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

Anonymous Authors

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. 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: XGBoost hyperparameter tuning and Electrostatic Precipitator optimisation.

Introduction

Surrogate modelling techniques such as Bayesian optimisation have a long history of success in optimising expensive black-box objective functions (Močkus 1975; Jones, Schonlau, and Welch 1998; Močkus 2012). 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.

While most attention in the literature has gone to problems in continuous domains, recently solutions for combinatorial optimisation problems have started to arise (Garrido-Merchán and Hernández-Lobato 2020; Baptista and Poloczek 2018; Bartz-Beielstein and Zaefferer 2017; Ueno et al. 2016; Bliek, Verwer, and de Weerdt 2019). Yet many problems contain a mix of continuous and discrete variables, for example material design (Iyer et al. 2019) and automated machine learning (Hutter, Kotthoff, and Vanschoren 2019). 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 (Garrido-Merchán and Hernández-Lobato 2020; Ru et al. 2019). 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 will perform poorly, 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 local optima that are located exactly in points of the search space 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. Finally, the last point eliminates the need for rounding techniques, and also eliminates the need for repeatedly using integer programming as is done by Daxberger et al. (2019).

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 an experimental proof of the effectiveness of this method on several large-scale synthetic benchmarks from related work and on two real-life benchmarks: XGBoost hyperparameter tuning and Electrostatic Precipitator optimisation.

Preliminaries

This work considers the problem of finding the minimum of a mixed-variable black-box objective function $f : \mathbb{R}^{d_c} \times$

$\mathbb{Z}^{d_d} \rightarrow \mathbb{R}$ 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

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

where d_c is the number of continuous variables, d_d the number of integer variables, $\epsilon \in \mathbb{R}$ is a zero-mean random variable with finite variance, and $X_c \subseteq \mathbb{R}^{d_c}$ and $X_d \subseteq \mathbb{Z}^{d_d}$ 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 $X_d \subseteq \mathbb{Z}^{d_d}$, we call $\mathbf{x}_d \in \mathbb{Z}^{d_d}$ 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 (Bergstra, Yamins, and Cox 2013) and many engineering problems (Bliek et al. 2018; Ueno et al. 2016). Therefore, we wish to solve (1) using as few samples as possible.

The problem is usually solved with a surrogate modelling technique such as Bayesian optimisation (Močkus 2012). 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.

Related work

In Bayesian optimisation, Gaussian processes are the most popular surrogate model (Močkus 2012). 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 (Garrido-Merchán and Hernández-Lobato 2020; Baptista and Poloczek 2018; Ueno et al. 2016; Bliek, Verwer, and de Weerdt 2019).

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 (Hutter, Hoos, and Leyton-Brown 2011) and HyperOpt (Bergstra, Yamins, and Cox 2013), 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 (Hutter, Hoos, and Leyton-Brown 2011) uses random forests as the surrogate model. This captures interactions between the variables nicely, but the main disadvantage is that the random forests are less accurate in unseen parts of the search space, at least compared to other surrogate models. HyperOpt (Bergstra, Yamins, and Cox 2013) 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 (Garrido-Merchán and Hernández-Lobato 2020; Ru et al. 2019).

More recently, the CoCaBO algorithm was proposed (Ru et al. 2019), 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. Other research groups have focused their attention to multi-objective mixed-variable problems (Yang et al. 2019; Iyer et al. 2019).

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 (Bliek et al. 2018) and for the discrete setting in the COMBO (Ueno et al. 2016) and IDONE algorithms (Bliek, Verwer, and de Weerdt 2019) by making use of parametric surrogate models that are linear in the parameters. The MiVaBO algorithm (Daxberger et al. 2019) 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.

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.

Proposed surrogate model

As in related work (Bliek, Verhaegen, and Wahls 2017; Bliek, Verwer, and de Weerdt 2019; Daxberger et al. 2019), we use a continuous surrogate model $g : \mathbb{R}^{d_c+d_d} \to \mathbb{R}$:

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

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 ϕ in such a way that all local optima $(\mathbf{x}_c^*, \mathbf{x}_d^*)$ of the model satisfy $\mathbf{x}_d \in \mathbb{Z}^{d_d}$, 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 $\mathbf{v}_k \in \mathbb{R}^{d_c}$, $\mathbf{w}_k \in \mathbb{R}^{d_d}$, 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 (Močkus 2012; Ru et al. 2019),
- choose them randomly once and then fix them (Bliek et al. 2018; Bliek, Verhaegen, and Wahls 2017; Ueno et al. 2016; Daxberger et al. 2019), or
- choose them according to the variable domains X_c, X_d and then fix them (Bliek, Verwer, and de Weerdt 2019).

