Fourier Representations for Black-Box Optimization over Categorical Variables

Hamid Dadkhahi[1][*], Jesus Rios[2], Karthikeyan Shanmugam[2], Payel Das[2]

[1] Amazon
[2] IBM Research AI

dadkhami@amazon.com, jriosal@us.ibm.com, karthikeyan.shanmugam2@ibm.com, daspa@us.ibm.com

Abstract

Optimization of real-world black-box functions defined over purely categorical variables is an active area of research. In particular, optimization and design of biological sequences with specific functional or structural properties have a profound impact in medicine, materials science, and biotechnology. Standalone search algorithms, such as simulated annealing (SA) and Monte Carlo tree search (MCTS), are typically used for such optimization problems. In order to improve the performance and sample efficiency of such algorithms, we propose to use existing methods in conjunction with a surrogate model for the black-box evaluations over purely categorical variables. To this end, we present two different representations, a group-theoretic Fourier expansion and an abridged one-hot encoded Boolean Fourier expansion. To learn such representations, we consider two different settings to update our surrogate model. First, we utilize an adversarial online regression setting where Fourier characters of each representation are considered as experts and their respective coefficients are updated via an exponential weight update rule each time the black box is evaluated. Second, we consider a Bayesian setting where queries are selected via Thompson sampling and the posterior is updated via a sparse Bayesian regression model (over our proposed representation) with a regularized horseshoe prior. Numerical experiments over synthetic benchmarks as well as real-world RNA sequence optimization and design problems demonstrate the representational power of the proposed methods, which achieve competitive or superior performance compared to state-of-the-art counterparts, while improving the computation cost and/or sample efficiency, substantially.

1 Introduction

A plethora of practical optimization problems involve black-box functions, with no simple analytical closed forms, that can be evaluated at any arbitrary point in the domain. Optimization of such black-box functions poses a unique challenge due to restrictions on the number of possible function evaluations, as evaluating functions of real-world complex processes is often expensive and time consuming. Efficient algorithms for global optimization of expensive black-box functions take past queries into account in order to select the next query to the black-box function more intelligently.

Black-box optimization of real-world functions defined over purely categorical type input variables has not been studied in the literature as extensively as its integer, continuous, or mixed variables counterparts.

Categorical type variables are particularly challenging when compared to integer or continuous variables, as they do not have a natural ordering. However, many real-world functions are defined over categorical variables. One such problem, which is of wide interest, is the design of optimal chemical or biological (protein, RNA, and DNA) molecule sequences, which are constructed using a vocabulary of fixed size, e.g. 4 for DNA/RNA. Designing optimal molecular sequences with novel structures or improved functionalities is of paramount importance in material science, drug and vaccine design, synthetic biology and many other applications, see (Dixon et al. 2010; Ng et al. 2019; Hoshika et al. 2019; Yamagami et al. 2019). Design of optimal sequences is a difficult black-box optimization problem over a combinatorially large search space (Stephens et al. 2015), in which function evaluations often rely on either wet-lab experiments, physics-inspired simulators, or knowledge-based computational algorithms, which are slow and expensive in practice. Another problem of interest is the constrained design problem, e.g. find a sequence given a specific structure (or property), which is inverse of the well-known folding problem discussed in (Dill and MacCallum 2012). This problem is complex due to the strict structural constraints imposed on the sequence. In fact, one of the ways to represent such a complex structural constraint is to constrain the next choice sequentially based on the sequence elements that have been chosen a priori. Therefore, we divide the black box optimization problem into two settings, depending on the constraint set: (i) Generic Black Box Optimization (BBO) problem referring to the unconstrained case and (ii) Design Problem that refers to the case with complex sequential constraints.

Let x_t be the t-th sequence evaluated by the black-box function f. The key question in both settings is the following: Given prior queries $x_1, x_2 \ldots x_t$ and their evaluations $f(x_1) \ldots f(x_t)$, *how to choose the next query x_{t+1}?* This selection must be devised so that over a finite budget of black-box evaluations, one is closest to the minimizer in an expected sense over the acquisition randomness.

In the literature, for design problems with sequential constraints, MCTS (Monte Carlo Tree Search) based search al-

gorithms are often used with black box function evaluations. In the generic BBO problems in the unconstrained scenario, Simulated Annealing (SA) based techniques are typically used as search algorithms. A key *missing* ingredient in the categorical domain is a surrogate model for the black-box evaluations that can interpolate between such evaluations and facilitate *cost-free* approximate evaluations from the surrogate model internally in order to reduce the need for frequently accessing real black-box evaluations, thus leading to improved sample efficiency in both generic and design black-box optimization problems. Due to the lack of efficient interpolators in the categorical domains, existing search algorithms suffer under constrained computational budgets, due to reliance on only real black-box evaluations.

Contributions: We address the above problem in our work. Our main contributions are as follows:

1. We present two representations for modeling real-valued combinatorial functions over categorical variables, which we then use in order to learn a surrogate model for the generic BBO problem and the design problem, with a finite budget on the number of queries. The abridged one-hot encoded Boolean Fourier representation is novel to this work. The use of group-theoretic Fourier representation for modeling functions over categorical variables, and in particular their optimization, is novel to this work.

2. Numerical experiments, over synthetic benchmarks as well as real-world biological (RNA) sequence optimization demonstrate the competitive or superior performance of the proposed methods incorporating our representations over state-of-the-art counterparts, while substantially reducing the computation time. We further evaluate the performance of our algorithms in design problems (inverse folding problem) and demonstrate the superior performance of our methods in terms of sample efficiency over the state-of-the-art counterparts.

