# Black-box Mixed-Variable Optimisation Using a Surrogate Model that Satisfies Integer Constraints

Laurens Bliek
l.bliek@tue.nl
Eindhoven University of Technology
Eindhoven, Netherlands

Arthur Guijt
arthur.guijt@cwi.nl
Centrum Wiskunde & Informatica
Amsterdam, Netherlands

Sicco Verwer
S.E.Verwer@tudelft.nl
Delft University of Technology
Delft, Netherlands

Mathijs de Weerdt
M.M.deWeerdt@tudelft.nl
Delft University of Technology
Delft, Netherlands

## ABSTRACT

A challenging problem in both engineering and computer science is that of minimising a function for which we have no mathematical formulation available, that is expensive to evaluate, and that contains continuous and integer variables, for example in automatic algorithm configuration. Surrogate-based algorithms are very suitable for this type of problem, but most existing techniques are designed with only continuous or only discrete variables in mind. Mixed-Variable ReLU-based Surrogate Modelling (MVRSM) is a surrogate-based algorithm that uses a linear combination of rectified linear units, defined in such a way that (local) optima satisfy the integer constraints. Unlike other methods, it also has a constant run-time per iteration. This method outperforms the state of the art on several synthetic benchmarks with up to 238 continuous and integer variables, and achieves competitive performance on two real-life benchmarks: XG-Boost hyperparameter tuning and Electrostatic Precipitator optimisation.

## CCS CONCEPTS

• **Mathematics of computing** → **Mathematical optimization**; • **Computing methodologies** → *Active learning settings*; **Search methodologies**; • **Theory of computation** → *Mixed discrete-continuous optimization.*

## KEYWORDS

surrogate models, expensive optimisation, Bayesian optimisation, mixed-variable optimisation

**ACM Reference Format:**
Laurens Bliek, Arthur Guijt, Sicco Verwer, and Mathijs de Weerdt. 2021. Black-box Mixed-Variable Optimisation Using a Surrogate Model that Satisfies Integer Constraints. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449726.3463136


*GECCO '21 Companion, July 10–14, 2021, Lille, France*

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3463136

## 1 INTRODUCTION

Surrogate modelling techniques such as Bayesian optimisation have a long history of success in optimising expensive black-box objective functions [16, 20, 21]. These are functions that have no mathematical formulation available and take some time or other resource to evaluate, which occurs for example when they are the result of some simulation, algorithm or scientific experiment. Often there is also randomness or noise involved in these evaluations. By approximating the objective with a cheaper surrogate model, the optimisation problem can be solved more efficiently. This also opens up possibilities to apply evolutionary algorithms on problems with expensive objectives, leading to surrogate-assisted evolutionary algorithms [15].

While most attention in the surrogate modelling literature has gone to problems in continuous domains, recently solutions for combinatorial optimisation problems have started to arise [1, 2, 6, 11, 27]. Yet a significant number of problems [7] contain a mix of continuous and discrete variables, for example material design [14], optical filter optimisation [30], and automated machine learning [13]. The literature on surrogate modelling techniques for these types of problems is even more sparse than for purely discrete problems. Discretising the continuous variables to make use of a purely discrete surrogate model, or applying rounding techniques to make use of a purely continuous surrogate model are both seen as common but inadequate ways to solve the problem [11, 25]. The few existing techniques that can deal with a mixed variable setting still have considerable room for improvement in accuracy or efficiency. When the surrogate model is not expressive enough and does not model any interaction between the different variables, it might perform poorly [3], especially when many variables are involved. On the other hand, most Bayesian optimisation techniques do model the interaction between all variables, but use a surrogate model that grows in size every iteration. This causes those algorithms to become slower over time, potentially even becoming more expensive than the expensive objective itself.

Our main contribution is a surrogate modelling algorithm called Mixed-Variable ReLU-based Surrogate Modelling (MVRSM) that can deal with problems with continuous and integer variables efficiently and accurately. This is realised by using a continuous surrogate model that:

- models interactions between all variables,

- does not grow in size over time and can be updated efficiently, and
- has the guarantee that any local optimum is located in a point where the integer constraints are satisfied.

The first point ensures that the model remains accurate, even for large-scale problems. The second point ensures that the algorithm does not slow down over time, as the number of basis functions and free parameters of the surrogate model remains fixed. Finally, the last point eliminates the need for rounding techniques, and also eliminates the need for repeatedly using integer programming as is done in [9]. Though the model used by MVRSM does not guarantee that integer constraints are satisfied everywhere, it is constructed in such a way that these constraints are satisfied in the local optima of the model.

Besides the proposed algorithm, the contributions include a proof that the local optima of the proposed surrogate model are integer-valued in the intended variables. We also include empirical evidence of the effectiveness of this method on several large-scale synthetic benchmarks from related work and two real-life benchmarks: XGBoost hyperparameter tuning and Electrostatic Precipitator optimisation.

## 2 PRELIMINARIES

This work considers the problem of finding the minimum of a mixed-variable black-box objective function $f : \mathbb{R}^{\gamma_c} \times \mathbb{Z}^{\gamma_d} \to \mathbb{R}$ that can only be accessed via expensive and noisy measurements $y = f(\mathbf{x}_c, \mathbf{x}_d) + \epsilon$. That is, we want to solve

$$\min_{\mathbf{x}_c \in X_c, \mathbf{x}_d \in X_d} f(\mathbf{x}_c, \mathbf{x}_d), \tag{1}$$

where $\gamma_c$ is the number of continuous variables, $\gamma_d$ the number of integer variables, $\epsilon \in \mathbb{R}$ is a zero-mean random variable with finite variance, and $X_c \subseteq \mathbb{R}^{\gamma_c}$ and $X_d \subseteq \mathbb{Z}^{\gamma_d}$ are the bounded domains of the continuous and integer variables respectively. In this work, the lower and upper bounds of either $X_c$ or $X_d$ for the $i$-th variable are denoted $l_i$ and $u_i$ respectively. Since $X_d \subseteq \mathbb{Z}^{\gamma_d}$, we call $\mathbf{x}_d \in \mathbb{Z}^{\gamma_d}$ the integer constraints. Expensive in this context means that it takes some time or other resource to evaluate $y$, as is the case in for example hyperparameter tuning problems [3] and many engineering problems [5, 27]. Therefore, we wish to treat (1) using as few queries as possible.