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 (Rahimi and Recht 2008; Bliek et al. 2018), 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 (Bliek, Verwer, and de Weerdt 2019). 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.*

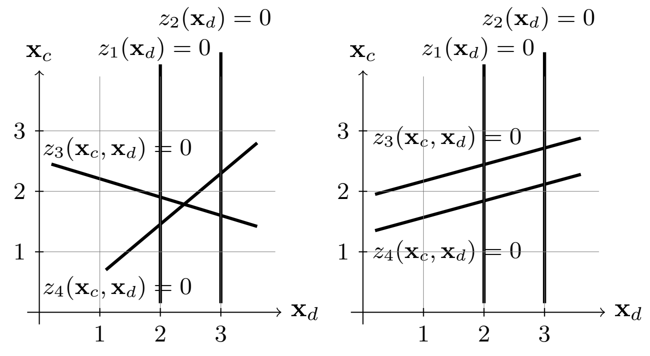

Figure 1: **(left)** Example of the problem with mixed basis functions for 1 integer ($\mathbf{x}_d$) and 1 continuous variable ($\mathbf{x}_c$). All local minima are located in points where two lines intersect. This works fine for the intersections with the integer z-functions z_1, z_2, but not for the two randomly chosen z-functions z_3, 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 d_c. This ensures that all intersections are located in points where $\mathbf{x}_d$ is integer.

By making use of the integer z-functions, we have a surrogate model with basis functions that depend on the integer variables. 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 (Daxberger et al. 2019), 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 (Daxberger et al. 2019), but we choose them pseudo-randomly rather than randomly. This benefits from the success that randomly chosen weights have had in the past (Bliek et al. 2018; Bliek, Verhaegen, and Wahls 2017; Ueno et al. 2016; Daxberger et al. 2019), 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* Ω *that contains* d_c *random vectors in* $\mathbb{R}^{d_c+d_d}$ *with a continuous probability distribution* p_ω*, and* b_k *is then chosen from a random continuous probability distribution* p_b *which depends on* ω_k*. This results in a basis function that depends on all continuous and on all integer variables.*

The probability distributions p_ω 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 supplementary material.) As a result of the definition, all mixed z-functions will be parallel to one of the d_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 $\mathbf{x}_d \in \mathbb{Z}^{d_d}$.*

This result makes it possible to apply continuous optimisation to find a minimum of our surrogate model, instead of having to solve a mixed-integer program which is more expensive, or having to resort to rounding which is suboptimal. 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 (Wright and Nocedal 1999) 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 $(d_c + d_d)$ linearly independent functions z_k (Bliek, Verwer, and de Weerdt 2019).*

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 d_d different linearly independent integer z-functions z_k, then $\mathbf{x}_d \in \mathbb{Z}^{d_d}$.*

Proof. The proof follows exactly the same reasoning as the proof of (Bliek, Verwer, and de Weerdt 2019, 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 $d_c + d_d$ linearly independent z_k. Since all mixed z-functions are parallel to one of the d_c randomly chosen vectors, there can only be d_c linearly independent mixed z-functions. As all other z-functions are integer z-functions, this means that there are d_d linearly independent integer z-functions. The result now follows from Lemma 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 (Bliek, Verwer, and de Weerdt 2019), which gives $D_d = 1 + 4|X_d| - |X_d[1]| - |X_d[d_d]|$ basis functions in total, with $X_d[i]$ the domain of the i-th integer variable. We then generate D_c mixed z-functions. 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 $D_c = \lceil d_c \cdot D_d / d_d \rceil$ 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 (Sayed and Kailath 1998), which can be done since the model is linear in its parameters c_k. We also add a regularisation factor of 10^{-8} here for numerical stability. 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. The minimum of the model is found with the L-BFGS method (Wright and Nocedal 1999), 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, as the goal is not to find the exact minimum of the surrogate model, but rather to find a promising area of the search space. 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 so that local optima can be avoided. The whole algorithm is shown in Algorithm 1.

