COMBO: Combinatorial Bayesian Optimization using Graph Representations

Changyong Oh [1] **Jakub M. Tomczak** [2] **Efstratios Gavves** [1] **Max Welling** [1 2]

Abstract

This paper focuses on Bayesian Optimization – typically considered with continuous inputs– for discrete search input spaces, including integer, categorical or graph structured input variables. In Gaussian process-based Bayesian Optimization a problem arises, as it is not straightforward to define a proper kernel on discrete input structures, where no natural notion of smoothness or similarity could be provided. We propose COMBO, a method that represents values of discrete variables as vertices of a graph and then use the diffusion kernel on that graph. As the graph size explodes with the number of categorical variables and categories, we propose the graph Cartesian product to decompose the graph into smaller sub-graphs, enabling kernel computation in linear time with respect to the number of input variables. Moreover, in our formulation we learn a scale parameter per subgraph. In empirical studies on four discrete optimization problems we demonstrate that our method is on par or outperforms the state-of-the-art in discrete Bayesian optimization.

1. Introduction

While most of the literature is interested in optimization of mathematically well-defined functions in continuous input spaces, a plethora of problems is concerned with finding optima of *black-box* functions, often involving discrete (categorical or ordinal) variables (Jones et al., 1998). Examples of such black-box functions include optimizing hyperparameters for machine learning algorithms (Snoek et al., 2012), finding optimal pipelines for engineering systems (Lam et al., 2018) or optimizing the architecture of a deep neural network (Liu et al., 2018). What distinguishes black-box functions from conventional function optimization is the following: *(i)* black-box functions cannot be defined by a closed-form mathematical expression, such that computing gradients with respect to a loss is not possible, *(ii)* they are expensive to evaluate, and *(iii)* their evaluations are noisy. Because of these properties, applying the popular gradient-based or reinforcement learning optimization methods is challenging (Wilson et al., 2014). Especially for black box functions with expensive evaluation cost, Bayesian Optimization (BO) is rapidly gaining popularity (Shahriari et al., 2016). Interestingly, Bayesian Optimization has mostly been visited in the context of continuous input spaces (Močkus, 1975). Many black box problems, however, live in discrete, combinatorial input spaces, such as scheduling problems, maximum flow problems, shortest path or other graph-related problems (Syslo et al., 2006).

Using classical BO for searching discrete spaces is unsuitable (Garrido-Merchán & Hernández-Lobato, 2018). The most accurate Bayesian Optimization algorithms employ Gaussian process (GP) models for surrogate models (Osborne et al., 2009). The problem with this modelling choice, however, is that it is not straightforward how to define a semi-positive definite kernel for the Gaussian process on discrete inputs. For one, rounding values causes a discrepancy between what the acquisition function of the BO recommends as a next evaluation point, and the actual point evaluated in the end (Garrido-Merchán & Hernández-Lobato, 2018). More importantly rounding leads to flat values for the objective function in between the discretized inputs. As these flat values are obtained after computing the GP, the flatness is ignored when computing the covariances at the next iteration (Garrido-Merchán & Hernández-Lobato, 2018). One can also define a convex relaxation of second-order pair relationships on the graph (Baptista & Poloczek, 2018). In that case, however, important higher order interactions are ignored, potentially underestimating the nonlinear complex dependency between variables including covariances.

To this end, in this work we propose a novel algorithm, which we coin COMBO for combinatorial Bayesian Optimization using graph representations. COMBO is specifically designed for efficient and large-scale Bayesian Optimization in discrete and combinatorial input spaces. Inspired by spectral graph theory (Chung, 1996), we propose to use diffusion kernels (Kondor & Lafferty, 2002; Smola & Kondor, 2003) for defining the surrogate model in our GP. As the size of the problem in discrete spaces increases

[1]QUVA Lab, Institute of Informatics, University of Amsterdam, Amsterdam, the Netherlands [2]Qualcomm AI Research, Qualcomm Technologies Netherlands B.V. Correspondence to: ChangYong Oh <C.Oh@uva.nl>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data

exponentially with respect to the number of categorical variables, and the average number of categories per categorical variable, computing the diffusion kernel soon becomes intractable. To this end, we show that if a graph can be expressed as the graph Cartesian product of smaller subgraphs, it admits a solution in linear time, thus, allowing to scale up to larger and more practical problems. Interestingly, we show that one can easily adapt the diffusion kernel to account for individual tuning parameters per categorical variable when computing the covariances between discrete variables, which yields more flexible kernels.