The problem is usually solved with a surrogate modelling technique such as Bayesian optimisation [21]. In this approach, the data samples $(\mathbf{x}_c, \mathbf{x}_d, y)$ are used to approximate the objective $f$ with a surrogate model $g$. Usually, $g$ is a machine learning model such as a Gaussian process, random forest or a weighted sum of nonlinear basis functions. In any case, it has an exact mathematical formulation, which means that $g$ can be optimised with standard techniques as it is not expensive to evaluate and it is not black-box. If $g$ is indeed a good approximation of the original objective $f$, it can be used to suggest new candidate points of the search space $X_c \times X_d$ where $f$ should be evaluated. This happens iteratively, where in every iteration $f$ is evaluated, the approximation $g$ of $f$ is improved, and optimisation on $g$ is used to suggest a next point to evaluate $f$.

## 3 RELATED WORK

The main focus of this work is to evaluate a novel surrogate model that enforces the integer constraints in the mixed-variable setting described in the previous section. For this evaluation we utilise a basic surrogate-based optimisation framework similar to Bayesian optimisation. In Bayesian optimisation, Gaussian processes are the most popular surrogate model [21]. On the one hand, these surrogate models lend themselves well to problems with only continuous variables, but not so much when they include integer variables as well. On the other hand, there have been several recent approaches to develop surrogate models for problems with only discrete variables [1, 6, 11, 27].

The mixed-variable setting is not as well-developed, although there are some surrogate modelling methods that can deal with this. We start by mentioning two well-known methods, namely SMAC [12] and HyperOpt [3], followed by more recent work, along with their strengths and shortcomings. We end this section with recent work on discrete surrogate models that we make use of throughout this paper.

SMAC [12] uses random forests as the surrogate model, though the software also supports Gaussian processes. Random forests are good at capturing interactions between the variables, but the main disadvantage is that they are less accurate in unseen parts of the search space, at least compared to other surrogate models. HyperOpt [3] uses a Tree-structured Parzen Estimator as the surrogate model. This algorithm is known to be fast in practice, has been shown to work in settings with over 200 variables, and also has the ability to deal with conditional variables, where certain variables only exist if other variables take on certain values. Its main disadvantage is that complex interactions between variables are not modelled. Most other existing Bayesian optimisation algorithms have to resort to rounding or discretisation in order to deal with the mixed variable setting, which both have their disadvantages [11, 25].

More recently, the CoCaBO algorithm was proposed [25], which is developed for problems with a mix of continuous and categorical variables. It makes use of a mix of multi-armed bandits and Gaussian processes. Similar ideas are utilised in [22, 28]. Another interesting new research direction is to combine the advantages of Gaussian processes and artificial neural networks [17], although more research is required to make this computationally feasible for larger problems. Other research groups have focused their attention on mixed-variable problems with multiple objectives [14, 30].

Most of the methods mentioned here suffer from the drawback that the surrogate model grows while the algorithm is running, causing the algorithms to slow down over time. This problem has been addressed and solved for the continuous setting in the DONE algorithm [5] and for the discrete setting in the COMBO [27] and IDONE algorithms [6] by making use of parametric surrogate models that are linear in the parameters. The MiVaBO algorithm [9] is, to the best of our knowledge, the first algorithm that applies this solution to the mixed variable setting. It relies on an alternation between continuous and discrete optimisation to find the optimum of the surrogate model.

In contrast with MiVaBO, the IDONE algorithm has the theoretical guarantee that any local minimum of the surrogate model

satisfies the integer constraints, so only continuous optimisation needs to be used. This is achieved by using a surrogate model consisting of a linear combination of rectified linear units (ReLUs), a popular basis function in the machine learning community. Using only continuous optimisation is much more efficient than the approach used in MiVaBO. However, this theory only applies to problems without continuous variables.

## 4 MIXED-VARIABLE RELU-BASED SURROGATE MODELLING

In this section, we use the theory from the IDONE algorithm to develop a ReLU-based surrogate model for the mixed-variable setting. This is far from trivial, as a wrong choice of surrogate model might result in limited interaction between all variables, in not being able to optimise the surrogate model efficiently, or in not being able to satisfy the integer constraints.

Below we present the Mixed-Variable ReLU-based Surrogate Modelling (MVRSM) algorithm. This algorithm makes use of a surrogate model based on rectified linear units and includes interactions between all variables, is easy to update and to optimise, and has its local optima situated in points that satisfy the integer constraints.

### 4.1 Proposed surrogate model

As in related work [4, 6, 9], we use a continuous surrogate model $g : \mathbb{R}^{\gamma_c+\gamma_d} \to \mathbb{R}$:

$$g(\mathbf{x}_c, \mathbf{x}_d) = \sum_{k=1}^{D} c_k \phi_k(\mathbf{x}_c, \mathbf{x}_d), \tag{2}$$

with $D$ being the number of basis functions. The model is linear in its own parameters $c$, which allows it to be trained with linear regression. We choose the basis functions $\phi$ in such a way that all local optima $(\mathbf{x}_c^*, \mathbf{x}_d^*)$ of the model satisfy $\mathbf{x}_d \in \mathbb{Z}^{\gamma_d}$, as explained later in this section. This leads to an efficient way of finding the minimum of the surrogate model for mixed variables. We choose rectified linear units as the basis functions:

$$\phi_k(\mathbf{x}_c, \mathbf{x}_d) = \max\{0, z_k(\mathbf{x}_c, \mathbf{x}_d)\}, \tag{3}$$

$$z_k(\mathbf{x}_c, \mathbf{x}_d) = [\mathbf{v}_k^T \mathbf{w}_k^T] \begin{bmatrix} \mathbf{x}_c \\ \mathbf{x}_d \end{bmatrix} + b_k, \tag{4}$$

with $\mathbf{v}_k \in \mathbb{R}^{\gamma_c}$, $\mathbf{w}_k \in \mathbb{R}^{\gamma_d}$, and $b_k \in \mathbb{R}$. This causes the surrogate model $g$ to be piece-wise linear. There are four strategies for choosing the model parameters $\mathbf{v}_k, \mathbf{w}_k, b_k$:

- optimise them together with the weights $c_k$,
- choose them directly according to the data samples in a non-parametric way using kernel basis functions [21, 25],
- choose them randomly once and then fix them [4, 5, 9, 27], or
- choose them according to the variable domains $X_c, X_d$ and then fix them [6].

