Technical Appendix for: Black-box Mixed-Variable Optimisation using a Surrogate Model that Satisfies Integer Constraints

Anonymous Authors

Details for generating mixed basis functions

In this section we show how to choose p_ω and p_b from Definition 2 of the main paper in such a way that the mixed z-functions are never completely outside the domain $X_c \times X_d$. We recommend to choose p_ω to be a uniform distribution over $[-\frac{1}{d_c+d_d}, \frac{1}{d_c+d_d}]^{d_c+d_d}$. This way, the term $\mathbf{v}_k^T \mathbf{x}_c + \mathbf{w}_k^T \mathbf{x}_d$ will not take on large values, which might cause numerical problems.

After sampling $\omega_k = \begin{bmatrix} \mathbf{v}_k \\ \mathbf{w}_k \end{bmatrix}$ from p_ω, we look for two cornerpoints $\mathbf{q}_1, \mathbf{q}_2$ of the space $X_c \times X_d$. For every dimension i, the i-th element of corner points $\mathbf{q}_1, \mathbf{q}_2$ is determined by

$$q_{1i} = \begin{cases} l_i, & \omega_{ki} \geq 0, \\ u_i, & \omega_{ki} < 0, \end{cases} \tag{1}$$

$$q_{2i} = \begin{cases} u_i, & \omega_{ki} \geq 0, \\ l_i, & \omega_{ki} < 0. \end{cases} \tag{2}$$

Here, l_i and u_i are the lower and upper bounds of the i-th variable respectively, so this gives

$$\omega_k^T \mathbf{q}_1 \leq \mathbf{v}_k^T \mathbf{x}_c + \mathbf{w}_k^T \mathbf{x}_d \leq \omega_k^T \mathbf{q}_2 \;\forall\, \mathbf{x}_c \in X_c, \mathbf{x}_d \in X_d. \tag{3}$$

Now we calculate the distance from the hyperplane generated by ω_k to these corner points, which can be done with the inner product:

$$\beta_1 = \omega_k^T \mathbf{q}_1,\ \beta_2 = \omega_k^T \mathbf{q}_2. \tag{4}$$

By the way β_1 and β_2 are constructed and because $l_i < u_i$, we now have $\beta_1 < \beta_2$. We choose p_b equal to the uniform distribution over $[-\beta_2, -\beta_1]$.

Next we prove that this choice of p_b prevents the hyperplane $z_k(\mathbf{x}_c, \mathbf{x}_d) = 0$ from being completely outside the set $X_c \times X_d$.

Theorem 1. *Let $\omega_k = \begin{bmatrix} \mathbf{v}_k \\ \mathbf{w}_k \end{bmatrix}$ be sampled from any continuous probability distribution p_ω and let b_k be sampled from the uniform distribution over $[-\beta_2, -\beta_1]$, with β_1, β_2 as in* (4)*. Let $z_k(\mathbf{x}_c, \mathbf{x}_d) = \mathbf{v}_k^T \mathbf{x}_c + \mathbf{w}_k^T \mathbf{x}_d + b_k$. Then, there exists a $(\mathbf{x}_c, \mathbf{x}_d) \in X_c \times X_d$ such that $z_k(\mathbf{x}_c, \mathbf{x}_d) = 0$.*

Proof. Suppose that $(\mathbf{x}_c, \mathbf{x}_d) \notin X_c \times X_d$ for all $(\mathbf{x}_c, \mathbf{x}_d)$ for which $z_k(\mathbf{x}_c, \mathbf{x}_d) = 0$. Then from (3), at least one of the following inequalities holds:

$$\mathbf{v}_k^T \mathbf{x}_c + \mathbf{w}_k^T \mathbf{x}_d > \omega_k^T \mathbf{q}_2, \tag{5}$$

$$\mathbf{v}_k^T \mathbf{x}_c + \mathbf{w}_k^T \mathbf{x}_d < \omega_k^T \mathbf{q}_1. \tag{6}$$

