FR = fail rate, $n_{x_c}$ and $n_{x_d}$ = number of continuous and discrete design variables, respectively, $n_f$ = number of objectives, $n_g$ = number of constraints.**

| Name | SBArchOpt Class | Ref. | $n_{x_c}$ | $n_{x_d}$ | $n_f$ | $n_g$ | IR | FR |
|---|---|---|---|---|---|---|---|---|
| Branin | Branin | [53] | 2 | | 1 | | | 0% |
| HC Branin | HCBranin | [49] | 2 | | 1 | | | 33% |
| Alimo | Alimo | [47] | 2 | | 1 | | | 51% |
| Alimo Edge | AlimoEdge | | 2 | | 1 | | | 53% |
| HC Sphere | HCSphere | [48] | 2 | | 1 | | | 51% |
| Müller 1 | Mueller01 | [24] | 5 | | 1 | | | 67% |
| Müller 2 | Mueller02 | [24] | 4 | | 1 | | | 40% |
| HC CantBeam | CantileveredBeamHC | | 4 | | 1 | 1 | | 83% |
| HC Carside Less | CarsideHCLess | | 7 | | 1 | 9 | | 39% |
| HC Carside | CarsideHC | | 7 | | 3 | 8 | | 66% |
| MD/HC CantBeam | MDCantileveredBeamHC | | 2 | 2 | 1 | 1 | | 81% |
| MD/HC Carside | MDCarsideHC | | 3 | 4 | 3 | 8 | | 66% |
| H Alimo | HierAlimo | | 2 | 5 | 1 | | 5.4 | 51% |
| H Alimo Edge | HierAlimoEdge | | 2 | 5 | 1 | | 5.4 | 53% |
| H Müller 2 | HierMueller02 | | 4 | 4 | 1 | | 5.4 | 37% |
| H/HC Rosenbrock | HierarchicalRosenbrockHC | [54] | 8 | 5 | 1 | 1 | 1.5 | 21% |
| MO/H/HC Rosenbrock | MOHierarchicalRosenbrockHC | [54] | 8 | 5 | 2 | 1 | 1.5 | 60% |
| Jet SM | SimpleTurbofanArchModel | [10] | 9 | 6 | 1 | 5 | 3.9 | 50% |

## C. Comparing Strategy Performances

To compare strategy performance, we use the test problems listed in Table 2 to run the set of hidden constraint strategies listed in Table 3. Compared to the previously identified predictor models, we additionally test a Variational GP and the mixed-discrete GP developed in [55]. A Variational GP does not assume a Gaussian distribution and therefore might be able to more accurately model discontinuous functions as seen in classification problems [56]. We use the implementation provided in Trieste[‡] [57]. Through a quick pre-study, we additionally discard Support Vector Machine (SVM) and piecewise linear RBF classifiers due to their bad performance in higher-dimensional and mixed-discrete test problems. To ensure there are enough viable points to train models on for the infill search, the DoE size of problems containing hidden constraints is increased:

$$n_{\text{doe}} = \frac{k_{\text{doe}} \cdot n_x}{1 - \text{FR}_{\text{exp}}} \tag{10}$$

where $k_{\text{doe}}$ is the DoE multiplier, $n_x$ the number of design variables, and $\text{FR}_{\text{exp}}$ the expected fail rate. We assume an expected fail rate of 60% and use $k_{\text{doe}} = 2$ for following tests. Each optimization is executed with $n_{\text{infill}} = 50$ and is repeated 16 times. Optimization performance is compared using $\Delta$HV regret, a measure that represents the distance to the known optimum or Pareto-front ($\Delta$HV) cumulatively over the course of an optimization (regret). Lower $\Delta$HV regret is better, as it shows that the optimum was approached more closely and/or reached sooner. Strategies are then ranked by the ranking procedure outlined in [12].

[‡]https://secondmind-labs.github.io/trieste/

**Table 3 Tested Bayesian Optimization hidden constraint strategies. Abbreviations: GP = Gaussian Process.**

| Strategy | Sub-strategy | Configuration | Model Implementation |
|---|---|---|---|
| Rejection | | | |
| Replacement | Neighborhood | Global, max | |
| | | Local | |
| | | 5-nearest, max | |
| | | 5-nearest, mean | |
| | Predicted worst | $\alpha$ = 1 | SMT |
| | | $\alpha$ = 2 | SMT |
| Prediction | Random Forest Classifier | $\text{PoV}_{\text{min}}$ = 50% | scikit-learn |
| | K-Nearest Neighbors | $\text{PoV}_{\text{min}}$ = 50% | scikit-learn |
| | Radial Basis Function | $\text{PoV}_{\text{min}}$ = 50% | Scipy |
| | GP Classifier | $\text{PoV}_{\text{min}}$ = 50% | scikit-learn |
| | Variational GP | $\text{PoV}_{\text{min}}$ = 50% | Trieste |
| | Mixed-Discrete GP | $\text{PoV}_{\text{min}}$ = 50% | SMT |