The first option is not recommended as nonlinear optimisation would have to be used, while linear regression techniques can be used for the parameters $c_k$. The second option has the downside that more and more basis functions need to be added as data samples are gathered, making the surrogate model grow in size while the algorithm is running. This is what happens in most Bayesian optimisation algorithms, which causes them to slow down over time. The third option fixes this problem, but even though there are good approximation theorems available for a random choice of the parameters [5, 23], it does not give any guarantees on satisfying the integer constraints. The fourth option does, but only for problems that have no continuous variables. Therefore, we propose to use a mix of the third and fourth option, getting the best of both options, as explained below.

We first state the required definitions, followed by our main theoretical contribution.

Definition 1 (Integer $z$-function). *An* integer $z$-function $z_k$ *is chosen according to* (4) *with* $\mathbf{v} = \mathbf{0}$ *and with* $\mathbf{w}$ *and* $b$ *having integer values chosen according to Algorithm* 2 *from [6]. That means it has one of the following forms:* $z_k(\mathbf{x}_c, \mathbf{x}_d) = z_k(\mathbf{x}_d) = \pm(x_i - \alpha)$, *with* $x_i$ *an element from* $\mathbf{x}_d$ *and* $\alpha \in \mathbb{Z}$ *chosen between* $l_i$ *and* $u_i$ *(the lower and upper bounds of* $x_i$*), or* $z_k(\mathbf{x}_c, \mathbf{x}_d) = z_k(\mathbf{x}_d) = \pm(x_i - x_{i-1} - \alpha)$, *for* $i > 1$ *and* $\alpha \in \mathbb{Z}$ *chosen between* $l_i - u_{i-1}$ *and* $u_i - l_{i-1}$. *This results in a basis function that depends only on one or two subsequent integer variables and does not depend on any continuous variables.*

By making use of the integer $z$-functions that are shaped according to the discrete part of the search space, we have a surrogate model with basis functions that depend on the integer variables only. If we would add basis functions that depend only on the continuous variables, the possible interaction between continuous and integer variables would not be modelled. But if we add randomly chosen mixed basis functions as in [9], then we might lose the guarantee that integer constraints are satisfied in local minima. See Figure 1 (left).

To avoid both problems, we propose to add mixed basis functions as in [9], but we choose them pseudo-randomly rather than randomly. This benefits from the success that randomly chosen weights have had in the past [4, 5, 9, 27], while avoiding the problem from Figure 1 (left).

Definition 2 (Mixed $z$-function). *A* mixed $z$-function $z_k$ *is chosen according to* (4) *with* $\omega_k = \begin{bmatrix} \mathbf{v}_k \\ \mathbf{w}_k \end{bmatrix}$ *sampled from a set* $\Omega$ *that contains* $\gamma_c$ *random vectors in* $\mathbb{R}^{\gamma_c+\gamma_d}$ *with a continuous probability distribution* $p_\omega$, *and* $b_k$ *is then chosen from a random continuous probability distribution* $p_b$ *which depends on* $\omega_k$. *This results in a basis function that depends on all continuous and on all integer variables.*

The probability distributions $p_\omega$ and $p_b$ are chosen in such a way that the mixed $z$-functions are never completely outside the domain $X_c \times X_d$. (The exact procedure for choosing them can be found in the technical appendix.) As a result of the definition, all mixed $z$-functions will be parallel to one of the $\gamma_c$ random vectors. See Figure 1 (right). This gives the following result, which guarantees the unique property of this continuous surrogate model, i.e., that all local minima are integer-valued in the intended variables:

Theorem 1. *If the surrogate model* $g$ *consists entirely of integer and mixed* $z$*-functions, then any strict local minimum* $(\mathbf{x}_c^*, \mathbf{x}_d^*)$ *of* $g$ *satisfies* $\mathbf{x}_d \in \mathbb{Z}^{\gamma_d}$.

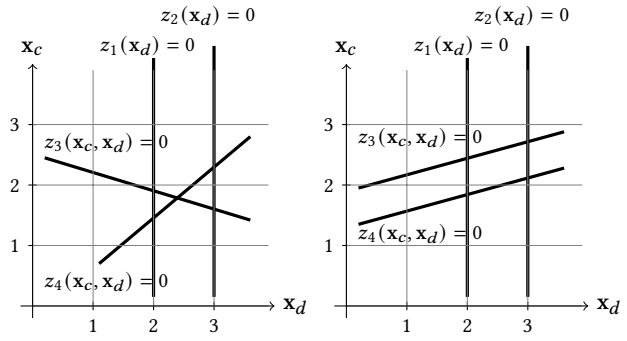


**Figure 1: (left) Example of the problem with mixed basis functions for $\gamma_d = 1$ integer ($\mathbf{x}_d$) and $\gamma_c = 1$ continuous variable ($\mathbf{x}_c$). All local minima are located in points where two lines $z_k(\mathbf{x}_c, \mathbf{x}_d) = 0$ intersect. This works fine for the intersections between the randomly chosen $z$-functions $z_3, z_4$ and the integer $z$-functions $z_1, z_2$, but not for the intersection of $z_3$ and $z_4$, as in that point $\mathbf{x}_d$ takes on a non-integer value. (right) A solution to the problem is to use mixed $z$-functions that are parallel to a number of linearly independent vectors equal to $\gamma_c$. This ensures that all intersections are located in points where $\mathbf{x}_d$ is integer.**

This result makes it possible to apply continuous optimisation to find a minimum of our surrogate model. This is much more efficient than having to solve a mixed-integer program, as discrete or mixed optimisation problems are generally more difficult to solve than continuous optimisation problems [29], and it also avoids having to resort to rounding which is sub-optimal. As the rectified linear units are linear almost everywhere, the surrogate model can be optimised relatively easily with a gradient-based technique such as L-BFGS [29] or other standard methods.