2. Method

2.1. Bayesian Optimization

Bayesian optimization aims at finding the global optimum of a black-box function f over a search space $\mathcal{X}$, namely

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}). \tag{1}$$

The general pipeline of Bayesian optimization is as follows. At each round, in the absence of any other information regarding the nature of $f(\mathbf{x})$, a surrogate model attempts to approximate the behavior of $f(\mathbf{x})$ based on the so far observed points $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, where $y_i = f(\mathbf{x}_i)$. The surrogate function is then followed by an acquisition function that suggests the next most interesting point $\mathbf{x}_{i+1}$ that should be evaluated. The pair $(\mathbf{x}_i, y_i)$ is added to the training dataset, $\mathcal{D} = \mathcal{D} \cup (\mathbf{x}_i, y_i)$, the Gaussian process is retrained and the process repeats until the optimization budget is depleted.

2.2. Bayesian Optimization on Discrete Structures

Search space as a graph To this end, we draw inspiration from spectral graph theory (Smola & Kondor, 2003). We represent the search space as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex in $\mathcal{V}$ is a different configuration of the input, and an edge in $\mathcal{E}$ determines whether two such configurations are considered to be similar or no. For instance, for m categorical variables with each d possible values there exist d^m nodes. The objective function is then a function defined on this graph, *i.e.*, the *graph signal* $f : \mathcal{V} \to \mathbb{R}$. Having defined a graph signal as a function, we can use Fourier analysis on graphs (Ortega et al., 2018) to obtain an approximation to the graph signal f. For this purpose, we need the graph Laplacian, $L(\mathcal{G}) = \mathbf{D}_{\mathcal{G}} - \mathbf{A}_{\mathcal{G}}$ where $\mathbf{A}_{\mathcal{G}}$ is the adjacency matrix and $\mathbf{D}_{\mathcal{G}}$ is the degree matrix. We recover the frequencies and the Fourier bases of our decomposition as the eigenvalues, $\{\lambda_i\}_{i=1,\dots,n}$, and the eigenvectors, $\{u_i\}_{i=1,\dots,n}$, of the laplacian $L(\mathcal{G})$, respectively. Then, the graph signal f on $\mathcal{G}$ is equivalent to the linear combination of the Fourier basis-vectors, namely

$$f([p]) = \sum_{i=1}^{n} w_i u_i([p]), \tag{2}$$

where w_i is a coefficient, $u_i([p])$ is the p-the entry of the vector u_i and n is the number of non-zero eigenvalues. An approximation of $f([p])$ could be obtained by capping eigenvalues and summing up over only a subset of them.

Kernels on graphs Following (Smola & Kondor, 2003), we can construct a kernel on a graph by smoothly regularizing frequencies. The idea is to penalize frequencies λ_i according to a regularization operator for graph signals. In our method we rely on $r(\lambda) = \exp(\beta\ \lambda))$ for a regularization operator. We can then use a kernel derived from the regularization operator to model a smooth function on the graph using a Gaussian Process. The smoothness of the kernel can be controlled by the regularization operator and the weights on the edges of the graph. Formally, the corresponding kernel is defined as follows (see Corollary 1 in (Smola & Kondor, 2003)):

$$k([p],[q]) = \sum_{i=1}^{n} \frac{1}{r(\lambda_i)} u_i([p]) u_i([q]), \tag{3}$$

which regularizes high frequencies because the function $1/r(\lambda_i)$ obtains lower values for higher λ_i.

Interestingly, taking the diffusion process regularization operator $r(\lambda_i) = \exp(\beta\ \lambda_i)$, where $\beta > 0$, we obtain a discrete version of the exponential kernel, the *diffusion kernel* (Kondor & Lafferty, 2002; Smola & Kondor, 2003):

$$\mathbf{K} = \exp\big(-\beta\ L(\mathcal{G})\big), \tag{4}$$

where $\mathbf{K}$ is the kernel matrix. Computing the kernel $\mathbf{K}$ requires calculating the matrix exponentiation defined by the limit $\exp(\mathbf{B}) = \lim_{n\to\infty}(\mathbf{I} - \mathbf{B}/n)^n$. As the kernel on a graph in eq. 4 is rather computationally troublesome due to the matrix exponentiation, we simplify the calculation by using the form in (3). For this purpose we need to calculate the eigendecomposition of the Laplacian matrix, $L(\mathcal{G}) = \mathbf{U}\Lambda\mathbf{U}^T$, where $\mathbf{U}$ is a matrix with eigenvectors in columns, and Λ is a diagonal matrix with eigenvalues on the diagonal. Then, the kernel matrix could be expressed as follows:

$$\mathbf{K} = \mathbf{U} r^{-1}(\Lambda) \mathbf{U}^T, \tag{5}$$

where $r^{-1}(\Lambda) = \mathrm{diag}\Big(\frac{1}{r(\lambda_i)}\Big)$.