Tables 4 and 5 presents optimization performance results. It can be seen that prediction with a Random Forest Classifier (RFC) performs best (rank 1) or good (rank $\leq$ 2) more compared to other strategies, with mixed-discrete (MD) GP prediction and predicted worst replacement ($\alpha$ = 1) closely following. A reduction in ΔHV regret is usually combined with a reduction in failure rate: this makes sense, as a lower failure rate indicates a better capability of avoiding the failed region and therefore more resources spent on exploring the viable region. Also shown is the change in training and infill times. Replacement strategies approximately double the training and infill cycle time, due to the fact that the trained GP models (one for each $f$ and $g$) contain the complete set of design points as training values, whereas for rejection and prediction only the viable points are used. Prediction strategies train an extra model to predict the PoV, which leads to a moderate increase in training and infill time.

**Table 4 Performance of hidden constraint strategies relative to the rejection strategy, averaged over all test problems at the end of the optimization runs. Darker color is better for ΔHV regret and Fail rate; worse for time columns. Abbreviations: HV = hypervolume, RFC = Random Forest Classifier, KNN = K-Nearest Neighbors, RBF = Radial Basis Functions, GP = Gaussian Process.**

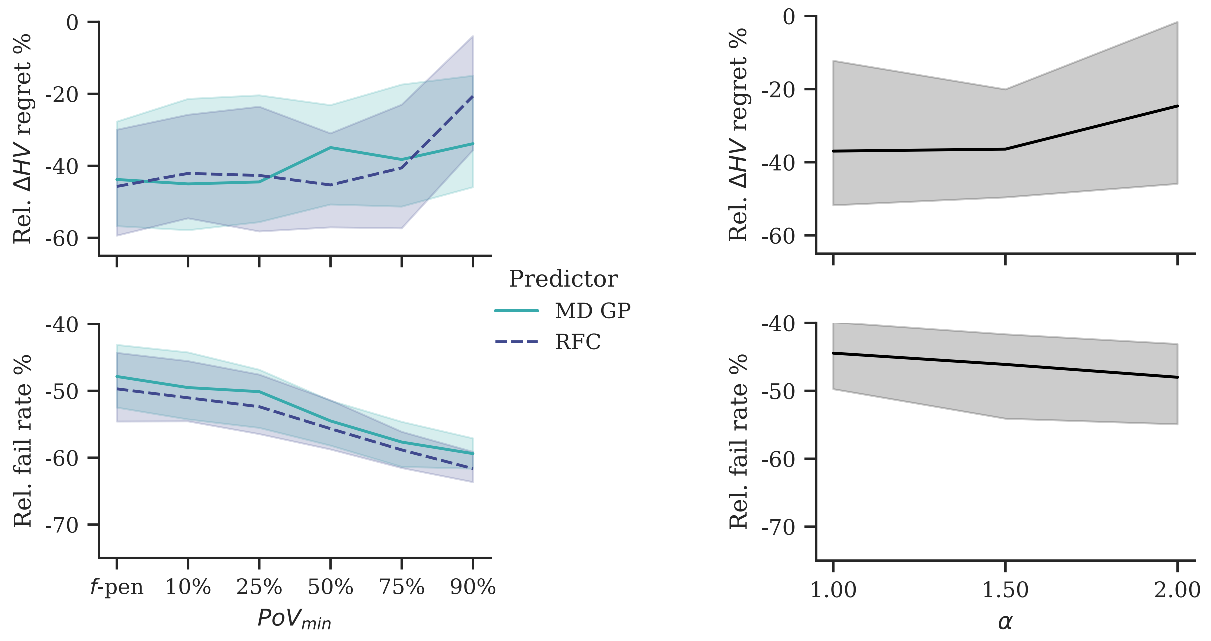
| Strategy | Sub-strategy | $\Delta HV$ regret | Fail rate | Training time | Infill time | Training + infill time |
|---|---|---|---|---|---|---|
| Rejection | | +0% | +0% | +0% | +0% | +0% |
| Replacement | Global max | -9% | -61% | +199% | +74% | +74% |
| Replacement | Local | -13% | -15% | +202% | +76% | +79% |
| Replacement | 5-nearest, max | -30% | -43% | +207% | +78% | +84% |
| Replacement | 5-nearest, mean | -35% | -23% | +193% | +78% | +77% |
| Replacement | Predicted worst | -33% | -48% | +192% | +67% | +73% |
| Replacement | Pred. worst ($\alpha$ = 2) | -30% | -53% | +199% | +75% | +78% |
| Prediction | RFC | -44% | -61% | +86% | +94% | +83% |
| Prediction | KNN | -27% | -39% | +48% | +59% | +49% |
| Prediction | RBF | -35% | -61% | +126% | +87% | +80% |
| Prediction | GP Classifier | -37% | -58% | +106% | +156% | +114% |
| Prediction | Variational GP | -32% | -60% | +68% | +243% | +189% |
| Prediction | MD GP | -38% | -62% | +116% | +95% | +91% |