Before presenting the proof, we state two results that are relevant to our approach:

Lemma 1. *Any strict local minimum of $g$ is located in a point $(\mathbf{x}_c^*, \mathbf{x}_d^*)$ with $z_k(\mathbf{x}_c^*, \mathbf{x}_d^*) = 0$ for $(\gamma_c + \gamma_d)$ linearly independent functions $z_k$ [6].*

This follows from the fact that $g$ is piece-wise linear, so any strict local minimum must be located in a point where the model is nonlinear in all directions.

Lemma 2. *If $z_k(\mathbf{x}_d) = 0$ for $\gamma_d$ different linearly independent integer $z$-functions $z_k$, then $\mathbf{x}_d \in \mathbb{Z}^{\gamma_d}$.*

Proof. The proof follows exactly the same reasoning as the proof of [6, Thm. 2 (II)]. □

We now show the proof of Theorem 1 below.

Proof of Theorem 1. From Lemma 1 it follows that $z_k(\mathbf{x}_c^*, \mathbf{x}_d^*) = 0$ for $(\gamma_c + \gamma_d)$ linearly independent $z_k$. Since all mixed $z$-functions are parallel to one of the $\gamma_c$ randomly chosen vectors, there can only be $\gamma_c$ linearly independent mixed $z$-functions. As all other $z$-functions are integer $z$-functions, this means that there are $\gamma_d$ linearly independent integer $z$-functions. The result now follows from Lemma 2. □

## 4.2 MVRSM details

In the proposed algorithm, we first initialise the model by adding basis functions consisting of integer and mixed $z$-functions. The procedure of generating integer $z$-functions is the same as in the advanced model of [6], which gives $D_d = 1+4|X_d| - |X_d[1]| - |X_d[\gamma_d]|$ basis functions in total, with $X_d[i]$ the domain of the $i$-th integer variable. We then generate $D_c$ mixed $z$-functions as in Def. 2 (see also the technical appendix). Since our approach allows us to choose any number of mixed $z$-functions without losing the guarantee of satisfying the integer constraints, computational resources are the only limiting factor here. We choose $D_c = \lceil \gamma_c \cdot D_d / \gamma_d \rceil$ to have the same number of mixed $z$-functions per continuous variable as the number of integer $z$-functions per integer variable.

The algorithm proceeds with an iterative procedure consisting of four steps: **1)** evaluating the objective, **2)** updating the model, **3)** finding the minimum of the model, and **4)** performing an exploration step. Evaluating the objective $f$ gives a data sample $(\mathbf{x}_c, \mathbf{x}_d, y)$. The update procedure of the surrogate model is performed with the recursive least squares algorithm [26], which is possible because the model is linear in its parameters $c_k$. This procedure has a computational complexity of $O(D^2)$, where $D = D_d + D_c$ is the number of basis functions. Note that this number stays fixed for the whole duration of the algorithm, which causes the proposed algorithm to have a fixed computation time per iteration. We also add a regularisation factor of $10^{-8}$ in the recursive least squares update for numerical stability, using $L_2$ regularisation. Furthermore, the weights $c_k$ from (2) are initialised as $c_k = 1$ for the basis functions corresponding to integer $z$-functions, and as $c_k = 0$ for the basis functions corresponding to the mixed $z$-functions (see Appendix). The minimum of the model is found with the L-BFGS method [29], which is improved by giving an analytical representation of the Jacobian. For this purpose, we define $[\frac{d}{dx} \max\{0, x\}](0) = 0.5$, as the rectified linear units are non-differentiable in 0. We run the L-BFGS method for 20 sub-iterations only. The reason for this is that the surrogate model is only an approximation of the objective, so finding a promising area of the search space is more important than finding the exact minimum of the surrogate model. Since the number of basis functions in the model stays fixed, the computational complexity of this step also does not grow over time. Lastly, we perform an exploration step on the point $(\mathbf{x}_c^*, \mathbf{x}_d^*)$ found by the L-BFGS algorithm, where the point is given a small perturbation $(\delta_c, \delta_d)$ so that local optima can be avoided (see technical appendix). Without such a step, the surrogate model might return the exact same point multiple times and get stuck in its own local optimum. Every iteration ends with suggesting the new point for another evaluation of the objective, until the objective is evaluated $N$ times in total. The whole algorithm is shown in Algorithm 1.

# 5 EXPERIMENTS

To see if the proposed algorithm overcomes the drawbacks of existing surrogate modelling algorithms for problems with mixed variables in practice, we compare MVRSM with different state-of-the-art methods and random search on several synthetic benchmarks from related literature, with mixed-variable problems of up to 238 variables, and on two real-life benchmarks: XGBoost hyperparameter tuning and Electrostatic Precipitator optimisation.