2   Related Work

Black-Box Optimization: Hutter et al. (Hutter, Hoos, and Leyton-Brown 2011) suggest a surrogate model based on random forests to address optimization problems over categorical variables. The proposed SMAC algorithm uses a randomized local search under the expected improvement acquisition criterion to obtain candidate points for black-box evaluations. Bergstra et al. (Bergstra et al. 2011) suggest a tree-structured Parzen estimator (TPE) for approximating the surrogate model, and maximizes the expected improvement criterion to find candidate points for evaluation. For optimization problems over Boolean variables, multilinear polynomials (Ricardo Baptista 2018; Dadkhahi et al. 2020) and Walsh functions (Leprêtre et al. 2019; Deshwal, Belakaria, and Doppa 2021) have been used in the literature.

Bayesian Optimization (BO) is a commonly used approach for optimization of black-box functions (Shahriari et al. 2015). However, limited work has addressed incorporation of categorical variables in BO. Early attempts were based on converting the black-box optimization problem over categorical variables to that of continuous variables (Gómez-Bombarelli et al. 2018; Golovin et al. 2017; Garrido-Merchán and Hernández-Lobato 2020). A few BO algorithms have been specifically

designed for black-box functions over combinatorial domains. In particular, the BOCS algorithm (Ricardo Baptista 2018), primarily devised for Boolean functions, employs a sparse monomial representation to model the interactions among different variables, and uses a sparse Bayesian linear regression method to learn the model coefficients. The COMBO algorithm of (Oh et al. 2019) uses Graph Fourier Transform over a combinatorial graph, constructed via graph Cartesian product of variable subgraphs, to gauge the smoothness of the black-box function. However, both BOCS and COMBO are hindered by associated high computation costs, which grow polynomially with both the number of variables and the number of function evaluations. More recently, a computationally efficient black-box optimization algorithm (COMEX) (Dadkhahi et al. 2020) was introduced to address the computational impediments of its Bayesian counterparts. COMEX adopts a Boolean Fourier representation as its surrogate model, which is updated via an exponential weight update rule. Nevertheless, COMEX is limited to functions over the Boolean hypercube.

We generalize COMEX to handle functions over categorical variables by proposing two representations for modeling functions over categorical variables: an abridged one-hot encoded Boolean Fourier representation and Fourier representation on finite Abelian groups. The utilization of the latter representation as a surrogate model in combinatorial optimization algorithms is novel to this work. Factorizations based on (vanilla) one-hot encoding has been previously (albeit briefly) suggested in (Ricardo Baptista 2018) to enable black-box optimization algorithms designed for Boolean variables to address problems over categorical variables. Different from (Ricardo Baptista 2018), we show that we can significantly reduce the number of additional variables introduced upon one-hot encoding, and that such a reduced representation is in fact complete and unique. We incorporate the latter representation to extend a modified version of BOCS to problems over categorical variables, efficiently.

Sequence Design: For design problems, we focus on the RNA sequence design problem (RNA inverse folding). The goal is to find an RNA sequence consistent with a given secondary structure, as the functional state of the RNA molecule is determined by the latter structure (Hofacker et al. 1994). Earlier RNA design methods explore the search space by trial and error and use classic cost function minimization approaches such as adaptive random walk (Hofacker 2003), probabilistic sampling (Zadeh et al. 2011), and genetic algorithms (Taneda 2015). Recent efforts employ more advanced machine learning methods such as different Monte Carlo Tree Search (MCTS) algorithms, e.g. MCTS-RNA (Yang et al. 2017) or Nested MCTS (Portela 2018), and reinforcement learning that either performs a local search as in (Eastman et al. 2018) or learns complete candidate solutions from scratch (Runge et al. 2019). In all these approaches, the assumption is that the algorithm has access to a large number of function evaluations, whereas we are interested in sample efficiency of each algorithm.

Learning to Search: As an alternative to parameter-free search methods (such as SA), (Swersky et al. 2020) suggests to use a parameterized policy to generate candidates that

maximize the acquisition function in Bayesian optimization over discrete search spaces. Our MCTS-based algorithm is similar in concept to (Swersky et al. 2020) in the sense that the tabular value functions are constructed and maintained over different time steps. However, we are maintaining value functions rather than a policy network. Finally, while (Swersky et al. 2020) trains a policy network, (Deshwal et al. 2020) uses a rank learning algorithm to search.

Meta-heuristics: A variety of discrete search algorithms and meta-heuristics have been studied in the literature for combinatorial optimization over categorical variables. Such algorithms, including Genetic Algorithms (Holland and Reitman 1978), Simulated Annealing (Spears 1993), and Particle Swarms (Kennedy and Eberhart 1995), are generally inefficient in finding the global minima. In the context of biological sequence optimization, the most popular method is directed evolution (Arnold 1998), which explores the space by only making small mutations to existing sequences. In the context of sequence optimization, a recent promising approach consists of fitting a neural network model to predict the black box function and then applying gradient ascent on the latter model (Killoran et al. 2017; Bogard et al. 2019; Liu et al. 2020). This approach allows for a continuous relaxation of the discrete search space making possible step-wise local improvements to the whole sequence at once based on a gradient direction. However, these methods have been shown to suffer from vanishing gradients (Linder and Seelig 2020). Further, the projected sequences in the continuous relaxation space may not be recognized by the predictors, leading to poor convergence. Generative model-based optimization approaches aim to learn distributions whose expectation coincides with evaluations of the black box and try to maximize such expectation (Gupta and Zou 2019; Brookes, Park, and Listgarten 2019). However, such approaches require a pre-trained generative model for optimization.