**Table 5 Comparison of hidden constraint strategies on various test problems, ranked by ΔHV regret (lower rank / darker color is better). Best performing strategy is underlined. Abbreviations: HC = hidden constraints, MD = mixed-discrete, H = hierarchical, MO = multi-objective, RFC = Random Forest Classifier, KNN = K-Nearest Neighbors, RBF = Radial Basis Functions, GP = Gaussian Process.**

| | | Branin | HC Branin | Alimo | Alimo Edge | Müller 2 | HC Sphere | Müller 1 | HC CantB | HC Carside Less | HC Carside | MD/HC CantB | MD/HC Carside | H Alimo | H Alimo Edge | H Müller 2 | H/HC Rosenbr. | MO/H/HC Rbr. | Jet SM | Rank 1 | Rank $\leq$ 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rejection | | 1 | 4 | 2 | 4 | 1 | 2 | 2 | 5 | 6 | 6 | 5 | 3 | 4 | 4 | 2 | 3 | 3 | 4 | 11% | 33% |
| Replacement | Global max | 2 | 3 | 2 | 5 | 1 | 4 | 3 | 2 | 3 | 4 | 2 | 2 | 2 | 3 | 1 | 4 | 5 | 3 | 11% | 44% |
| Replacement | Local | 2 | 2 | 1 | 4 | 3 | 2 | 2 | 4 | 5 | 5 | 4 | 3 | 3 | 3 | 2 | 2 | 4 | 3 | 6% | 39% |
| Replacement | 5-nearest, max | 2 | 2 | 1 | 3 | 1 | 2 | 1 | 3 | 4 | 3 | 3 | 2 | 2 | 2 | 1 | 2 | 4 | 2 | 22% | 67% |
| Replacement | 5-nearest, mean | 1 | 2 | 1 | 3 | 2 | 1 | 1 | 4 | 2 | 2 | 3 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 39% | 83% |
| Replacement | Predicted worst | 2 | 1 | 1 | 4 | 1 | 3 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 1 | 1 | 3 | 1 | 56% | 83% |
| Replacement | Pred. worst ($\alpha$ = 2) | 2 | 2 | 2 | 3 | 2 | 4 | 2 | 1 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 39% | 72% |
| Prediction | RFC | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 3 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 78% | 94% |
| Prediction | KNN | 2 | 2 | 1 | 3 | 2 | 2 | 2 | 2 | 3 | 4 | 2 | 2 | 3 | 4 | 1 | 3 | 1 | 2 | 17% | 67% |
| Prediction | RBF | 2 | 1 | 1 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 2 | 1 | 2 | 3 | 1 | 1 | 1 | 1 | 50% | 83% |
| Prediction | GP Classifier | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 3 | 3 | 1 | 1 | 3 | 3 | 1 | 2 | 1 | 1 | 61% | 78% |
| Prediction | Variational GP | 1 | 1 | 1 | 1 | 1 | 3 | 4 | 1 | 4 | 2 | 1 | 1 | 2 | 3 | 1 | 2 | 2 | 1 | 56% | 78% |
| Prediction | MD GP | 2 | 1 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 | 1 | 72% | 94% |

Predicted worst replacement, prediction with RFC, and prediction with MD GP are selected as most promising candidates. These three strategies are further investigated to find out the influence of their parameter settings: $\alpha$ for predicted worst replacement, and $\text{PoV}_{\text{min}}$ for the prediction strategies. In addition, the prediction strategies are tested with integration as $f$-infill penalty (Eq. (9)). Tests are run with the same settings as the previous experiment.

Figure 4 shows the relative improvement over the rejection strategy and Table 6 presents ranking of algorithm performance for the tested strategy configurations. Fail rate is reduced significantly for higher values of $\text{PoV}_{\text{min}}$ and $\alpha$, which is expected as higher values result in a more conservative approach and therefore less exploration of the failed region. A reduction in failure rate, however, decreases optimizer performance (seen by an increase in $\Delta$HV regret), showing that a certain amount of failed evaluations is needed to sufficiently explore the design space. The best performing strategies are the prediction strategies at low $\text{PoV}_{\text{min}}$ values. It also shows that $f$-infill penalty behaves similar as low $\text{PoV}_{\text{min}}$ values. Prediction with MD GP or RFC are behaving similarly-well, although MD GP for a little wider range of $\text{PoV}_{\text{min}}$ than RFC. From these results, we recommend that either MD GP or RFC prediction should be used to deal with hidden constraints in Bayesian Optimization. PoV should be integrated as a constraint, because it allows more control over exploration vs exploitation compared to integration as $f$-infill penalty. $\text{PoV}_{\text{min}}$ should be kept relatively low to promote sufficient exploration: a value of $\text{PoV}_{\text{min}} = 25\%$ will be used subsequently.