Because $z_k(\mathbf{x}_c, \mathbf{x}_d) = 0$, we have $b_k = -\mathbf{v}_k^T \mathbf{x}_c - \mathbf{w}_k^T \mathbf{x}_d$. Because b_k is sampled from p_b, from (4) we also have $-\omega_k^T \mathbf{q}_2 \leq b_k \leq -\omega_k^T \mathbf{q}_1$. This gives $\omega_k^T \mathbf{q}_1 \leq \mathbf{v}_k^T \mathbf{x}_c + \mathbf{w}_k^T \mathbf{x}_d \leq \omega_k^T \mathbf{q}_2$, which is in conflict with (5)-(6). By contradiction, there has to exist a $(\mathbf{x}_c, \mathbf{x}_d) \in X_c \times X_d$ with $z_k(\mathbf{x}_c, \mathbf{x}_d) = 0$. □

Algorithm 1 Determining δ_d

Input Domain X_d, current solution $\mathbf{x}_d^*$
Output $\delta_d \in \mathbb{Z}^{d_d}$

```
for i = 1, ..., d_d do
    r_1 ~ Uniform[0, 1]
    r_2 ~ Uniform[0, 1]        ▷ Whether to increase or decrease x_i, the i-th element of x_d^*
    p = 1/(d_c + d_d)
    while r_1 < p do
        if x_i = l_i then x_i ← x_i + 1
        else if x_i = u_i then x_i ← x_i − 1
        else
            if r_2 < 0.5 then x_i ← x_i + 1
            else x_i ← x_i − 1
        r_1 ← 2r_1
```

Details on the exploration step for integer variables

This section gives more details on the last step of the MV-RSM algorithm, the exploration step. For the integer variables $\mathbf{x}_d^*$, the exploration step consists of determining a random perturbation $\delta_d \in \mathbb{Z}^{d_d}$ that is added to the solution. Our approach is similar to the one in (Bliek, Verwer, and de Weerdt 2019, Sec. 3.4), except that we allow perturbations that are larger than 1. We determine δ_d according to Algorithm 1.

For the continuous variables, we use the procedure from (Bliek et al. 2018), adding a random variable $\delta_c \in$

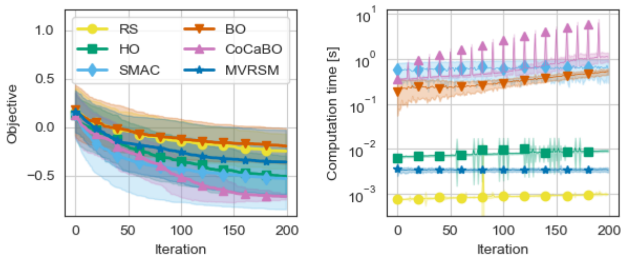

Figure 1: Results on the func3C (Ru et al. 2019, Sec. 5.1) benchmark (3 categorical, 2 continuous), averaged over 100 runs. The compared methods are random search (RS), HyperOpt (HO), SMAC, Bayesian optimisation (BO), CoCaBO and MVRSM.

$\mathbb{R}^{d_c}$ to $\mathbf{x}_c^*$. For each continuous variable $\mathbf{x}_c[i]$, δ_c is zero-mean normally distributed with a standard deviation of $0.1|X_c[i]|/\sqrt{d_c + d_d}$. The exploration step for both integer and continuous variables is done in such a way that the solution stays within the bounds X_c, X_d.

Additional experiments on synthetic benchmark functions

In this section we show the results on some additional synthetic benchmarks with lower dimensions.

Func3C This benchmark was taken from (Ru et al. 2019, Sec. 5.1). It has 3 categorical and 2 continuous variables.

Figure 1 shows the results of 200 iterations averaged over 100 runs. We have managed to reproduce the results from (Ru et al. 2019, Fig. 6(b)) for both HO (also called TPE) and CoCaBO. Our result of SMAC is better here due to not using the default setting. As this benchmark has categorical variables and was one of CoCaBO's benchmarks, we expect CoCaBO to perform best, which it does, though it uses more computation time than the other methods.