Latent Space Optimization: The key idea is to employ an encoder-decoder architecture to learn a continuous representation from the data and perform BO in a latent space (Gómez-Bombarelli et al. 2018; Tripp, Daxberger, and Hernández-Lobato 2020). This approach has two main drawbacks: it could generate a large fraction of invalid structures, and it requires a large database of relevant structures, for training an auto-encoder, which may not be available in many applications with scarce data availability.

Variable-Length Sequence Optimization: Moss et al. (Moss et al. 2020) propose an algorithm called BOSS for optimization over strings of variable lengths. They use a Gaussian process surrogate model based on string kernels and perform acquisition function maximization for spaces with syntactic constraints. On the contrary, our main focus is on fixed length generic BBO and design problems.

Ensemble (Population-based) Methods: Angermueller et al. (Angermüller et al. 2020) propose a population-based approach which generates batches of sequences by sampling from an ensemble of methods, rather than from a single method. This sampling is carried out proportional to the quality of sequences that each method has found in the past. A similar ensemble concept has been used in (Sinai et al. 2020). In our work, we focus on individual methods; however, we

add that since we are proposing two distinct representations employed via different algorithms (online and Bayesian), and the performance of each varies across different tasks, it would be interesting to consider an ensemble method over different algorithms as a future direction.

Batch (Offline) Optimization: Batch optimization techniques (Angermueller et al. 2020; Brookes, Park, and Listgarten 2019; Fannjiang and Listgarten 2020) are a related but fundamentally distinct line of work, where black-box evaluations are carried out in sample batches rather than in an active/sequential manner. To highlight the distinction between the two settings we iterate our problem setting: Given query access to a combinatorial black box (i.e. it can be queried on a single point), what is the best solution we can obtain in a given number T of such queries. One wishes to minimize the number of queries and hence goes for a highly adaptive sequential algorithm.

3 Black-Box Optimization over Categorical Variables

Problem Setting: Given the combinatorial categorical domain $\mathcal{X} = [k]^n$ and a constraint set $\mathcal{C} \subseteq \mathcal{X}$, with n variables each of cardinality k, the objective is to find

$$x^* = \arg\min_{x \in \mathcal{C}} f(x) \qquad (1)$$

where $f : \mathcal{X} \mapsto \mathbb{R}$ is a real-valued combinatorial function. We assume that f is a black-box function, which is computationally expensive to evaluate. As such, we are interested in finding x^* in as few evaluations as possible. We consider two variations of the problem depending on how the constraint set $\mathcal{C}$ is specified.

Generic BBO Problem: In this case, the constraint set $\mathcal{C} = \mathcal{X}$. For example, RNA sequence optimization problem that searches for an RNA sequence with a specific property optimized lies within this category. A score for every RNA sequence, reflecting the property we wish to optimize, is evaluated by a black box function.

Design Problem: The constraint set is complex and is only sequentially specified. For every sequence of $x_1 x_2 \ldots x_i$ consisting of i characters from the alphabet $[k]$, the choice of the next character $x_{i+1} \in \mathcal{C}(x_1 x_2 \ldots x_i) \subseteq [k]$ is specified by a constraint set function $\mathcal{C}(x_1 \ldots x_i)$. The RNA inverse folding problem in (Runge et al. 2019) falls into this category, where the constraints on the RNA sequence are determined by the sequential choices one makes during the sequence design. The goal is to find the sequence that is optimal with respect to a pre-specified structure that also obeys complex sequential constraints.

Proposed Techniques: To address this problem, we adopt a surrogate model-based learning framework as follows. The surrogate model (to represent the black-box function) is updated sequentially via black-box function evaluations observed until time step t. An acquisition function is then obtained from the surrogate model. The selection of candidate points for black-box function evaluation is carried out via an acquisition function optimizer (AFO), which uses the acquisition function as an inexpensive proxy (to make many internal

calls) for the black-box function and produces the next candidate point to be evaluated. We consider this problem in two different settings: online and Bayesian. The difference between the two settings is that in the former the acquisition function is exactly our surrogate, whereas in the latter the acquisition function is obtained by sampling from the posterior distribution of the surrogate.

In the sequel, we propose two representations that can be used in surrogate models for black-box combinatorial functions over categorical variables. These representations serve as direct generalizations of the Boolean surrogate model based on Fourier expansion proposed in (Dadkhahi et al. 2020; Ricardo Baptista 2018) in the sense that they reduce to the Fourier representation for real-valued Boolean functions when the cardinality of the categorical variables is two. In addition, both approaches can be modified to address the more general case where different variables are of different cardinalities. However, for ease of exposition, we assume that all the variables are of the same cardinality. Finally, we introduce two popular search algorithms to be used as acquisition function optimizers in conjunction with the proposed surrogate models in order to select new queries for subsequent black-box function evaluations.

3.1 Representations for the Surrogate Model

We present two representations for combinatorial functions $f : [k]^n \to \mathbb{R}$ and an algorithm to update from the black-box evaluations. The representations use the Fourier basis in two different ways.