**(a) Predictor strategies** **(b) Predicted worst replacement strategy**

**Fig. 4 Comparison of hidden constraint strategy settings relative to the rejection strategy, averaged over all test problems at the end of the optimization runs. Abbreviations: HV = hypervolume, MD = mixed-discrete, GP = Gaussian Process, RFC = Random Forest Classifier.**

**Table 6 Comparison of hidden constraint strategy settings on various test problems, ranked by ΔHV regret (lower rank / darker color is better). Best performing strategy is underlined. Abbreviations: HC = hidden constraints, MD = mixed-discrete, H = hierarchical, MO = multi-objective, RFC = Random Forest Classifier, KNN = K-Nearest Neighbors, RBF = Radial Basis Functions, GP = Gaussian Process.**

| | | Alimo | Alimo Edge | Müller 1 | MD/HC Carside | H Alimo | H/HC Rbr. | MO/H/HC Rbr. | Jet SM | Rank 1 | Rank ≤ 2 | Penalty |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Predicted Worst | $\alpha$ = 1.00 | 1 | 3 | 2 | 3 | 1 | 2 | 2 | 2 | 25% | 75% | 15% |
| Predicted Worst | $\alpha$ = 1.50 | 1 | 3 | 2 | 2 | 1 | 1 | 2 | 2 | 38% | 88% | 13% |
| Predicted Worst | $\alpha$ = 2.00 | 2 | 4 | 1 | 3 | 2 | 2 | 3 | 2 | 12% | 62% | 36% |
| <u>MD-GP</u> | <u>$f_{infill}$ penalty</u> | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | <u>75%</u> | <u>100%</u> | <u>1%</u> |
| MD-GP | $PoV_{min}$ = 10% | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 62% | 100% | 3% |
| <u>MD-GP</u> | <u>$PoV_{min}$ = 25%</u> | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | <u>75%</u> | <u>100%</u> | <u>-1%</u> |
| MD-GP | $PoV_{min}$ = 50% | 1 | 2 | 1 | 1 | 1 | 2 | 4 | 1 | 62% | 88% | 9% |
| MD-GP | $PoV_{min}$ = 75% | 1 | 2 | 2 | 1 | 1 | 2 | 3 | 1 | 50% | 88% | 7% |
| MD-GP | $PoV_{min}$ = 90% | 1 | 2 | 2 | 1 | 1 | 2 | 4 | 1 | 50% | 88% | 13% |
| RFC | $f_{infill}$ penalty | 1 | 1 | 2 | 3 | 2 | 1 | 1 | 1 | 62% | 88% | -2% |
| RFC | $PoV_{min}$ = 10% | 1 | 1 | 3 | 2 | 2 | 2 | 1 | 2 | 38% | 88% | 10% |
| RFC | $PoV_{min}$ = 25% | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 50% | 100% | 6% |
| RFC | $PoV_{min}$ = 50% | 1 | 2 | 2 | 3 | 2 | 1 | 1 | 1 | 50% | 88% | 1% |
| RFC | $PoV_{min}$ = 75% | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 38% | 100% | 9% |
| RFC | $PoV_{min}$ = 90% | 1 | 3 | 4 | 4 | 2 | 2 | 4 | 3 | 12% | 38% | 54% |

## IV. Application: Jet Engine Architecture Optimization

To demonstrate the hidden constraint optimization strategies presented in this work, the open-source jet engine optimization problem presented in [10] is used. The problem contains several architectural choices, such as whether to include a fan and bypass flow, the number of stages, the addition of a gearbox, and where to apply bleed and power offtakes. Each generated engine architecture is evaluated using a pyCycle [58] OpenMDAO [59] problem. The Multidisciplinary Design Optimization (MDO) problem is automatically constructed from a class-based engine architecture definition. Next to thermodynamic cycle analysis disciplines, additional disciplines include weight estimation, length and diameter calculation, and noise and NOx estimations.

We solve the simple version of the problem, which is a single-objective minimization of Thrust-Specific Fuel Consumption (TSFC) with following architectural choices: fan inclusion, number of compressor stages, gearbox inclusion, mixed nozzle selection, and power offtake locations. There are 70 valid discrete design vectors. However, the Cartesian product of discrete variables leads to 216 combinations: the discrete imputation ratio therefore is $\mathrm{IR}_d = 216/70 = 3.1$ (see Eq. (4)). The continuous imputation ratio $\mathrm{IR}_c = 1.26$ (see Eq. (5)), meaning that there are on average $9/1.26 = 7.14$ continuous variables active (as seen over all valid discrete design vectors). The overall imputation ratio is $\mathrm{IR} = 3.89$ (see Eq. (6)). Additionally, the underlying thermodynamic cycle analysis and sizing code does not always converge, leading to a hidden constraint being violated in approximately 50% of design points generated in a random DoE. This version of the problem is implemented in SBArchOpt as SimpleTurbofanArch.