2.3. Cartesian Product Graph Kernels

In the current formulation, the Fourier basis and the frequencies are given by an eigendecomposition of a $|\mathcal{V}| \times |\mathcal{V}|$ matrix, with $|\mathcal{V}|$ being the number of nodes in the graph. Unfortunately, this approach does not scale up to graphs with a large number of vertices due to the cubic computational complexity of the eigendecomposition. However, an arbitrary graph can be uniquely decomposed into the Cartesian product of small graphs up to graph isomorphisms

(Hammack et al., 2011). To improve scalability, therefore, we approximate a large graph $\mathcal{G}$ by the graph Cartesian product of smaller graphs $\{\mathcal{G}_i\}_i$.

Importantly, the Laplacian of the $\mathcal{G}_1 \square \mathcal{G}_2$ can be expressed algebraically using the Kronecker product $\otimes$ and the Kronecker sum $\oplus$ as follows: (Hammack et al., 2011):

$$\begin{aligned} L(\mathcal{G}_1 \square \mathcal{G}_2) &= L(\mathcal{G}_1) \oplus L(\mathcal{G}_2) \\ &= L(\mathcal{G}_1) \otimes \mathbf{I}_1 + \mathbf{I}_2 \otimes L(\mathcal{G}_2), \end{aligned} \tag{6}$$

where $\mathbf{I}$ denotes the identity matrix.

In the case of the diffusion kernel for m discrete (categorical or ordinal) variables, we can take advantage of the properties of the graph Cartesian product in eq. (6) and the matrix exponentiation to obtain:

$$\mathbf{K} = \bigotimes_{i=1}^{m} \exp\big(-\beta\, L(\mathcal{G}_i)\big). \tag{7}$$

Then, we can compute the kernel matrix by calculating the Kronecker product of individual kernels. Using eq. (5) we can compute the eigendecomposition of the individual Laplacians to obtain the kernel for the i-th subgraph.

In general, the application of the graph Cartesian product enables us to reduce kernel computations from $O(\prod_{i=1}^{m} |\mathcal{V}_i|)$ to $O(\sum_{i=1}^{m} |\mathcal{V}_i|)$. The advantage is that for graphs that can be decomposed we can obtain much higher computational efficiency. A potential problem may arise, however, if the assumed independence between subgraphs $\mathcal{G}_1$ and $\mathcal{G}_2$ is invalid. Taking the decomposition to the limit by recursion, for example, would lead to each node in the graph being independent, which is clearly undesirable. For the proposed decomposition there is a trade-off between relying on many small subgraphs, and, thus, increasing efficiency, or having fewer subgraphs, and increasing model flexibility in capturing higher-order interactions. Provided that we have reasonable prior knowledge of the search domain, which is typically the case for Bayesian optimization (Shahriari et al., 2016), we can scale up to large graph search spaces with reasonable flexibility.

Variable-wise edge scaling Further, we notice that we can make the kernel more flexible by considering an individual scaling factor for each variable instead of a global scaling parameter. Then, the diffusion kernel becomes:

$$\mathbf{K} = \exp\big(\bigoplus_{i=1}^{m} -\beta_i\, L(\mathcal{G}_i)\big), \tag{8}$$

where $\beta_i > 0$ for $i = 1, \ldots, m$. Since the diffusion kernel is a discrete version of the exponential kernel, the application of the individual β_i for each variable is equivalent to the automatic relevance determination (ARD) kernel (MacKay, 1994; Neal, 1995). Hence, we can determine which variables (subgraphs) are more relevant than the others. We refer to this kernel as the *ARD diffusion kernel*.