Abridged One-Hot Encoded Boolean Fourier Representation: The one-hot encoding of each variable $x_i : i \in [n]$ can be expressed as a $(k-1)$-tuple $(x_{i1}, x_{i2}, \ldots, x_{i(k-1)})$, where $x_{ij} \in \{-1, 1\}$ are Boolean variables with the constraint that at most one such variable can be equal to -1 for any given $x_i \in [k]$.

We consider the following representation for the combinatorial function f:

$$f_\alpha(x) = \sum_{m=0}^{n} \sum_{\mathcal{I} \in \binom{[n]}{m}} \sum_{\mathcal{J} \in [k-1]^{|\mathcal{I}|}} \alpha_{\mathcal{I},\mathcal{J}} \psi_{\mathcal{I},\mathcal{J}}(x) \quad (2)$$

where $[k-1]^{|\mathcal{I}|}$ denotes $|\mathcal{I}|$-fold cartesian product of the set $[k-1] = \{1, 2, \ldots, k-1\}$, $\binom{[n]}{m}$ designates the set of m-subsets of the set $[n]$, and the monomials $\psi_{\mathcal{I},\mathcal{J}}(x)$ can be written as

$$\psi_{\mathcal{I},\mathcal{J}}(x) = \prod_{\{(i,j): i = \mathcal{I}_\ell, j = \mathcal{J}_\ell, \ell \in [|\mathcal{I}|]\}} x_{ij} \quad (3)$$

A second order approximation (i.e. at $m = 2$) of the representation in (2) can be expanded in the following way:

$$\widehat{f}_\alpha(x) = \alpha_0 + \sum_{i \in [n]} \sum_{\ell \in [k-1]} \alpha_{i\ell} x_{i\ell}$$
$$+ \sum_{(i,j) \in \binom{[n]}{2}} \sum_{(p,q) \in [k-1]^2} \alpha_{ijpq} x_{ip} x_{jq}. \quad (4)$$

Example. *For $n = 2$ variables x_1 and x_2, each of which with cardinality $k = 3$, we have the one-hot encoding of*

(x_{11}, x_{12}) and (x_{21}, x_{22}) respectively. From Equation (4), the one-hot encoding factorization for this example can be written as

$$f(x) = \alpha_0 + \alpha_1 x_{11} + \alpha_2 x_{12} + \alpha_3 x_{21} + \alpha_4 x_{22}$$
$$+ \alpha_5 x_{11} x_{21} + \alpha_6 x_{11} x_{22} + \alpha_7 x_{12} x_{21} + \alpha_8 x_{12} x_{22}.$$

Note that the representation in Equation (2) has far less terms than a vanilla one-hot encoding with all the combinations of one-hot variables included (as suggested in (Ricardo Baptista 2018)). The reason for this reduction is twofold: (i) $(k-1)$ Boolean variables model each categorical variable of cardinality k, and more importantly (ii) each monomial term has at most one Boolean variable x_{ij} from its corresponding parent categorical variable x_i (See the Appendix for the exact quantification of this reduction). The following theorem states that this reduced representation is in fact unique and complete.

Theorem 3.1. *The representation in Equation (2) is complete and unique for any real-valued combinatorial function.*

Proof. See Appendix. □

Fourier Representation on Finite Abelian Groups: We define a cyclic group structure $\mathbb{Z}/k_i\mathbb{Z}$ over the elements of each categorical variable x_i ($i \in [n]$), where k_i is the cardinality of the latter variable. From the fundamental theorem of abelian groups (Terras 1999), there exists an abelian group G which is isomorphic to the direct sum (a.k.a direct product) of the cyclic groups $\mathbb{Z}/k_i\mathbb{Z}$ corresponding to the n categorical variables:

$$G \cong \mathbb{Z}/k_1\mathbb{Z} \oplus \mathbb{Z}/k_2\mathbb{Z} \oplus \ldots \oplus \mathbb{Z}/k_n\mathbb{Z} \quad (5)$$

where the latter group consists of all vectors $(a_1, a_2, \ldots, a_n)$ such that $a_i \in \mathbb{Z}/k_i\mathbb{Z}$ and $\cong$ denotes group isomorphism. We assume that $k_i = k$ ($\forall i \in [n]$) for simplicity, but the following representation could be easily generalized to the case of arbitrary cardinalities for different variables.

The Fourier representation of any complex-valued function $f(x)$ on the finite abelian group G is given by (Terras 1999)

$$f(x) = \sum_{\mathcal{I} \in [k]^n} \alpha_{\mathcal{I}} \psi_{\mathcal{I}}(x) \quad (6)$$

where $\alpha_{\mathcal{I}}$ are (in general complex) Fourier coefficients, $[k]^n$ is the n-fold cartesian product of the set $[k]$ and $\psi_{\mathcal{I}}(x)$ are complex exponentials [1] (k-th roots of unity) given by

$$\psi_{\mathcal{I}}(x) = \exp(2\pi j \langle x, \mathcal{I} \rangle / k).$$

Note that the latter complex exponentials are the *characters* of the representation, and reduce to the *monomials* (i.e. in $\{-1, 1\}$) when the cardinality of each variable is two. A

---

[1]Note that in the general case of different cardinalities for different variables, $\mathcal{I} \in [k_1] \times [k_2] \times \ldots \times [k_n]$ where $\times$ denotes the cartesian product and the exponent denominator in the complex exponential character is replaced by $k = \mathsf{LCM}(k_1, k_2, \ldots, k_n)$.