The BO algorithm is executed 24 times with $n_{\mathrm{doe}} = 113$, $n_{\mathrm{infill}} = 187$ (a budget of 300 evaluations), and $n_{\mathrm{batch}} = 4$. Both hidden constraint strategies use $\mathrm{PoV}_{\mathrm{min}} = 25\%$. Figure 5 presents the results of the jet engine optimization. It shows that the RFC and MD GP predictors perform similarly, with the MD GP predictor slightly outperforming, and both being able to find the known optimum within the 300 evaluations. The known optimum was found with NSGA-II and an evaluation budget of 3250 [12]; BO therefore can be considered to be able to find the same result in 92% less function evaluations.

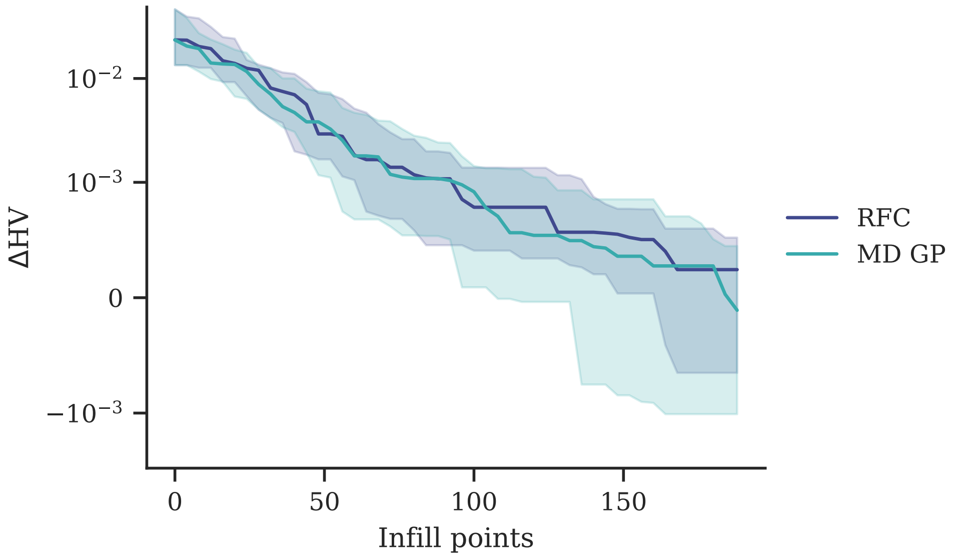


**Fig. 5 Optimization progression of the BO algorithm with the prediction hidden constraint strategy with two different predictors for the jet engine optimization problem. ΔHV represents the distance to the known Pareto front. The bands around the lines represent the 50 percentile range around the median, obtained from 24 repeated runs. Abbreviations: MD = mixed-discrete, GP = Gaussian Process, RFC = Random Forest Classifier.**

## V. Conclusions and Outlook

System Architecture Optimization (SAO) problems are challenging to solve due to their mixed-discrete and hierarchical design spaces combined with constrained, multi-objective, black-box, and expensive-to-evaluate solution spaces that potentially are subject to hidden constraints. Bayesian Optimization (BO) algorithms are gradient-free, global optimization algorithms that are specifically developed to optimize with a small evaluation budget. A BO algorithm is presented that has been developed for optimization in hierarchical design spaces, featuring:

- Hierarchical mixed-discrete Gaussian Process models, optionally using Kriging with Partial Least Squares (KPLS) to reduce training times for high-dimensional problems;
- Ensemble infill criteria with a sequential-optimization procedure for batch infill generation for single- and multi-objective optimization problems;
- Hierarchical sampling algorithm that groups valid discrete design vectors by active design variables $x_{\text{act}}$ to deal with rate diversity effects;
- Availability of correction and imputation as a repair operator to deal with imputation ratio effects.

The presented algorithm is extended to deal with hidden constraints. First, three high-level strategies are identified: rejection (ignore failed points), replacement (replace output values of failed points by some value derived from viable points), and prediction (predict the Probability of Viability). Replacement strategies are subdivided into replacement from neighboring points and replacement by prediction. Prediction strategies are subdivided based on the used classification models. It is demonstrated how prediction strategies can be implemented in BO, either as a penalty to the infill objectives or as additional inequality constraints to the infill problem parameterized by a lower threshold $\text{PoV}_{\text{min}}$.

Then, multiple hidden constraint strategies are tested on a database of test problems. It is shown that predicted worst replacement, prediction with a Random Forest Classifier (RFC), and prediction with a mixed-discrete Gaussian Process (GP) perform best. Compared to rejection, training and infill times are increased due to the increase in training points for replacement strategies and training of the PoV prediction model for prediction strategies. These three strategies are further tested with different values of $\text{PoV}_{\text{min}}$ (for prediction) and $\alpha$ for predicted worst replacement. It is shown that prediction with MD GP or RFC performs best, and with lower $\text{PoV}_{\text{min}}$ values. A default value of $\text{PoV}_{\text{min}} = 25\%$ is recommended. Finally, a jet engine architecture optimization problem is solved with the two predictor strategies, showing that prediction with MD GP performs best although it only slightly outperforms the RFC predictor. The extended BO algorithm and all used test problems are available in the open-source SBArchOpt library.