**Algorithm 1** MVRSM algorithm

**Input** Objective $f$, domains $X_c$, $X_d$, budget $N$
**Output** $\mathbf{x}_c^{(N)}, \mathbf{x}_d^{(N)}, y^{(N)}$
Initialise surrogate $g$ with integer and mixed $z$-functions
Initialise $c_k = 1$ for integer $z$-functions and $c_k = 0$ for mixed $z$-functions, initialise other recursive least squares parameters
**for** $n = 1, \ldots, N$ **do**
Evaluate $y^{(n)} = f\left(\mathbf{x}_c^{(n)}, \mathbf{x}_d^{(n)}\right) + \epsilon$
Update the parameters of $g$ with data point $\left(\mathbf{x}_c^{(n)}, \mathbf{x}_d^{(n)}, y^{(n)}\right)$ using recursive least squares
Solve $\min g(\mathbf{x}_c, \mathbf{x}_d)$ over domains $X_c, X_d$ with relaxed integer constraints using L-BFGS
Explore around the found solution $(\mathbf{x}_c^*, \mathbf{x}_d^*)$ by adding random perturbation $(\delta_c, \delta_d) \in \mathbb{R}^{Y_c} \times \mathbb{Z}^{Y_d}$: $\left(\mathbf{x}_c^{(n+1)}, \mathbf{x}_d^{(n+1)}\right) = (\mathbf{x}_c^*, \mathbf{x}_d^*) + (\delta_c, \delta_d)$

In the comparison with other methods, we consider state-of-the-art surrogate modelling algorithms that are able to deal with a mixed-variable setting, have code available, and are concerned with single-objective problems. We compare our method with HyperOpt [3] (HO) and SMAC [12] as two popular and established surrogate modelling algorithms that can deal with mixed variables, and we compare with CoCaBO [25] as a more recent method that can deal with a mix of continuous and categorical variables. As is good practice in surrogate modelling, we include random search (RS) in the comparisons as a baseline, as well as a standard Bayesian optimisation (BO) algorithm, where we use rounding on the integer variables when calling the objective function.

Though we consider MiVaBO [9] also to be part of the state of the art, at the time of writing the authors have not made their code available yet. We still include their benchmarks in the comparison. As MiVaBO uses a more expensive optimisation method, we expect MVRSM to outperform MiVaBO in terms of efficiency, but further research is required to confirm this. We make no comparison with multi-fidelity methods such as Hyperband [19] or BOHB [10], or with the multi-objective methods from the related work section, as we did not find a way to make a fair comparison for single-objective single-fidelity problems.

## 5.1 Implementation details

To enable the use of categorical variables in MVRSM, we convert those variables to integers. To enable the use of integer or binary variables for CoCaBO, we convert those variables to categorical variables. For CoCaBO, we chose a mixture weight [25, Eq. (2)] of 0.5 as this seemed to give the best results on synthetic benchmarks. SMAC is put in deterministic mode instead of the default, as this improved the results in all of our experiments: the default often repeats function evaluations at the same location, leading to an inefficient method. The random search uses HyperOpt's implementation. The code for HyperOpt[1], SMAC[2], CoCaBO[3], and MVRSM[4] is availabe online. For Bayesian Optimisation we use an existing implementation[5] which uses Gaussian processes with a Matérn 5/2 kernel and the Upper Confidence Bound acquisition function $\mu(\mathbf{x}_c, \mathbf{x}_d) + \kappa\sigma(\mathbf{x}_c, \mathbf{x}_d)$, with $\mu$ and $\sigma$ the expected value and standard deviation of the Gaussian process respectively, and with $\kappa = 2.576$ as a default hyperparameter. Experiments were done in Python on an Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz with 32 GB of RAM, and each experiment was performed using only a single CPU core. In line with [25], all methods start with 24 initial random guesses from a uniform distribution, but these are not shown in the figures.

All methods are compared using the same number of iterations, and the best function value found at each iteration is reported, averaged over multiple runs. The standard deviations are indicated with shaded areas in the relevant figures. The computation time of the methods is also reported for every iteration.

## 5.2 Results on relevant synthetic benchmarks

In this section we investigate the drawbacks of other algorithms that were mentioned in Section 3, and see if MVRSM overcomes them. The main drawbacks that were mentioned were: algorithms slowing down over time, limited interaction between variables, and having to resort to rounding. We make a comparison on several large-scale synthetic functions from related literature, including literature on the competing algorithms.

The Ackley functions is a well-known benchmark in the black-box optimisation community that can be scaled to any dimension[6]. We choose a dimension of 53, but 50 of the variables were adapted to binary variables in $X_d = \{0, 1\}^{50}$. The 3 continuous variables were limited to $X_c = [-1, 1]^3$. This causes the problem to be of a similar scale as the problem of variational auto-encoder hyperparameter tuning after binarising the discrete hyperparameters [9, Sec. 4.2]. Uniform noise in $[0, 10^{-6}]$ was added to each function evaluation. We also investigated a publicly available mixed-variable benchmark function, namely cvxnonsep_psig40 from the MINLP library[7]. Finally, we investigated a randomly generated synthetic test function from [9, Appendix C.1, Gaussian weights variant]. We scaled this problem up to have 119 integer and 119 continuous variables. No bounds were reported for this problem so we set them to $X_d = \{0, 1, 2, 3\}^{119}$ for the integer variables and $X_c = [0, 3]^{119}$ for the continuous variables.

Figures 2-4 show the performance of the different algorithms on these three benchmarks, both the best found objective at each iteration (left plots) as well as the computation time used by the algorithm at each iteration (right plots). Note that the performance of surrogate modelling algorithms is usually plotted against the number of function evaluations (iteration), as this is assumed to be the bottleneck in expensive optimisation problems, but we are still interested in their computation time as well. MVRSM clearly outperforms the other methods in terms of accuracy, and the computation times of BO and CoCaBO become prohibitively large. Thus, BO and CoCaBO are not evaluated on all problems. The slowdown of the other surrogate-based algorithms except SMAC is clearly visible, with their computation time increasing every iteration. Random

[1] https://github.com/hyperopt/hyperopt
[2] https://github.com/automl/SMAC3
[3] https://github.com/rubinxin/CoCaBO_code
[4] DOI removed for double-blind reviewing
[5] https://github.com/fmfn/BayesianOptimization
[6] https://www.sfu.ca/~ssurjano/optimization.html
[7] https://www.minlplib.org/cvxnonsep_psig40.html