Rosenbrock10 The Rosenbrock function[1] is a standard benchmark in continuous optimisation that can be scaled to any dimension. For any dimension, the function has its global minimum in the point $(1, 1, 1, \dots, 1)$, where it achieves the value 0. This benchmark has a dimension of 10, but 3 of the variables were adapted to integers in $X_d = \{-2, -1, 0, 1, 2\}^3$. The 7 remaining continuous variables were limited to $X_c = [-2, 2]^7$. The function was scaled with a factor $1/300$, and uniform noise in $[0, 10^{-6}]$ was added to every function evaluation. This problem is of the same scale as the problem of gradient boosting hyperparameter tuning (Daxberger et al. 2019, Sec. 4(a)).

Figure 2 shows the results of 100 iterations averaged over 100 runs. Surprisingly, BO has the best performance, though it is much slower than MVRSM. This method is typically used on continuous problems and widely assumed to be inadequate for discrete or mixed problems. Here, we have experimentally shown that this is a false assumption. MVRSM and CoCaBO get similar results as BO on this problem, with MVRSM being the most efficient.

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

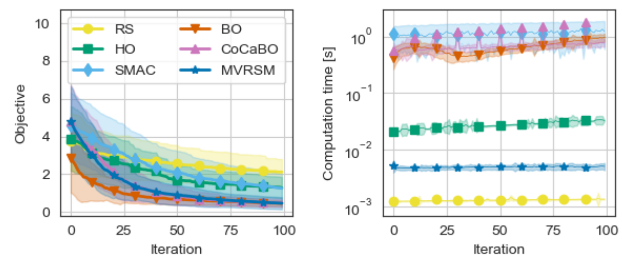

Figure 2: Results on the Rosenbrock10 benchmark (3 integer, 7 continuous), averaged over 100 runs. This problem is of a similar scale as gradient boosting hyperparameter tuning (Daxberger et al. 2019, Sec. 4(a)).

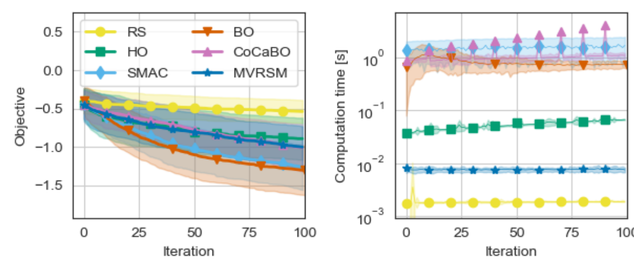

Figure 3: Results on 8 randomly generated MiVaBO synthetic benchmarks (Daxberger et al. 2019, Appendix C.1, Gaussian weights variant) (8 integer, 8 continuous), averaged over 16 runs and over the 8 different benchmarks.

MiVaBO synthetic function We also compare with one of the randomly generated synthetic test functions from (Daxberger et al. 2019, Appendix C.1, Gaussian weights variant) . This problem has 16 variables of which 8 integer and 8 continuous. No bounds were reported so we set them to $X_d = \{0, 1, 2, 3\}^8$ for the integer variables and $X_c = [0, 3]^8$ for the continuous variables. We generated 8 of these random functions and ran all algorithms 16 times on each of them for 100 iterations.

Figure 3 shows the average over all 128 runs. Again, the standard BO algorithm performs best, which is a result that was not concluded in (Daxberger et al. 2019).

Reproducibility checklist

In this section, we clarify some answers given in the reproducibility checklist.

Some benchmark functions, such as the MiVaBO synthetic function, were generated randomly. In this case, the seed was fixed for each run, and the seed numbers are given in the code repository. The algorithms themselves also relied on randomness, but no seeds were fixed for this source of randomness. Instead, we relied on repeating the same experiment multiple times.

The exact versions of the software libraries are given in the relevant poetry files.

The following settings and hyperparameters were tried in the development of the paper, but not chosen in the end. We tried to convert categorical variables to integers for HyperOpt, and although this was faster, it gave worse results than keeping categorical variables as categorical. For MVRSM, during the development of the paper we have also tried a fixed number of mixed z-functions, which gave good results on some problems but not on all of them, and we have tried to normalize the objective function, which did not lead to significant improvements.

References

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* .

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

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* .