This work has shown that hidden constraints can be effectively dealt with by BO algorithms in the context of SAO. To validate its applicability, BO should be applied to more aerospace and non-aerospace architecture optimization problems. In future work the use of BO for SAO should be expanded, specifically to larger design space sizes with tens to hundreds of design variables, for example using dimension reduction techniques such as EGORSE [30], and multi-fidelity BO should be considered as a way to make BO more effective and further reduce required computational resources.

## Acknowledgments

The research presented in this paper has been performed in the framework of the COLOSSUS project (Collaborative System of Systems Exploration of Aviation Products, Services and Business Models) and has received funding from the European Union Horizon Europe Programme under grant agreement n° 101097120. The authors would like to recognize Luca Boggero and Björn Nagel for supporting the secondment of Jasper Bussemaker at ONERA DTIS in Toulouse, during which the research presented in this paper was mostly produced.

## References

[1] Judt, D., and Lawson, C., "Development of an automated aircraft subsystem architecture generation and analysis tool," *Engineering Computations*, Vol. 33, No. 5, 2016, pp. 1327–1352. https://doi.org/10.1108/EC-02-2014-0033.

[2] Iacobucci, J., "Rapid Architecture Alternative Modeling (Raam): a Framework for Capability-Based Analysis of System of Systems Architectures," Ph.D. thesis, Georgia Institute of Technology, 2012.

[3] McDermott, T., Folds, D., and Hallo, L., "Addressing Cognitive Bias in Systems Engineering Teams," *30th Annual INCOSE International Symposium*, Virtual Event, 2020. https://doi.org/10.1002/j.2334-5837.2020.00721.x.

[4] Bussemaker, J. H., and Ciampa, P., "MBSE in Architecture Design Space Exploration," *Handbook of Model-Based Systems*

*Engineering*, edited by A. Madni, N. Augustine, and M. Sievers, Springer, Switzerland, 2022. https://doi.org/10.1007/978-3-030-27486-3_36-1.

[5] Bussemaker, J. H., Boggero, L., and Nagel, B., "System Architecture Design Space Exploration: Integration with Computational Environments and Efficient Optimization," *AIAA AVIATION 2024 FORUM*, Las Vegas, NV, USA, 2024.

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

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

[8] 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, Springer Berlin Heidelberg, Berlin, Heidelberg, 2018, pp. 399–410. https://doi.org/10.1007/978-3-319-99259-4_32.

[9] Garg, S., García Sánchez, R., Bussemaker, J. H., Boggero, L., and Nagel, B., "Dynamic Formulation and Excecution of MDAO Workflows for Architecture Optimization," *AIAA AVIATION 2024 FORUM*, Las Vegas, NV, USA, 2024.

[10] Bussemaker, J. H., De Smedt, T., La Rocca, G., Ciampa, P. D., and Nagel, B., "System Architecture Optimization: An Open Source Multidisciplinary Aircraft Jet Engine Architecting Problem," *AIAA AVIATION 2021 FORUM*, Virtual Event, 2021. https://doi.org/10.2514/6.2021-3078.

[11] Gamot, J., Balesdent, M., Tremolet, A., Wuilbercq, R., Melab, N., and Talbi, E.-G., "Hidden-variables genetic algorithm for variable-size design space optimal layout problems with application to aerospace vehicles," *Engineering Applications of Artificial Intelligence*, Vol. 121, 2023, p. 105941.

[12] Bussemaker, J. H., Saves, P., Bartoli, N., Lefebvre, T., Lafage, R., and Nagel, B., "System Architecture Optimization Strategies: Dealing with Expensive Hierarchical Problems," *Journal of Global Optimization*, 2024. Article submitted.

[13] 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. https://doi.org/10.1016/j.neucom.2019.11.004.

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

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

[16] Zuniga, 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. https://doi.org/10.1080/03155986.2020.1730677.

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

[18] Audet, C., Hallé-Hannan, E., and Digabel, S. L., "A general mathematical framework for constrained mixed-variable blackbox optimization problems with meta and categorical variables," 2022. https://doi.org/10.48550/ARXIV.2204.00886.

[19] Saves, P., Lafage, R., Bartoli, N., Diouane, Y., Bussemaker, J. H., Lefebvre, T., Hwang, J. T., Morlier, J., and Martins, J. R., "SMT 2.0: A Surrogate Modeling Toolbox with a focus on hierarchical and mixed variables Gaussian processes," *Advances in Engineering Software*, Vol. 188, 2024, p. 103571. https://doi.org/10.1016/j.advengsoft.2023.103571.