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 $(\delta_c, \delta_d) \in \mathbb{R}^{d_c} \times \mathbb{Z}^{d_d}$:
    $\left(\mathbf{x}_c^{(n+1)}, \mathbf{x}_d^{(n+1)}\right) = (\mathbf{x}_c^*, \mathbf{x}_d^*) + (\delta_c, \delta_d)$
```

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 two real-life benchmarks and on several synthetic benchmark functions used in related work. For the real-life benchmarks we consider one from machine learning and one from engineering, namely XGBoost hyperparameter tuning and Electrostatic Precipitator (ESP) optimisation. For the synthetic benchmarks we consider mixed-variable problems of up to 238 variables from related literature.

For 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 (Bergstra, Yamins, and Cox 2013) (HO) and SMAC (Hutter, Hoos, and Leyton-Brown 2011) as two popular and established surrogate modelling algorithms that can deal with mixed variables, and we compare with CoCaBO (Ru et al. 2019) 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 to confirm whether more sophisticated methods are even necessary. For the same reason, we include a standard Bayesian optimisation (BO) algorithm, where we use rounding on the integer variables when calling the objective function.

Though we consider MiVaBO (Daxberger et al. 2019) 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. We make no comparison with multi-fidelity methods such as Hyperband (Li et al. 2017) or BOHB (Falkner, Klein, and Hutter 2018), as these methods can only be applied to our hyperparameter tuning benchmark and not to the other benchmarks. We also did not compare 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 problems, even though they were specifically developed for the mixed-variable setting. Because MiVaBO uses a more expensive optimisation method, we expect MVRSM to outperform not only multi-objective methods but also MiVaBO on single-objective domains in terms of efficiency, but further research is required to confirm this.

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 (Ru et al. 2019, 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 the Upper Confidence Bound acquisition function. 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 (Ru et al. 2019), all methods start with 24 initial random guesses, which are not shown in the figures. We used each algorithm's own implementation for this, but made sure to set it to the same uniform probability distribution over the whole search space.

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.

Results on XGBoost hyperparameter tuning

First, we consider a problem similar to that of hyperopt-sklearn (Komer, Bergstra, and Eliasmith 2014), 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 (Chen and Guestrin 2016), which has several hyperparameters of different shapes (continuous, integer, binary, categorical, and conditional).[6]

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 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 over 116 categorical/binary/conditional variables. The preprocessing method and XGBoost are applied to the steel-plates-faults dataset[7], 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 2 shows the results on this benchmark for 200 iterations, averaged over 10 runs. MVRSM gets a similar performance as its competitors on this problem, 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. All other methods (except SMAC, as shown later in this paper) become slower over time. This is especially important for problems where the evaluation time of the objective takes a similar time as the surrogate-based

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

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

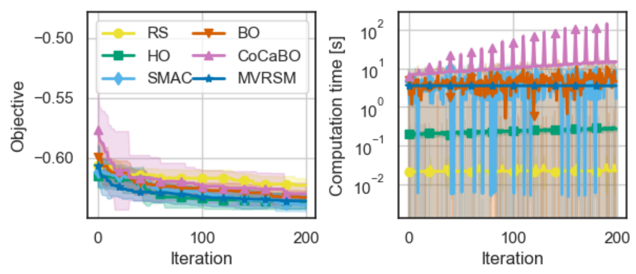

Figure 2: Results on the XGBoost hyperparameter tuning benchmark (7 integer, 11 continuous, >116 categorical/binary/conditional), averaged over 7 runs.

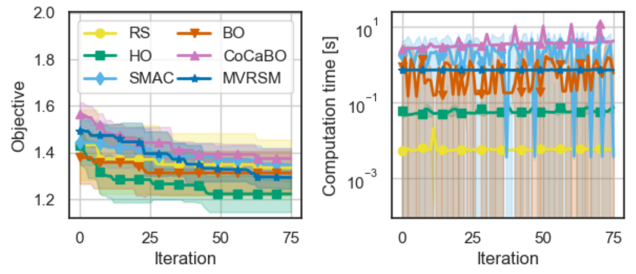

Figure 3: Results on the ESP benchmark (49 categorical, 5 continuous), averaged over 5 runs.

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.

Results on Electrostatic Precipitator optimisation