Algorithm 1 COMBO: Combinatorial Bayesian Optimization with kernels on a graph

Input: n **categorical variables**
Set a search space: # See Sect. 2.2 and 2.3
  ▷ Specify the search space as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
  ▷ Calculate Fourier basis-vectors and frequencies of the given graph $\{(\lambda_i, u_i)\}$.
Set $\mathcal{D} = \emptyset$.
Initialize $\mathcal{C}$ that is a set of k randomly selected vertices.
repeat
  Estimate β_i using samples from $p(\beta_i|\mathcal{D})$ using slice sampling.
  Evaluate acquisition function on vertices in $\mathcal{C}$, *i.e.*, the predictive mean and the predictive standard deviation at a vertex $[v] \in \mathcal{C}$: $\mu([v]), \sigma([v])$, using the ARD diffusion kernel in (8).
  Pick the best performing vertex from $\mathcal{C}$: $[v^*] = \arg\max_{[v]} a\big(\mu([v]), \sigma([v])\big)$.
  Evaluate the objective at $[v^*]$, $f([v^*])$, and $\mathcal{D} = \mathcal{D} \cup \{[v^*], f([v^*])\}$.
  Determine $\mathcal{C}$ of k candidates by running a greedy search from $[v^*]$.
until stopping criterion

2.4. COMBO Algorithm

Having defined an appropriate kernel on large-scale graphs, we can employ large-scale Gaussian processes. The Gaussian process on the graph provides the predictive mean and the predictive standard deviation for every graph vertex. We use these two quantities to maximize the acquisition function $a(\cdot, \cdot)$ and determine the next vertex to be evaluated, as in standard Bayesian Optimization.

In this paper, in order to optimize the acquisition function, we rely on greedy optimization on a graph. In our framework we can use any existing acquisition function like GP-UBC or the Expected Improvement (EI) (Rasmussen & Williams, 2006). We opt for EI. Determining β's is crucial for obtaining a flexible kernel function. For each β_i we use a uniform prior over $[0, 2]$. Then, we sample from the posterior of β's, $p(\boldsymbol{\beta}|\mathcal{D})$, using slice sampling (Neal, 2003) within Gibbs sampling. All these steps are presented in Algorithm in 1. We refer to this algorithm as the Combinatorial Bayesian Optimization (COMBO) with graph representation.

3. Experiments

We evaluate our approach with the following experiments. As an illustration we conduct four experiments: *(i)* dis-

cretized Branin benchmark (Laguna & Martí, 2005), *(ii)* a contamination control of a food supply chain with 25 stages (Baptista & Poloczek, 2018), *(iii)* a sparsification of Ising models with 24 discrete variables (Baptista & Poloczek, 2018), and *(iv)* a pest control system with 25 categorical variables, each variable taking 5 possible values. The detailed description of these problems could be found in the Supplementary Material. We compare COMBO with the following approaches: Simulated Annealing (SA) (Spears, 1993), PS: sequential particle sampling for binary problems (Schäfer, 2013), SMAC (Hutter et al., 2011): an approach similar to EI with a local search for a candidate with high expected improvement using a random forest model, Oblivious Local Search (OLS) (Khanna et al., 1998): a local search with the Hamming distance starting at a randomly chosen point, Random Search (RS), Bayesian Optimization of Combinatorial Structures (BOCS-SDP) (Baptista & Poloczek, 2018): a combination of the sparse Bayesian linear regression with second-order relationships and the semi-definite programming (SDP) as an acquisition function, Expected improvement (EI) (Jones et al., 1998): a Gaussian Process with categorical variables represented by one-hot encoding.

3.1. Results and Discussion

Results for the proposed approach and other methods are prestend in Tables 1, 2, 3, and 4. More detailed results are provided in the Supplementary Material.

Table 1. Results on the discretized Branin benchmark. We present an average with a standard error calculated over 25 runs.

METHOD	
SA	0.71±0.17
SMAC	0.84±0.11
RS	0.96±0.08
TPE	1.06±0.14
COMBO	**0.40**±0.00

Table 2. Results on the contamination control benchmark. We present an average with a standard error calculated over 25 runs.

METHOD	$\lambda = 0$	$\lambda = 0.0001$	$\lambda = 0.01$
SA	21.49±0.04	21.52±0.04	21.68±0.03
PS	21.97±0.06	21.91±0.06	22.08±0.06
SMAC	21.67±0.04	21.76±0.05	21.87±0.05
RS	21.90±0.05	21.92±0.04	22.12±0.03
OLS	21.47±0.04	21.42±0.05	21.61±0.04
BOCS-SDP	**21.28**±0.03	21.31±0.03	**21.44**±0.03
EI	21.33±0.02	21.34±0.03	21.50±0.03
COMBO	**21.26**±0.03	**21.28**±0.03	**21.43**±0.03

We notice that on three out of four cases COMBO performed the best or on par with the SOTA methods. Importantly, our approach performed outstanding in the case of multi-category discrete variable (Branin and Pest) showing its great potential. Moreover, we notice that COMBO converges fast and is always among three fastest converging methods (see figures in the Supplementary Material).