[20] 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. https://doi.org/10.1109/TEVC.2005.851274.

[21] 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. https://doi.org/10.1016/j.cor.2019.104869.

[22] Bussemaker, J. H., Bartoli, N., Lefebvre, T., Ciampa, P. D., and Nagel, B., "Effectiveness of Surrogate-Based Optimization Algorithms for System Architecture Optimization," *AIAA AVIATION 2021 FORUM*, Virtual Event, 2021. https://doi.org/10.2514/6.2021-3095.

[23] Bussemaker, J. H., García Sánchez, R., Fouda, M., Boggero, L., and Nagel, B., "Function-Based Architecture Optimization: An Application to Hybrid-Electric Propulsion Systems," *33rd Annual INCOSE International Symposium*, Honolulu, HI, USA, 2023. https://doi.org/10.1002/iis2.13020.

[24] Müller, 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. https://doi.org/10.1287/ijoc.2018.0864.

[25] Le Digabel, S., and Wild, S. M., "A taxonomy of constraints in black-box simulation-based optimization," *Optimization and Engineering*, 2023. https://doi.org/10.1007/s11081-023-09839-3.

[26] Krengel, M., and Hepperle, M., "Effects of Wing Elasticity and Basic Load Alleviation on Conceptual Aircraft Designs," *AIAA SCITECH 2022 Forum*, 2022. https://doi.org/10.2514/6.2022-0126.

[27] Forrester, A. I., Sóbester, A., and Keane, A. J., "Optimization with missing data," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 462, No. 2067, 2006, pp. 935–945. https://doi.org/10.1098/rspa.2005.1608.

[28] Bussemaker, J. H., "SBArchOpt: Surrogate-Based Architecture Optimization," *Journal of Open Source Software*, Vol. 8, No. 89, 2023, p. 5564. https://doi.org/10.21105/joss.05564.

[29] Saves, P., Diouane, Y., Bartoli, N., Lefebvre, T., and Morlier, J., "A mixed-categorical correlation kernel for Gaussian process," *Neurocomputing*, Vol. 550, 2023, p. 126472. https://doi.org/10.1016/j.neucom.2023.126472.

[30] Priem, R., Bartoli, N., Diouane, Y., Dubreuil, S., and Saves, P., "High-dimensional efficient global optimization using both random and supervised embeddings," *AIAA AVIATION 2023 Forum*, American Institute of Aeronautics and Astronautics, San Diego, CA, USA, 2023. https://doi.org/10.2514/6.2023-4448.

[31] Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P., "Manifold Gaussian Processes for regression," *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Vancouver, CA, 2016. https://doi.org/10.1109/ijcnn.2016.7727626.

[32] Bouhlel, M., Bartoli, N., Otsmane, A., and Morlier, J., "Improving Kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction," *Structural and Multidisciplinary Optimization*, Vol. 53, No. 5, 2016, pp. 935–952. https://doi.org/10.1007/s00158-015-1395-9.

[33] Saves, P., Bartoli, N., Diouane, Y., Lefebvre, T., Morlier, J., David, C., Nguyen Van, E., and Defoort, S., "Bayesian optimization for mixed variables using an adaptive dimension reduction process: applications to aircraft design," *AIAA SCITECH 2022 Forum*, American Institute of Aeronautics and Astronautics, San Diego, CA, USA, 2022. https://doi.org/10.2514/6.2022-0082.

[34] Charayron, R., Lefebvre, T., Bartoli, N., and Morlier, J., "Multi-fidelity Bayesian optimization strategy applied to Overall Drone Design," *AIAA SCITECH 2023 Forum*, American Institute of Aeronautics and Astronautics, National Harbor, MD, USA, 2023. https://doi.org/10.2514/6.2023-2366.

[35] Cowen-Rivers, A. I., Lyu, W., Tutunov, R., Wang, Z., Grosnit, A., Griffiths, R. R., Maraval, A. M., Jianye, H., Wang, J., Peters, J., and Ammar, H. B., "HEBO: Pushing The Limits of Sample-Efficient Hyperparameter Optimisation," 2020. https://doi.org/10.48550/ARXIV.2012.03826.

[36] Lyu, W., Yang, F., Yan, C., Zhou, D., and Zeng, X., "Batch Bayesian Optimization via Multi-objective Acquisition Ensemble for Automated Analog Circuit Design," *Proceedings of the 35th International Conference on Machine Learning*, PMLR, Stockholm, SE, 2018.

[37] Garnett, R., *Bayesian Optimization*, Cambridge University Press, Cambridge, UK, 2023. https://doi.org/10.1017/9781108348973.

[38] Cox, D., and John, S., "A statistical method for global optimization," *1992 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, Chicago, IL, USA, 1992. https://doi.org/10.1109/icsmc.1992.271617.

[39] Hawe, G., and Sykulski, J., "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.