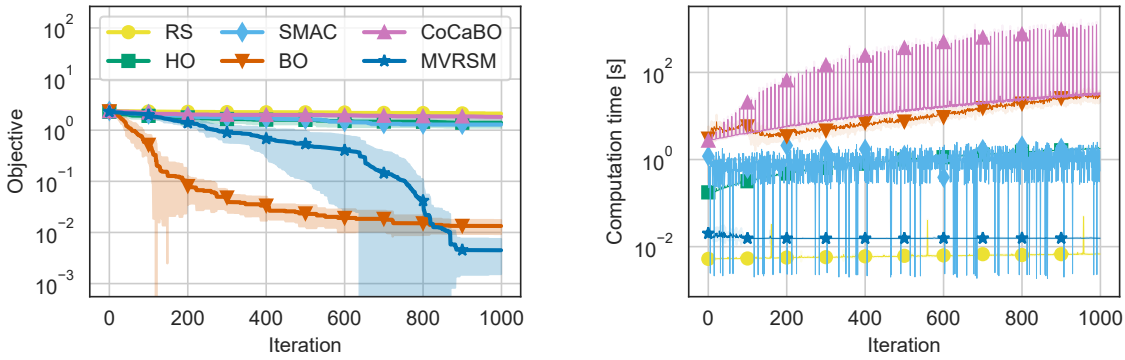


**Figure 2: Results on the Ackley53 benchmark (50 binary, 3 continuous), averaged over 7 runs. Note that the figure has a logarithmic scale. This problem is of a similar scale as variational auto-encoder hyperparameter tuning [9, Sec. 4.2] .**

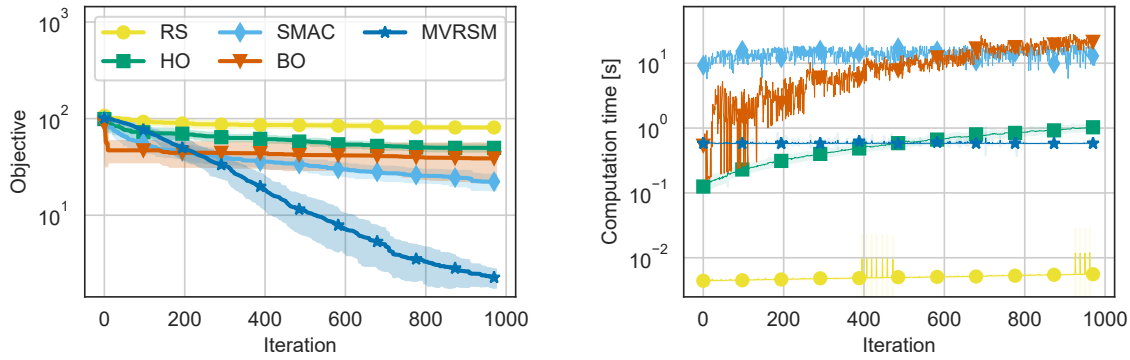


**Figure 3: Results on the cvxnonsep_psig40 benchmark (20 integer, 20 continuous), averaged over 7 runs. The known objective value at the global optimum has been subtracted from the values in the left plot to allow a logarithmic scale.**

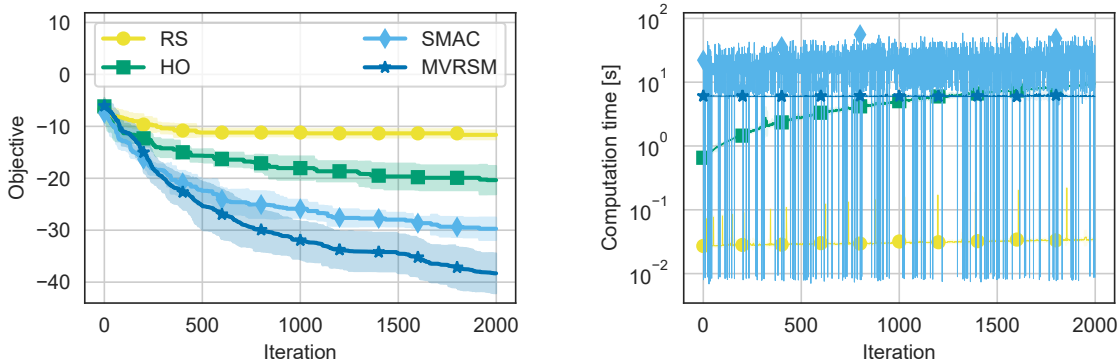


**Figure 4: Results on one randomly generated MiVaBO synthetic benchmark [9, Appendix C.1, Gaussian weights variant] (119 integer, 119 continuous), averaged over 7 runs. BO and CoCaBO were not evaluated for this benchmark due to the large computation time. This problem is of a similar scale as feed-forward classification model hyperparameter tuning [3].**

search, while taking negligible computation time, does not have a good performance if the function evaluations are assumed to be the bottleneck. The large variance in computation time for SMAC can be explained by this method's way of alternating between using points suggested by its surrogate model and using randomly chosen points.

The fact that MVRSM outperforms both HO and SMAC even with a constant runtime per iteration is surprising, considering that the scale of the largest problem is similar to that of one of HO's own benchmarks, while the authors of HO consider SMAC a potentially superior optimiser [3, p. 8].

### 5.3 Results on XGBoost hyperparameter tuning

To see if the good performance of MVRSM on synthetic problems also holds in a real application, we consider a problem similar to that of hyperopt-sklearn [18], where hyperparameters for a preprocessing method as well as for a classifier need to be selected and tuned simultaneously. The choice of classifier is limited to the XGBoost method only [8], which has several hyperparameters of different shapes (continuous, integer, binary, categorical, and conditional).[8]

Conditional variables only exist when other variables take on certain values. SMAC and HO can both deal with these efficiently, but for the other methods, including MVRSM, we use a naïve encoding where these variables still exist but do not influence the objective function if other choices are made. Together with the hyperparameters for preprocessing, there are 7 integer, 11 continuous, and 117 categorical/binary/conditional variables. We included the possibility to use different preprocessing methods for different features of the dataset that was used. The preprocessing method and XGBoost are applied to the steel-plates-faults dataset[9], and the objective is the result of a 5-fold cross-validation, multiplied by $-1$ to make it a minimisation problem. To find not just accurate but also efficient hyperparameters, we set a time limit of 8 seconds, chosen roughly equal to twice the time it takes when using default hyperparameters. If the objective took longer than that to evaluate, an objective value of 0 was returned. On average, the evaluation of the objective took just over 3 seconds on our hardware.