Table 3. Results on the Ising sparsification benchmark. We present an average with a standard error calculated over 25 runs.

METHOD	$\lambda = 0$	$\lambda = 0.0001$	$\lambda = 0.01$
SA	0.12±0.05	0.06±0.03	0.29±0.05
PS	1.30±0.23	0.94±0.18	1.30±0.21
SMAC	0.33±0.07	0.40±0.07	0.51±0.06
RS	0.80±0.14	0.74±0.14	1.02±0.15
OLS	0.40±0.17	0.36±0.10	0.34±0.06
BOCS-SDP	**0.01**±0.02	**0.02**±0.04	**0.21**±0.05
EI	0.13±0.04	0.14±0.05	0.30±0.04
COMBO	0.12±0.04	0.11±0.03	0.31±0.05

Table 4. Results on the pest control benchmark. We present an average with a standard error calculated over 25 runs.

METHOD	
SA	**13.04**±0.13
SMAC	14.66±0.08
RS	15.79±0.07
TPE	15.05±0.05
COMBO	**12.99**±0.11

4. Conclusion

In this work we propose COMBO, an algorithm tailored for Bayesian Optimization for discrete and combinatorial inputs input search spaces. To the best of our knowledge, COMBO is the first Bayesian Optimization algorithm using Gaussian Processes as a surrogate model suitable for high-dimensional combinatorial inputs. To efficiently tackle the exponentially increasing complexity of discrete search spaces, we rest upon the following ideas: *(i)* we represent the search space as a graph with configurations in vertices and edges corresponding to similarity among vertices, *(ii)* we propose a flexible ARD diffusion kernel on graphs. *(iii)* we rely on graph Cartesian products for graph decomposition, allowing for calculating the kernel in a linear time with respect to the number of vertices and last, *(iv)* we use greedy local search for selecting next points for evaluation. We evaluate the proposed algorithm on four discrete optimization problems –binary and multi-category ones–. On the binary problems COMBO performs on par or a better than the state-of-the-art, whereas on the multi-category discrete optimization problems COMBO outperforms all competitors.

References

Baptista, R. and Poloczek, M. Bayesian Optimization of Combinatorial Structures. In *International Conference on Machine Learning*, pp. 462–471, 2018.

Chung, F. R. Spectral graph theory (cbms regional conference series in mathematics, no. 92). 1996.

Garrido-Merchán, E. C. and Hernández-Lobato, D. Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes. *arXiv preprint arXiv:1805.03463*, 2018.

Hammack, R., Imrich, W., and Klavžar, S. *Handbook of product graphs*. CRC press, 2011.

Hu, Y., Hu, J., Xu, Y., Wang, F., and Cao, R. Z. Contamination control in food supply chain. In *Simulation Conference (WSC), Proceedings of the 2010 Winter*, pp. 2678–2681. IEEE, 2010.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pp. 507–523. Springer, 2011.

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

Khanna, S., Motwani, R., Sudan, M., and Vazirani, U. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1998.

Kondor, R. I. and Lafferty, J. Diffusion kernels on graphs and other discrete structures. In *International Conference on Machine Learning*, volume 2002, pp. 315–322, 2002.

Laguna, M. and Martí, R. Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *Journal of Global Optimization*, 33(2): 235–255, 2005.

Lam, R., Poloczek, M., Frazier, P., and Willcox, K. E. Advances in Bayesian Optimization with Applications in Aerospace Engineering. In *2018 AIAA Non-Deterministic Approaches Conference*, pp. 1656, 2018.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

MacKay, D. J. Bayesian nonlinear modeling for the prediction competition. *ASHRAE transactions*, 100(2):1053–1062, 1994.

Močkus, J. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pp. 400–404. Springer, 1975.

Neal, R. M. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

Neal, R. M. Slice sampling. *Annals of Statistics*, pp. 705–741, 2003.

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., and Vandergheynst, P. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

Osborne, M. A., Garnett, R., and Roberts, S. J. Gaussian processes for global optimization. In *3rd International Conference on Learning and Intelligent Optimization (LION3)*, pp. 1–15, 2009.

Rasmussen, C. E. and Williams, C. K. *Gaussian processes for machine learning*. the MIT Press, 2006.