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

[41] Deb, K., "Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction," *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, Springer, London, UK, 2011, pp. 3–34. https://doi.org/10.1007/978-0-85729-652-8_1.

[42] Renardy, M., Joslyn, L. R., Millar, J. A., and Kirschner, D. E., "To Sobol or not to Sobol? The effects of sampling schemes in systems biology applications," *Mathematical Biosciences*, Vol. 337, 2021, p. 108593. https://doi.org/10.1016/j.mbs.2021.108593.

[43] Simmons, W., "A Framework for Decision Support in Systems Architecting," Ph.D. thesis, Massachusetts Institute of Technology, 2008.

[44] Frank, C., Marlier, R., Pinon-Fischer, O., and Mavris, D., "An Evolutionary Multi-Architecture Multi-Objective Optimization Algorithm for Design Space Exploration," *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Reston, Virginia, 2016, pp. 1–19. https://doi.org/10.2514/6.2016-0414.

[45] Huyer, W., and Neumaier, A., "SNOBFIT – Stable Noisy Optimization by Branch and Fit," *ACM Transactions on Mathematical Software*, Vol. 35, No. 2, 2008, pp. 1–25. https://doi.org/10.1145/1377612.1377613.

[46] 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.

[47] Alimo, S. R., Beyhaghi, P., and Bewley, T. R., "Delaunay-Based Global Optimization in Nonconvex Domains Defined by Hidden Constraints," *Computational Methods in Applied Sciences*, Springer International Publishing, 2018, pp. 261–271. https://doi.org/10.1007/978-3-319-89890-2_17.

[48] Sacher, M., Duvigneau, R., Maître, O. L., Durand, M., Berrini, É., Hauville, F., and Astolfi, J.-A., "A classification approach to efficient global optimization in presence of non-computable domains," *Structural and Multidisciplinary Optimization*, Vol. 58, No. 4, 2018, pp. 1537–1557. https://doi.org/10.1007/s00158-018-1981-8.

[49] Gelbart, M. A., Snoek, J., and Adams, R. P., "Bayesian Optimization with Unknown Constraints," *UAI'14: Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, Arlington, VA, USA, 2014.

[50] Bachoc, F., Helbert, C., and Picheny, V., "Gaussian process optimization with failures: classification and convergence proof," *Journal of Global Optimization*, Vol. 78, No. 3, 2020, pp. 483–506. https://doi.org/10.1007/s10898-020-00920-0.

[51] Audet, C., Caporossi, G., and Jacquet, S., "Binary, unrelaxable and hidden constraints in blackbox optimization," *Operations Research Letters*, Vol. 48, No. 4, 2020, pp. 467–471. https://doi.org/10.1016/j.orl.2020.05.011.

[52] Tfaily, A., Kokkolaras, M., Bartoli, N., and Diouane, Y., "Efficient Acquisition Functions for Bayesian Optimization in the Presence of Hidden Constraints," *AIAA Aviation 2023 Forum*, San Diego, USA, 2023. https://doi.org/10.2514/6.2023-4261.

[53] Forrester, A. I. J., Sóbester, A., and Keane, A. J., *Engineering Design via Surrogate Modelling*, 2008. https://doi.org/10.1002/9780470770801.

[54] Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E., 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, Springer International Publishing, Cham, 2020, pp. 189–224. https://doi.org/10.1007/978-3-030-18764-4_9.

[55] Saves, P., Diouane, Y., Bartoli, N., Lefebvre, T., and Morlier, J., "A general square exponential kernel to handle mixed-categorical variables for Gaussian process," *AIAA AVIATION 2022 Forum*, American Institute of Aeronautics and Astronautics, Chicago, IL, USA, 2022. https://doi.org/10.2514/6.2022-3870.

[56] Hensman, J., Matthews, A., and Ghahramani, Z., "Scalable Variational Gaussian Process Classification," *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 38, PMLR, San Diego, California, USA, 2015, pp. 351–360.

[57] Picheny, V., Berkeley, J., Moss, H. B., Stojic, H., Granta, U., Ober, S. W., Artemev, A., Ghani, K., Goodall, A., Paleyes, A., Vakili, S., Pascual-Diaz, S., Markou, S., Qing, J., Loka, N. R. B. S., and Couckuyt, I., "Trieste: Efficiently Exploring The Depths of Black-box Functions with TensorFlow," 2023. https://doi.org/10.48550/ARXIV.2302.08436.

[58] Hendricks, E., and Gray, J., "pyCycle: A Tool for Efficient Optimization of Gas Turbine Engine Cycles," *Aerospace*, Vol. 6, No. 8, 2019, p. 87. https://doi.org/10.3390/aerospace6080087.

[59] Gray, J., Hwang, J., Martins, J., Moore, K., and Naylor, B., "OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, Vol. 59, No. 4, 2019, pp. 1075–1104. https://doi.org/10.1007/s00158-019-02211-z.