The ESP problem (Rehbach et al. 2018) 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 3 shows the results on this benchmark for 76 iterations, as the problem typically has a budget of 100 function evaluations (Rehbach, Rebolledo, and Bartz-Beielstein 2020) and we used 24 of them for random initial guesses. MVRSM ends up with an average objective of 1.29. A pairwise Student's T-test on the final iteration shows no significant difference between MVRSM and the other methods ($p \approx 0.13$ when compared with HO), except when comparing with CoCaBO ($p \approx 0.024$) which performs more poorly on this problem. This indicates that MVRSM is a competitive method in realistic expensive optimisation problems. However, the effect of slowdown in the other algorithms is not clearly visible due to the low number of iterations used. The real life benchmarks are too expensive to evaluate for a large number of iterations, which is why we now turn to investigate synthetic benchmarks. Besides a larger number of function evaluations, the use of synthetic benchmarks also allow us to investigate the performance of MVRSM on large-scale problems.

Results on relevant synthetic benchmarks

To investigate the effect of algorithms slowing down, as well as the scalability of MVRSM and how it compares to other algorithms on their own benchmarks, we make a comparison on several large-scale synthetic functions from related literature. The Ackley and Rosenbrock functions are two well-known benchmarks in the black-box optimisation community[8]. Both can be scaled to any dimension. For the Ackley function 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 (Daxberger et al. 2019, App. E.1). For the Rosenbrock function we choose a dimension of 239, with the first 119 variables adapted to integers in $X_d = \{-2,-1,0,1,2\}^{119}$, and 119 continuous variables limited to $X_c = [-2,2]^{119}$. The function was scaled with a factor $1/50000$. This problem is of the same scale as the problem of feed-forward classification model hyperparameter tuning (Bergstra, Yamins, and Cox 2013), except that the ratio between continuous and integer variables is chosen to be $1:1$. Uniform noise in $[0,10^{-6}]$ was added to each function evaluation in both functions. Finally, we investigated a randomly generated synthetic test function from (Daxberger et al. 2019, 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.

[8]Details available at https://www.sfu.ca/~ssurjano/optimization.html

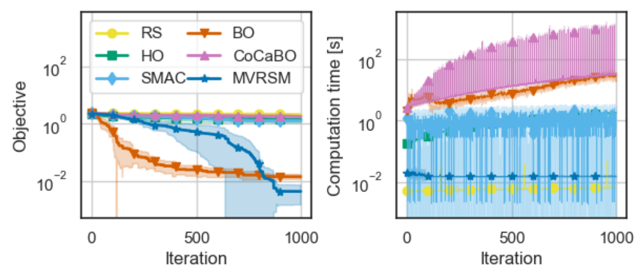

Figure 4: Results on the Ackley53 benchmark (50 binary, 3 continuous), averaged over 7 runs. Note that the left figure has a logarithmic scale. This problem is of a similar scale as variational auto-encoder hyperparameter tuning (Daxberger et al. 2019, Sec. 4.2).

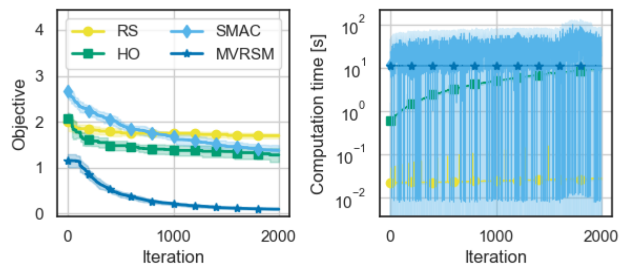

Figure 5: Results on the Rosenbrock238 benchmark (119 integer, 119 continuous), averaged over 7 runs. BO and CoCaBO were not evaluated for this benchmark due to the large computation time. This problem is of a similar scale as feed-forward classification model hyperparameter tuning (Bergstra, Yamins, and Cox 2013).

Figures 4-6 show the performance of the different algorithms on these three benchmarks. MVRSM clearly outperforms the other methods in terms of accuracy, and the computation times of BO and CoCaBO become prohibitively large. The slowdown of the other surrogate-based algorithms is now clearly visible, with their computation time increasing every iteration, although SMAC does not suffer from this.

The fact that MVRSM outperforms both HO and SMAC is surprising, considering that the scale of the larger problems is similar to that of one of HO's own benchmarks, while the authors of HO consider SMAC a potentially superior optimiser (Bergstra, Yamins, and Cox 2013, p. 8).

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 model inaccuracies due to limited interaction between the mixed variables. MVRSM's

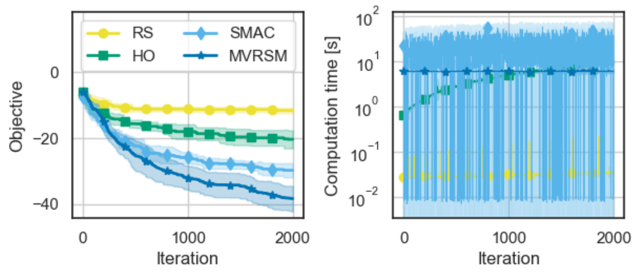

Figure 6: Results on one randomly generated MiVaBO synthetic benchmark (Daxberger et al. 2019, Appendix C.1, Gaussian weights variant) with a larger scale (119 integer, 119 continuous), averaged over 7 runs. BO and CoCaBO were not evaluated for this benchmark due to the large computation time. This problem is of a similar scale as feed-forward classification model hyperparameter tuning (Bergstra, Yamins, and Cox 2013).