Figure 5 shows the results on this benchmark for 200 iterations, averaged over 7 runs. While HO is an efficient and accurate method on this problem, MVRSM achieves a similar performance as its competitors, ending up with an average objective of $-0.637$. A pair-wise Student's T-test on the final iteration shows no significant difference between MVRSM and the other surrogate-based methods ($p > 0.05$), though it outperforms random search ($p \approx 0.003$).

It is important to note that besides random search, MVRSM is the only method that has a fixed computation time per iteration. This is especially important for problems where the evaluation time of the objective takes a similar time as the surrogate-based algorithm, e.g. 10 seconds or less for CoCaBO, which is the case for this hyperparameter tuning problem. In this case it is not possible anymore to disregard the computation time of the algorithm, even though this is often done in literature. Furthermore, CoCaBO tunes its own hyperparameters every 10 iterations, which costs even more computational resources. In contrast, MVRSM has quite a low number of hyperparameters, and we choose them the same way in all reported experiments. This makes it much easier to apply than other methods, or in the case of CoCaBO, much more efficient. The practical use of this fact should not be underestimated, as especially on hyperparameter tuning problems one wants to avoid having to tune the hyperparameters of the surrogate-based algorithm.

[8]The hyperparameters for XGBoost can be found at https://xgboost.readthedocs.io/en/latest/parameter.html#learning-task-parameters

[9]https://archive.ics.uci.edu/ml/datasets/Steel+Plates+Faults

## 6 RESULTS ON ELECTROSTATIC PRECIPITATOR OPTIMISATION

To test MVRSM on a real-life application from engineering, we applied it to the ESP problem [24]. This is a real-life industrial problem where components of a gas cleaning system need to be designed. The goal is to reduce environmental pollution. The system contains 49 different slots that can each hold one of 8 different types of metal plates that each influence the gas flow in a different way. After choosing the configuration of the plates, an expensive computational fluid dynamics simulator calculates the corresponding objective, taking around 27 seconds on average on our hardware. This problem has 8 categories for each variable, though 5 of the categories correspond to ordinal variables, namely the size of holes in the metal plates.

We have adapted the ESP problem such that the 5 hole sizes are not restricted to fixed values, but are free to take on different continuous values. This adds 5 continuous variables to the problem with otherwise only categorical variables, using the same five options for each slot, as having each slot take on a different value would substantially increase the manufacturing costs.

Figure 6 shows the results on this benchmark averaged over 7 runs, for 1000 iterations, of which 24 were again used as initial random guesses. BO and CoCaBO are not included as they contained iterations where the computation time exceeded the time to evaluate the expensive objective function, making them unsuitable for such an application.

Again MVRSM has a similar performance as its competitors, ending up with an average objective of 1.004, and in total it is over 4 times faster than SMAC, the method that achieves the lowest average objective. A pair-wise Student's T-test on the final iteration shows no significant difference between MVRSM and the other surrogate-based methods ($p > 0.05$), though it outperforms random search ($p \approx 0.0001$). However, it is the only surrogate-based method with a constant computation time per iteration.

## 7 CONCLUSION AND FUTURE WORK

We showed how Mixed-Variable ReLU-based Surrogate Modelling (MVRSM) solves three problems present in methods that can deal with mixed variables in expensive black-box optimisation. First, it solves the problem of slowing down over time due to a growing surrogate model. Second, it solves the problem of sub-optimality and inefficiency that may arise due to the need to satisfy integer constraints. Third, it solves the problem of limited interaction between the mixed variables. MVRSM's surrogate model, based on a linear combination of rectified linear units, avoids all of these problems by having a fixed number of basis functions that contain interaction

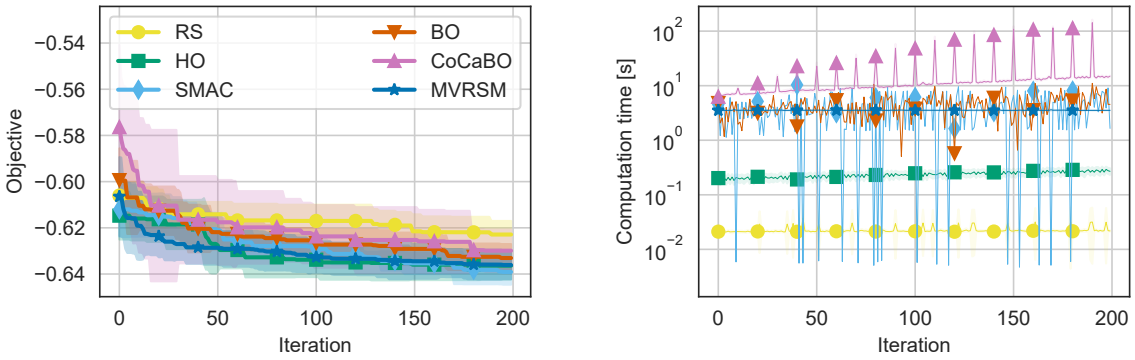


**Figure 5: Results on the XGBoost hyperparameter tuning benchmark (7 integer, 11 continuous, 117 categorical/binary/conditional), averaged over 7 runs.**