Schäfer, C. Particle algorithms for optimization on binary spaces. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 23(1):8, 2013.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. Taking the human out of the loop: A review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Smola, A. J. and Kondor, R. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pp. 144–158. Springer, 2003.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Spears, W. M. Simulated annealing for hard satisfiability problems. In *Cliques, Coloring, and Satisfiability*, pp. 533–558. Citeseer, 1993.

Syslo, M., Deo, N., and Kowalik, J. S. *Discrete optimization algorithms: with Pascal programs*. Dover Publications, 2006.

Wilson, A., Fern, A., and Tadepalli, P. Using trajectory data to improve Bayesian optimization for reinforcement learning. *The Journal of Machine Learning Research*, 15 (1):253–282, 2014.

Supplementary Material
Combinatorial Bayesian Optimization using Graph Representation

1. The graph Cartesian product

The resulting graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from the Cartesian product of two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, namely, $\mathcal{G} = \mathcal{G}_1 \,\square\, \mathcal{G}_2$, is defined as follows:

$$\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2, \tag{9}$$

$$\begin{aligned}\big((v_1, v_2), (v_1', v_2')\big) \in \mathcal{E} \Leftrightarrow \\ v_1 = v_1' \in \mathcal{V}_1 \wedge (v_2, v_2') \in \mathcal{E}_2 \vee \\ (v_1, v_1') \in \mathcal{E}_1 \wedge v_2 = v_2' \in \mathcal{V}_2.\end{aligned} \tag{10}$$

To scale up computations for large graphs, we use the following property of the graph Cartesian product (Hammack et al., 2011). For the eigensystems $\{(\lambda_i^{(1)}, u_i^{(1)})\}$ and $\{(\lambda_j^{(2)}, u_j^{(2)})\}$ of $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, the eigensystem of $\mathcal{G}_1 \square \mathcal{G}_2$ is given by $\{(\lambda_i^{(1)} + \lambda_j^{(2)}, u_i^{(1)} \otimes u_j^{(2)})\}$, where $\otimes$ denotes the Kronecker product.

2. COMBO algorithm: Additional details

Optimization of the acquisition function Since the search space is represented by a graph, we need to evaluate graph nodes by the acquisition function. In this paper, in order to optimize the acquisition function, we rely on greedy optimization on a graph. Namely, at a given vertex, we compare values of the acquisition function at all neighboring vertices and move to the vertex with the highest acquisition function value, if it is different than the given vertex. Having evaluated the objective function at the next vertex, we repeat the procedure until a stopping criterion is met. We start by randomly picking a starting vertex.

To further increase exploration capabilities of the optimization procedure, we keep a set of k candidates[1], denoted by $\mathcal{C}$, for which we run greedy optimization in parallel. The candidates consists of vertices that are closest to the current best vertex.

In our framework we can use any existing acquisition function like GP-UBC or the Expected Improvement (EI) (Rasmussen & Williams, 2006). We opt for EI.

Determination of the scaling factors Determining β's is crucial for obtaining a flexible kernel function. For each β_i we use a uniform prior over $[0, 2]$. Then, we sample from the posterior of β's, $p(\boldsymbol{\beta}|\mathcal{D})$, using slice sampling (Neal, 2003) within Gibbs sampling. At each step of the Bayesian Optimization procedure we use a couple of samples[2] to determine β_i. For each β_i the sampling procedure is the following:

1. Set $t = 0$ and choose a starting $\beta_i^{(t)}$ for which the probability is non-zero.
2. Sample a value q uniformly from $[0, p(\beta_i^{(0)}|\mathcal{D}, \beta_{-i}^{(0)})]$.
3. Draw a new value $\beta_i^{(t+1)}$ uniformly from regions for which $p(\beta_i^{(t)}|\mathcal{D}, \beta_{-i}^{(t)}) > q$.
4. Repeat from 2 using $\beta_i^{(t+1)}$.

[1] In the experiments we use $k = 20$ points.

[2] In the experiments we use 10 samples.

3. Problem descriptions

3.1. Discretized Branin

The Branin benchmark is an optimization problem of a non-linear function over a 2D search space (Jones et al., 1998). We discretize the search space, namely, we consider a grid of points. The problem becomes a discrete optimization problems with ordinal variables. Since the Branin cost function is a smooth function, the problem preserves this property. That said, examining the behavior of COMBO and other competing methods on this simplified problem allows for deriving insights regarding Bayesian Optimization methods on ordinal inputs. We set the budget to 100 evaluations.