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 between all variables, while also having the guarantee that any local optimum is located in points where the integer constraints are satisfied. These properties cause MVRSM to give competitive performance on two real-life benchmarks, which we have shown experimentally. It also makes MVRSM more accurate than the state-of-the-art on large-scale synthetic problems (e.g. >50 variables) and more efficient than most competitors. 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 investigate the exploration part of the surrogate model, for example by applying techniques with more theoretical guarantees such as Thompson sampling, and adapt the method to efficiently deal with categorical and conditional variables and with constraints.

References

Baptista, R.; and Poloczek, M. 2018. Bayesian Optimization of Combinatorial Structures. In *ICML*, 471–480.

Bartz-Beielstein, T.; and Zaefferer, M. 2017. Model-based methods for continuous and discrete global optimization. *Applied Soft Computing* 55: 154–167.

Bergstra, J.; Yamins, D.; and Cox, D. 2013. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML - Volume 28*, I–115.

Bliek, L.; Verhaegen, M.; and Wahls, S. 2017. Online function minimization with convex random ReLU expansions. In *MLSP*, 1–6. IEEE.

Bliek, L.; Verstraete, H. R. G. W.; Verhaegen, M.; and Wahls, S. 2018. Online Optimization With Costly and Noisy Measurements Using Random Fourier Expansions. *IEEE Transactions on Neural Networks and Learning Systems* 29(1): 167–182. ISSN 2162-237X.

Bliek, L.; Verwer, S.; and de Weerdt, M. 2019. Black-box Combinatorial Optimization using Models with Integer-valued Minima. *arXiv preprint arXiv:1911.08817* .

Chen, T.; and Guestrin, C. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 785–794.

Daxberger, E.; Makarova, A.; Turchetta, M.; and Krause, A. 2019. Mixed-Variable Bayesian Optimization. *arXiv preprint arXiv:1907.01329* .

Falkner, S.; Klein, A.; and Hutter, F. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774* .

Garrido-Merchán, E. C.; and Hernández-Lobato, D. 2020. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing* 380: 20–35.

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, 507–523. Springer.

Hutter, F.; Kotthoff, L.; and Vanschoren, J. 2019. *Automated Machine Learning*. Springer.

Iyer, A.; Zhang, Y.; Prasad, A.; Tao, S.; Wang, Y.; Schadler, L.; Brinson, L. C.; and Chen, W. 2019. Data-Centric Mixed-Variable Bayesian Optimization For Materials Design. In *ASME*. American Society of Mechanical Engineers Digital Collection.

Jones, D. R.; Schonlau, M.; and Welch, W. J. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13(4): 455–492.

Komer, B.; Bergstra, J.; and Eliasmith, C. 2014. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In *ICML workshop on AutoML*, volume 9, 50. Citeseer.

Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; and Talwalkar, A. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* 18(1): 6765–6816.

Močkus, J. 1975. On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, 400–404. Springer.

Močkus, J. 2012. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media.

Rahimi, A.; and Recht, B. 2008. Uniform approximation of functions with random bases. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, 555–561. IEEE.

Rehbach, F.; Rebolledo, M.; and Bartz-Beielstein, T. 2020. GECCO2020 Industrial Challenge. https://www.th-koeln.de/informatik-und-ingenieurwissenschaften/gecco-challenge-2020_72989.php. Accessed 30-06-2020.

Rehbach, F.; Zaefferer, M.; Stork, J.; and Bartz-Beielstein, T. 2018. Comparison of Parallel Surrogate-Assisted Optimization Approaches. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, 1348–1355. New York, NY, USA: Association for Computing Machinery. ISBN 9781450356183. doi:10.1145/3205455.3205587.

Ru, B.; Alvi, A. S.; Nguyen, V.; Osborne, M. A.; and Roberts, S. J. 2019. Bayesian optimisation over multiple continuous and categorical inputs. *arXiv preprint arXiv:1906.08878* .

Sayed, A. H.; and Kailath, T. 1998. Recursive least-squares adaptive filters. *The Digital Signal Processing Handbook* 21(1).

Ueno, T.; Rhone, T. D.; Hou, Z.; Mizoguchi, T.; and Tsuda, K. 2016. COMBO: An efficient Bayesian optimization library for materials science. *Materials discovery* 4: 18–21.

Wright, S.; and Nocedal, J. 1999. Numerical optimization. *Springer Science* 35: 67–68.

Yang, K.; van der Blom, K.; Bäck, T.; and Emmerich, M. 2019. Towards single-and multiobjective Bayesian global optimization for mixed integer problems. In *Proceedings of the 14th International Global Optimization workshop*, volume 2070, 020044. AIP Publishing LLC.