second order approximation of the representation in (6) can be written as:

$$\widehat{f}_\alpha(x) = \alpha_0 + \sum_{i \in [n]} \sum_{\ell \in [k-1]} \alpha_{i\ell} \exp(2\pi j x_i \ell/k) \qquad (7)$$
$$+ \sum_{(i,j) \in \binom{[n]}{2}} \sum_{(p,q) \in [k-1]^2} \alpha_{ijpq} \exp(2\pi j(x_i p + x_j q)/k).$$

For a real-valued function $f_\alpha(x)$ (which is of interest here), the representation in (6) reduces to

$$f_\alpha(x) = \Re \left\{ \sum_{\mathcal{I} \in [k]^n} \alpha_{\mathcal{I}} \psi_{\mathcal{I}}(x) \right\}$$
$$= \sum_{\mathcal{I} \in [k]^n} \alpha_{r,\mathcal{I}} \psi_{r,\mathcal{I}}(x) - \sum_{\mathcal{I} \in [k]^n} \alpha_{i,\mathcal{I}} \psi_{i,\mathcal{I}}(x) \qquad (8)$$

where

$$\psi_{r,\mathcal{I}}(x) = \cos(2\pi\langle x, \mathcal{I}\rangle/k) \quad \text{and} \quad \psi_{i,\mathcal{I}}(x) = \sin(2\pi\langle x, \mathcal{I}\rangle/k)$$
$$\alpha_{r,\mathcal{I}} = \Re\{\alpha_{\mathcal{I}}\} \quad \text{and} \quad \alpha_{i,\mathcal{I}} = \Im\{\alpha_{\mathcal{I}}\} \qquad (9)$$

See the Appendix for the proof of this representation. We note that the number of characters utilized in this representation is almost twice as many as that of monomials used in the previous representation.

3.2 Surrogate Model Learning

Sparse Online Regression: We adopt the learning algorithm of combinatorial optimization with expert advice (Dadkhahi et al. 2020) in the following way. We consider the monomials $\psi_{\mathcal{I},\mathcal{J}}(x)$ in (3) and the characters $\psi_{\ell,\mathcal{I}}(x)$ in (9) as experts. For each surrogate model, we maintain a pool of such experts, the coefficients of which are refreshed sequentially via an exponential weight update rule. We refer to the proposed algorithm as *Expert-Based Categorical Optimization* (ECO) and the two versions of the algorithm with the two proposed surrogate models are called ECO-F (based on the One-Hot Encoded Boolean Fourier Representation) and ECO-G (based on Fourier Representation on Finite Abelian Groups), respectively. See Appendix for more details.

Sparse Bayesian Regression: In order to incorporate the uncertainty of the surrogate model in acquisition function for exploration-exploitation trade-off, we adopt a Bayesian setting based on Thompson sampling (TS) (Thompson 1933, 1935). In particular, we model the surrogate via a sparse Bayesian regression model with a regularized horseshoe prior (Piironen and Vehtari 2017). At each time step, we learn the posterior using the data observed so far, and draw a sample for the coefficients α from the posterior. We use the No-U-Turn Sampler (NUTS) (Hoffman and Gelman 2014) in order to sample the coefficients efficiently. See Appendix B for more details. Our algorithm pursues the framework of BOCS (Ricardo Baptista 2018), with the following differences for improved performance and computational efficiency: BOCS (*i*) employs a horseshoe prior (Carvalho, Polson, and Scott 2010; Makalic and Schmidt 2016) and (*ii*) uses a Gibbs sampler. Experimentally, the regularized horseshoe prior outperforms the horseshoe prior, and the NUTS sampler leads to a significant speed-up in computations (e.g. a $10\times$ speed-up for

the RNA optimization problem), which is critical due to the higher dimensionality of the categorical domains. We use the proposed abridged one-hot encoded Fourier representation in conjunction with this model, due to its lower number of coefficients leading to a more manageable computation cost. We refer to this algorithm as *TS-based Categorical Optimization* (TCO-F).

3.3 Acquisition Function Optimizers

In this subsection, we discuss how two popular search algorithms, namely Simulated Annealing (SA) and Monte Carlo Tree Search (MCTS), can be utilized as acquisition function optimizers (AFO) in conjunction with a surrogate model and use cost-free evaluations of the surrogate model to select the next query for the black box evaluation. In the literature, SA has been used for the generic BBO problems, whereas MCTS has been used for the design problems.

SA as AFO: Our AFO is devised so as to minimize $\widehat{f}_\alpha(x)$, the current estimate for the surrogate model. A simple strategy to minimize $\widehat{f}_\alpha(x)$ is to iteratively switch each variable into the value that minimizes $\widehat{f}_\alpha$ given the values of all the remaining variables, until no more changes occur after a sweep through all the variables x_i ($\forall i \in [n]$). A strategy to escape local minima in this context (Pincus 1970) is to allow for occasional increases in $\widehat{f}_\alpha$ by generating a Markov Chain (MC) sample sequence (for x), whose stationary distribution is proportional to $\exp(-\widehat{f}_\alpha(x)/s)$, where s is gradually reduced to 0. This optimization strategy was first applied by (Kirkpatrick, Gelatt, and Vecchi 1983) in their Simulated Annealing algorithm to solve combinatorial optimization problems. We use the Gibbs sampler (Geman and Geman 1984) to generate such an MC by sampling from the full conditional distribution of the stationary distribution, which in our case is given by the softmax distribution over $\{-\widehat{f}_\alpha(x_i = \ell, x_{-i})/s\}_{\ell \in [k]}$, for each variable x_i conditional on the values of the remaining variables x_{-i}. By decreasing s from a high value to a low one, we allow the MC to first search at a coarse level avoiding being trapped in local minima.