3.2. Contamination Control

The contamination control in food supply chain is a binary optimization problem (Hu et al., 2010). The problem is about minimizing the contamination of food where at each stage a prevention effort can be made to decrease a possible contamination. Applying the prevention effort results in an additional cost c_i. However, if the food chain is contaminated at stage i, the contamination spreads at rate α_i. The contamination at the i-th stage is represented by a random variable Γ_i. A random variable z_i denotes a fraction of contaminated food at the i-th stage, and it could be expressed in an recursive manner, namely, $z_i = \alpha_i(1 - x_i)(1 - z_{i-1}) + (1 - \Gamma_i x_i)z_{i-1}$, where $x_i \in \{0, 1\}$ is the decision variable representing the preventing effort at stage i. Hence, the optimization problem is to make a decision at each stage whether the prevention effort should be applied so that to minimize the general cost while also ensuring that the upper limit of contamination is u_i with probability at least $1 - \varepsilon$. The initial contamination and other random variables follow beta distributions that results in the following objective function: $\mathcal{L}(x) = \sum_{i=1}^{d} \left[c_i x_i + \frac{\rho}{T} \sum_{k=1}^{T} 1_{\{z_k > u_i\}} \right] + \lambda \|x\|_1$, where λ is a regularization coefficient, ρ is a penalty coefficient (we use $\rho = 1$) and we set $T = 100$. Following (Baptista & Poloczek, 2018), we assume $u_i = 0.1$, $\varepsilon = 0.05$, and $\lambda \in \{0, 10^{-4}, 10^{-2}\}$. We set the budget to 270 evaluations.

3.3. Sparsification of Ising models

This optimization problem is about approximating a zero-field Ising model expressed by $p(z) = \frac{1}{Z_p} \exp\{z^\top J^p z\}$, where $z \in \{-1, 1\}^n$, $J^p \in \mathbb{R}^{n \times n}$ is an interaction symmetric matrix, and $Z_p = \sum_z \exp\{z^\top J^p z\}$ is the partition function, using a model $q(z)$ with $J^q_{ij} = x_{ij} J^p_{ij}$ where $x_{ij} \in \{0, 1\}$ are the decision variables. The objective function is the regularized Kullback-Leibler divergence between p and q, namely: $\mathcal{L}(x) = D_{KL}(p||q) + \lambda \|x\|_1$, where $\lambda > 0$ is the regularization coefficient. D_{KL} could be calculated analytically (Baptista & Poloczek, 2018). We follow the same setup as presented in (Baptista & Poloczek, 2018), namely, we consider 4×4 grid of spins, and interactions are sampled randomly from a uniform distribution over $[0.05, 5]$. The exhaustive search requires enumerating all 2^{24} configurations of x that is infeasible. We consider $\lambda \in \{0, 10^{-4}, 10^{-2}\}$. We set the budget to 170 evaluations.

4. Pest control

In the chain of locations, pest is spread in one direction, at each pest control point, the pest control officer can choose to use a pesticide from 4 different companies which differ in their price and effectiveness.

For N pest control points, the search space for this problem is 5^N, 4 choices of a pesticide and the choice of not using any of it.

The price and effectiveness reflect following dynamics.

- If you have purchased a pesticide a lot, then in your next purchase of the same pesticide, you will get discounted proportional to the amount you have purchased.
- If you have used a pesticide a lot, then pests will acquire strong tolerance to that specific product, which decrease effectiveness of that pesticide.

Formally, there are four variables: at i-th pest control Z_i is the portion of the product having pest, A_i is the action taken, $C_i^{(l)}$ is the adjusted cost of pesticide of type l, $T_i^{(l)}$ is the beta parameter of the Beta distribution for the effectiveness of pesticide of type l. It starts with initial Z_0 and follows the same evolution as in the contamination control, but after each choice of pesticide type whenever the taken action is to use one out of 4 pesticides or no action. $\{C_i^{(l)}\}_{1,\cdots,4}$ are adjusted

in the manner that the pesticide which has been purchased most often will get a discount for the price. $\{T_i^{(l)}\}_{1,\cdots,4}$ are adjusted in the fashion that the pesticide which has been frequently used in previous control point cannot be as effective as before since the insects have developed tolerance to that.