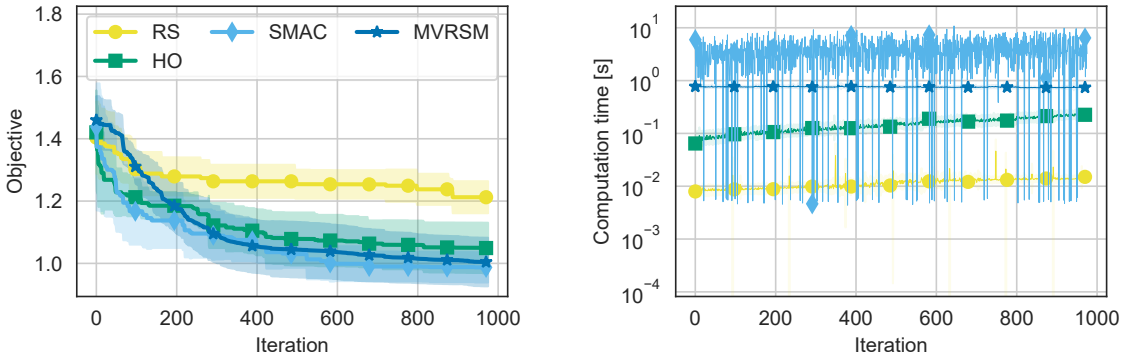


**Figure 6: Results on the ESP benchmark (49 categorical, 5 continuous), averaged over 7 runs.**

between all variables, while also having the guarantee that any local optimum is located in points where the integer constraints are satisfied. This last point is a unique selling point of MVRSM, as it has only been used in the context of surrogate models once before, and never for the mixed-variable case. Together, all these properties cause MVRSM to give competitive performance on two real-life benchmarks from computer science and engineering, while outperforming the state-of-the-art in both speed and accuracy on various synthetic problems. All of this is achieved using the same hyperparameter settings for MVRSM, while for other methods it might be necessary to spend some time on finding the right settings.

For future work we will adapt the method to efficiently deal with constraints, as well as use the theory and insights from this work to develop a new surrogate-assisted evolutionary algorithm.

## ACKNOWLEDGMENTS

This work is part of the research programme Real-time data-driven maintenance logistics with project number 628.009.012, which is financed by the Dutch Research Council (NWO). The authors thank Erik Daxberger for providing the code for generating one of MiVaBO's synthetic test functions, Frederik Rehbach for providing information on the ESP problem, and anonymous reviewers of an earlier version of this paper for providing constructive feedback.

## REFERENCES

[1] R. Baptista and M. Poloczek. Bayesian optimization of combinatorial structures. In *ICML*, pages 471–480, 2018.
[2] T. Bartz-Beielstein and M. Zaefferer. Model-based methods for continuous and discrete global optimization. *Applied Soft Computing*, 55:154–167, 2017.
[3] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML - Volume 28*, pages I–115, 2013.
[4] L. Bliek, M. Verhaegen, and S. Wahls. Online function minimization with convex random ReLU expansions. In *MLSP*, pages 1–6. IEEE, 2017.
[5] L. Bliek, H. R. Verstraete, M. Verhaegen, and S. Wahls. Online optimization with costly and noisy measurements using random Fourier expansions. *IEEE Transactions on Neural Networks and Learning Systems*, 29(1):167–182, Jan 2018.
[6] L. Bliek, S. Verwer, and M. de Weerdt. Black-box combinatorial optimization using models with integer-valued minima. *Annals of Mathematics and Artificial Intelligence*, pages 1–15, 2020.
[7] K. V. D. Blom, T. Deist, V. Volz, M. Marchi, Y. Nojima, B. Naujoks, A. Oyama, and T. Tusar. Identifying properties of real-world optimisation problems through a questionnaire. *arXiv preprint arXiv:2011.05547*, 2020.
[8] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *ACM SIGKDD*, pages 785–794, 2016.
[9] E. Daxberger, A. Makarova, M. Turchetta, and A. Krause. Mixed-variable Bayesian optimization. In *IJCAI*, pages 2633–2639, 2020.
[10] S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. 2018.

[11] E. C. Garrido-Merchán and D. Hernández-Lobato. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing*, 380:20–35, 2020.
[12] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, pages 507–523. Springer, 2011.
[13] F. Hutter, L. Kotthoff, and J. Vanschoren. *Automated Machine Learning*. Springer, 2019.
[14] A. Iyer, Y. Zhang, A. Prasad, S. Tao, Y. Wang, L. Schadler, L. C. Brinson, and W. Chen. Data-centric mixed-variable Bayesian optimization for materials design. In *ASME*. American Society of Mechanical Engineers Digital Collection, 2019.
[15] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm Evol. Comput.*, 1:61–70, 2011.
[16] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.
[17] S. H. Kim and F. Boukouvala. Surrogate-based optimization for mixed-integer nonlinear problems. *Computers & Chemical Engineering*, page 106847, 2020.
[18] B. Komer, J. Bergstra, and C. Eliasmith. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In *ICML workshop on AutoML*, volume 9, page 50. Citeseer, 2014.
[19] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.
[20] J. Močkus. On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.
[21] J. Močkus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.
[22] D. Nguyen, S. Gupta, S. Rana, A. Shilton, and S. Venkatesh. Bayesian optimization for categorical and category-specific continuous inputs. In *AAAI*, 2020.
[23] A. Rahimi and B. Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561. IEEE, 2008.
[24] F. Rehbach, M. Zaefferer, J. Stork, and T. Bartz-Beielstein. Comparison of parallel surrogate-assisted optimization approaches. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, page 1348–1355, New York, NY, USA, 2018. Association for Computing Machinery.
[25] B. Ru, A. Alvi, V. Nguyen, M. A. Osborne, and S. Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *ICML*, pages 8276–8285, 2020.
[26] A. H. Sayed and T. Kailath. Recursive least-squares adaptive filters. *The Digital Signal Processing Handbook*, 21(1), 1998.
[27] T. Ueno, T. D. Rhone, Z. Hou, T. Mizoguchi, and K. Tsuda. COMBO: An efficient Bayesian optimization library for materials science. *Materials discovery*, 4:18–21, 2016.
[28] X. Wan, V. Nguyen, H. Ha, B. Ru, C. Lu, and M. A. Osborne. Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. *arXiv preprint arXiv:2102.07188*, 2021.
[29] S. Wright and J. Nocedal. Numerical optimization. *Springer Science*, 35:67–68, 1999.
[30] K. Yang, K. van der Blom, T. Bäck, and M. Emmerich. Towards single-and multiobjective Bayesian global optimization for mixed integer problems. In *Proceedings of the 14th International Global Optimization workshop*, volume 2070, page 020044, 2019.