Algorithm 1 presents our simulated annealing (SA) version for categorical domains, where $s(t)$ is an annealing schedule, which is a non-increasing function of t. We use the annealing schedule suggested in (Spears 1993), which follows an exponential decay with parameter ℓ given by $s(t) = \exp(-\ell t/n)$. In each step of the algorithm, we pick a variable x_i ($i \in [n]$) uniformly at random, evaluate the surrogate model (possibly in parallel) k times, once for each categorical value $\ell \in [k]$ for the chosen variable x_i given the current values x_{-i} for the remaining variables. We then update x_i with the sampled value in $[k]$ from the corresponding softmax distribution.

MCTS as AFO: We formulate the design problem as an undiscounted Markov decision process $(\mathcal{S}, \mathcal{A}, T, R)$. Each state $s \in \mathcal{S}$ corresponds to a partial or full sequence of categorical variables of lengths in $[0, n]$. The process in each episode starts with an empty sequence s_0, the initial state. Actions are selected from the set of permissible additions to the current state (sequence) s_t at each time step t, $\mathcal{A}_t = \mathcal{A}(s_t) \subset \mathcal{A}$. The transition function T is deterministic, and defines the sequence obtained from the jux-

---

Algorithm 1: SA for Categorical Variables with Surrogate Model

1: **Inputs:** surrogate model $\widehat{f}_\alpha$, annealing schedule $s(t)$, categorical domain $\mathcal{X}$
2: Initialize $x \in \mathcal{X}$
3: $t = 0$
4: **repeat**
5:   $i \sim \texttt{unif}([n])$
6:   $x_i | x_{-i} \sim \texttt{Softmax}\big(\{-\widehat{f}_{\alpha_t}(x_i=\ell, x_{-i})/s(t)\}_{\ell \in [k]}\big)$
7:   $t \leftarrow t+1$
8: **until** Stopping Criteria
9: **return** x

Algorithm 2: MCTS with Surrogate Reward

1: **Inputs:** surrogate model $\widehat{f}_\alpha$, search tree $\mathcal{T}$
2: Initialize $s_{\text{best}} = \{\}, r_{\text{best}} = -\infty$
3: **repeat**
4:   $s_{\text{leaf}} \leftarrow \texttt{Selection}(\pi^{\mathcal{T}})$
5:   $\mathcal{T} \leftarrow \mathcal{T} \cup \{s_{\text{leaf}}\}$
6:   $s_t \leftarrow \texttt{Simulation}(\pi^{RS}, s_{\text{leaf}})$
7:   $r \leftarrow -\widehat{f}_\alpha(s_t)$
8:   $\texttt{Backup}(s_{\text{leaf}}, r)$
9:   **if** $r > r_{\text{best}}$ **then**
10:     $r_{\text{best}} \leftarrow r$ and $s_{\text{best}} \leftarrow s_t$
11:   **end if**
12: **until** Stopping Criteria
13: **return** s_{best}

---

taposition of the current state s_t with the action a_t, i.e. $s_{t+1} = T(s_t, a_t) = s_t \circ a_t$. The transitions leading to incomplete sequences yield a reward of zero. Complete sequences are considered as terminal states, from which no further transitions (juxtapositions) can be made. Once the sequence is complete (i.e. at a terminal state), the reward is obtained from the current surrogate reward model $\widehat{f}_\alpha$. Thus, the reward function is defined as $R(s_t, a_t, s_{t+1}) = -\widehat{f}_\alpha(s_{t+1})$ if s_{t+1} is terminal, and zero otherwise.

MCTS is a popular search algorithm used for design problems. MCTS is a rollout algorithm which keeps track of the value estimates obtained via Monte Carlo simulations in order to progressively make better selections. The UCT selection criteria, see (Kocsis and Szepesvári 2006), is typically used as tree policy, where action a_t at state s_t in the search tree is selected via: $\pi^{\mathcal{T}}(s_t) = \arg\max_{a \in \mathcal{A}(s_t)} Q(s_t, a) + c\sqrt{\ln N(s_t)/N(s_t, a)}$, where $\mathcal{T}$ is the search tree, c is the exploration parameter, $Q(s, a)$ is the state-action value estimate, and $N(s)$ and $N(s, a)$ are the visit counts for the parent state node and the candidate state-action edge, respectively. For the selection of actions in states outside the search tree, a random default policy is used: $\pi^{RS}(s_t) = \texttt{unif}(\mathcal{A}_t)$.

A summary of the proposed algorithm is given in Algorithm 2. Starting with an empty sequence s_0 at the root of the tree, we follow the tree policy until a leaf node of the search tree is reached (selection step). At this point, we append the state corresponding to the leaf node to the tree and initialize a value function estimate for its children (extension step). From the reached leaf node we follow the default policy until a terminal state is reached. At this point, we plug the sequence corresponding to this terminal state into the surrogate reward model $-\widehat{f}_\alpha$ and observe the reward r. This reward is backed up from the leaf node to the root of the tree in order to update the value estimates $Q(s, a)$ via Monte Carlo (i.e. using the average reward) for all visited (s, a) pairs along the path. This process is repeated until a stopping criterion (typically a max number of playouts) is met, at which point the sequence s_{best} with maximum reward r_{best} is returned as the output of the algorithm.