The portion of the product having pest follows the dynamics below

$$z_i = \alpha_i(1 - x_i)(1 - z_{i-1}) + (1 - \Gamma_i x_i)z_{i-1} \tag{11}$$

when the pesticide is used, the effectiveness x_i of pesticide follows beta distribution with the parameters, which has been adjusted according to the sequence of actions taken in previous control points.

Under this setting, our goal is to minimize the expense for pesticide control and the portion of products having pest while going through the chain of pest control points. The objective is similar to the contamination control problem

$$\mathcal{L}(x) = \sum_{i=1}^{d} \left[c_i x_i + \frac{\rho}{T} \sum_{k=1}^{T} 1_{\{z_k > u_i\}} \right] \tag{12}$$

However, we want to stress out that the dynamics of this problem is far more complex than the one in the contamination control case. First, it has 25 variables and each variable has 5 categories. More importantly, the price and effectiveness of pesticides are dynamically adjusted depending on the previously made choice.

5. Additional results

In the next subsection we present additional results for the four benchmark problems considered in the paper.

5.1. Discretized Branin

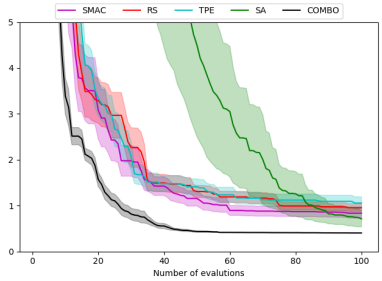

Figure 1. Results for the discretized Branin benchmark.

5.2. Contamination control

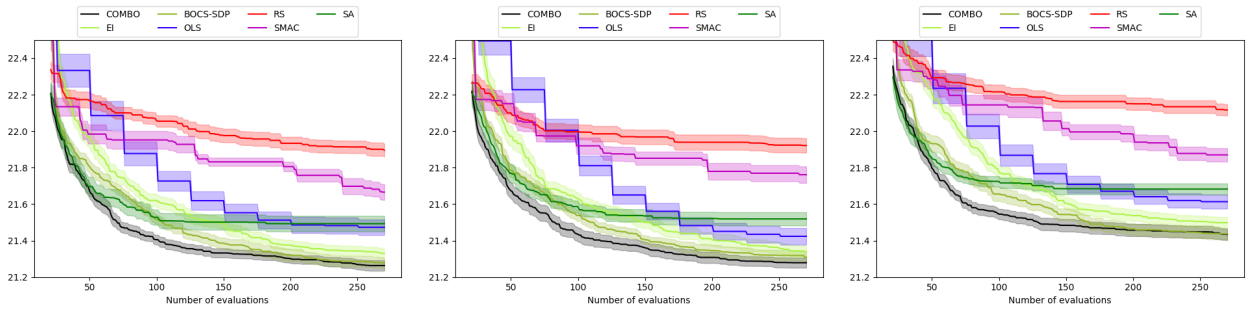

Figure 2. Results for the contamination control: (*left*) $\lambda = 0$, (*middle*) $\lambda = 0.0001$, (*right*) $\lambda = 0.01$.

5.3. Sparsification of Ising models

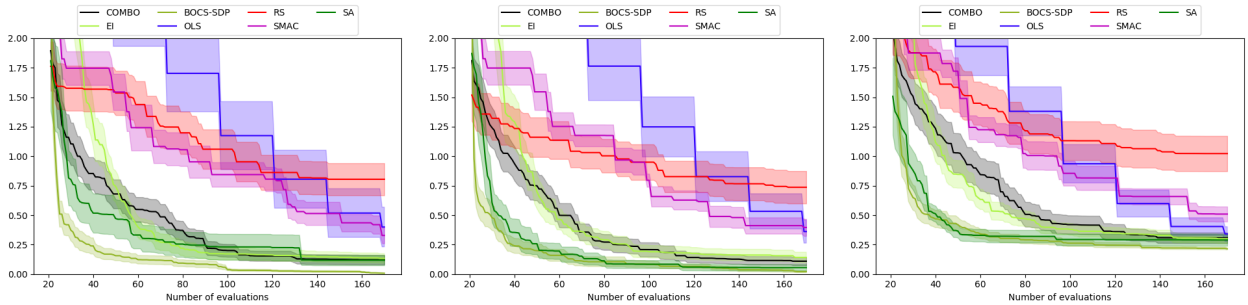

Figure 3. Results for the sparsification of Ising models: (*left*) $\lambda = 0$, (*middle*) $\lambda = 0.0001$, (*right*) $\lambda = 0.01$.