Computational Complexity: The computational complexity per time step associated with learning the surrogate model via the Hedge algorithm, for both representations introduced in 3.1, is in $\mathcal{O}(d) = \mathcal{O}(k^{m-1}n^m)$, and is thus linear in the number of experts d. Moreover, the complexity of Algorithm 1 is in $\mathcal{O}(kk^{m-1}n^{m-1}n) = \mathcal{O}(kd)$, assuming that the number of iterations in each SA run is in $\mathcal{O}(n)$. Hence, the overall complexity of the algorithm is in $\mathcal{O}(kd)$. Finally, the computational complexity of each playout in Algorithm 2 is in $\mathcal{O}(kn)$, leading to an overall complexity of $\mathcal{O}(kd)$, assuming $\mathcal{O}(\frac{d}{n})$ playouts per time step. On the other hand, when Bayesian regression is used to learn the surrogate model, NUTS dominates the computation time with a complexity of $\mathcal{O}(d^{5/4})$ per independent sample.

4 Experiments and Results

In this section, we measure the performance of the proposed representations, when used as surrogate/reward model in conjunction with search algorithms (SA and MCTS) in BBO and design problems. The learning rate used in exponential weight updates is selected via the anytime learning rate schedule suggested in (Dadkhahi et al. 2020) and (Gerchinovitz and Yu 2011) (see Appendix). The maximum degree of interactions used in our surrogate models is set to two for all the problems; increasing the max order improved the results only marginally (see Appendix). The sparsity parameter λ in exponential weight updates is set to 1 in all the experiments following the same choice made in (Dadkhahi et al. 2020). Experimentally, the learning algorithm is fairly insensitive to the variations in the latter parameter. In each experiment, we report the results averaged over multiple runs (20 runs in BBO experiments; 10 runs in design experiments) $\pm$ one standard error of the mean. The experiments are run on machines with CPU cores from the Intel Xeon E5-2600 v3 family.

BBO Experiments: We compare the performance of our ECO/TCO algorithms in conjunction with SA with two baselines, random search (RS) and simulated annealing (SA), as well as a state-of-the-art Bayesian combinatorial optimization algorithm (COMBO) (Oh et al. 2019). In particular, we consider two synthetic benchmarks (Latin square problem and pest control problem) and a real-word problem in biology (RNA sequence optimization). In addition to the performance of the algorithms in terms of the best value of $f(x)$ observed until a given time step t, we measure the average computation time per time step of our algorithm versus that of COMBO. The decay parameter used in the annealing schedule of SA is

DATA	LATIN SQUARE	PEST CONT.	RNA OPT.
COMBO	170.4	151.0	253.8
ECO-F	1.5	1.4	2.0
ECO-G	3.6	3.3	5.7
TCO-F	55.7	53.2	67.0

Table 1: Average computation time per step (in Seconds).

set to $\ell = 3$ in all the experiments. In addition, the number of SA iterations T is set to $3 \times n$ and $6 \times n$ for ECO and TCO, respectively. Intuitively, for the ECO algorithm, each of these parameters creates an exploration-exploitation trade-off. The smaller (larger) the value of ℓ or T, the more exploratory (exploitative) is the behavior of SA. The selected values seem to create a reasonable balance; tuning these parameters may improve the performance of the algorithm. On the other hand, the TCO algorithm has an exploration mechanism built into it via Thompson sampling; hence, we use a higher number of SA iterations to maximize exploitation in AFO.

Synthetic Benchmarks: We consider two synthetic problems: Latin square problem (Colbourn and Dinitz 2006), a commonly used combinatorial optimization benchmark, and the pest control problem considered in (Oh et al. 2019) (see Appendix for the latter results). In both problems, we have $n = 25$ categorical variables, each of cardinality $k = 5$. A Latin square of order k is a $k \times k$ matrix of elements $x_{ij} \in [k]$, such that each number appears in each row and column exactly once. When $k = 5$, the problem of finding a Latin square has $161,280$ solutions in a space of dimensionality 5^{25}. We formulate the problem of finding a Latin square of order k as a black-box optimization by imposing an additive penalty of 1 for any repetition of numbers in any row or column. Hence, function evaluations are in the range $[0, 2k(k-1)]$, and a function evaluation of 0 corresponds to a Latin square of order k. We consider a noisy version of this problem, where an additive Gaussian noise with 0 mean and standard deviation of 0.1 is added to function evaluations observed by each algorithm.

Figure 1 demonstrates the performance of different algorithms, in terms of the best function value found until time t, over 500 time steps. Both ECO-F and ECO-G outperform the baselines with a considerable margin. In addition, ECO-G outperforms COMBO at 130 samples. At larger time steps, COMBO outperforms the other algorithms; however, this performance comes at the price of a far larger computation time. As demonstrated in Table 1, ECO-F and ECO-G offer a speed-up of roughly 100 and 50, resp., over COMBO. We note that TCO-F (not included in the plot) performs poorly (similar to RS) on this problem, which can be attributed to the strong promotion of sparsity by the regularized horseshoe prior and the fact that the Latin Square problem has a dense representation (we observed a similar behavior from the horseshoe prior of (Ricardo Baptista 2018)).

RNA Sequence Optimization Problem: Consider an RNA sequence as a string $A = a_1 \ldots a_n$ of n letters (nucleotides) over the alphabet $\Sigma = \{A, U, G, C\}$. A pair of complementary nucleotides a_i and a_j, where $i < j$, can
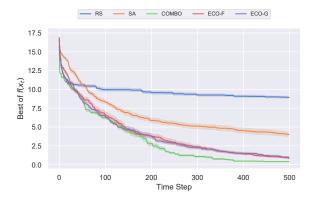

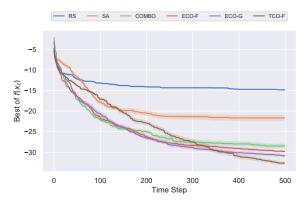Figure 1: Latin Square problem with $n = 25$.

Figure 2: RNA BBO Problem with $n = 30$

interact with each other and form a base pair (denoted by (i, j)), A-U, C-G and G-U being the energetically stable pairs. Thus, the secondary structure, i.e. the minimum free energy structure, of an RNA can be represented by an ensemble of pairing bases. A number of RNA folding algorithms (Lorenz et al. 2011; Markham and Zuker 2008) use a thermodynamic model (e.g. (Zuker and Stiegler 1981)) and dynamic programming to estimate MFE of a sequence. However, the $\mathcal{O}(n^3)$ time complexity of these algorithms prohibits their use for evaluating substantial numbers of RNA sequences (Gould, Hendy, and Papamichail 2014) and exhaustively searching the space to identify the global free energy minimum, as the number of sequences grows exponentially as 4^n.

We formulate the RNA sequence optimization problem as follows: For a sequence of length n, find the RNA sequence which folds into a secondary structure with the lowest MFE. In our experiments, we set $n = 30$ and $k = 4$. We then use the popular RNAfold package (Lorenz et al. 2011) to evaluate the MFE for a given sequence. The goal is to find the lowest MFE sequence by calling the MFE evaluator minimum number of times. As shown in Figure 2, both ECO-F and particularly ECO-G outperform the baselines as well as COMBO by a large margin. At higher number of evaluations, TCO-F beats the rest of the algorithms, which can be attributed to its exploration-exploitation trade-off.

RNA Design Experiments: The problem is to find a primary RNA sequence ϕ which folds into a target structure ω,

given a folding algorithm F. Such target structures can be represented as a sequence of dots (for unpaired bases) and brackets (for paired bases). In our algorithm, the action sets are defined as follows. For unpaired sites $\mathcal{A}_t = \{A, G, C, U\}$ and for paired sites $\mathcal{A}_t = \{GC, CG, AU, UA\}$. At the beginning of each run of our algorithm (ECO-F/G, TCO-F in conjunction with MCTS), we draw a random permutation for the order of locations to be selected in each level of the tree. The reward value offered by the environment (i.e. the black-box function) at any time step t corresponds to the normalized Hamming distance between the target structure ω and the structure $y_t = F(x_t)$ of the sequence x_t found by each algorithm, i.e. $d_H(\omega, y_t)$.

We compare the performance of our algorithms against RS as a baseline, where random search is carried out over the given structure (i.e. default policy π^{RS}) rather than over unstructured random sequences. We also include two state-of-the-art algorithms in our experiments: MCTS-RNA (Yang et al. 2017) and LEARNA (Runge et al. 2019). MCTS-RNA has an exploration parameter, which we tune in advance (per sequence). LEARNA has a set of 14 hyper-parameters tuned a priori using training data and is provided in (Runge et al. 2019). Note that the latter training phase (for LEARNA) as well as the former exploration parameter tuning (for MCTS-RNA) are offered to the respective algorithms as an advantage, whereas for our algorithm we use a global set of heuristic choices for the two hyper-parameters, rather than attempting to tune the two hyper-parameters. In particular, we set the exploration parameter c to 0.5 for ECO and 0.25 for TCO; and the number of MCTS playouts at each time step to $30 \times h$, where h is the tree height (i.e. number of dots and bracket pairs). The latter heuristic choice is made since the bigger the tree, the more playouts are needed to explore the space.

We point out that the entire design pipeline in state-of-the-art algorithms typically also includes a local improvement step (as a post-processing step), which is either a rule-based search (e.g. in (Yang et al. 2017)) or an exhaustive search (e.g. in (Runge et al. 2019)) over the mismatched sites. We do not include the local improvement step in our experiments, since we are interested in measuring sample efficiency of different algorithms. In other words, the question is the following: given a fixed and finite evaluation budget, which algorithm is able to get closer to the target structure.

In our experiments, we focus on three puzzles from the Eterna-100 dataset (Anderson-Lee et al. 2016). Two of the selected puzzles (#15 and #41 of lengths 30 and 35, resp.), despite their fairly small lengths, are challenging for many algorithms (see (Anderson-Lee et al. 2016)). In both puzzles, ECO-F, ECO-G, and TCO-F (with MCTS as AFO) are able to significantly improve the performance of MCTS when limited number of black-box evaluations is available. All algorithms outperform RS as expected. Within the given 500 evaluation budget, TCO-F, ECO-G, and esp. ECO-F, are superior to LEARNA by a substantial margin (see Figure 3). In puzzle 41 (Figure 4), again both ECO-G and ECO-F significantly outperform LEARNA over the given number of evaluations. ECO-F is able to outperform LEARNA throughout the evaluation process, and on average finds a far better final solution than LEARNA. With MCTS as AFO, ECO

algorithms outperform TCO, which can be attributed to the exploratory behavior of the latter in both AF (via TS) and AFO (via MCTS). See Appendix for puzzle #70.
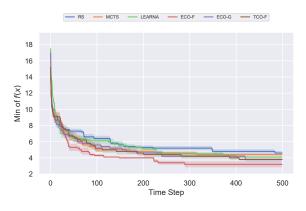

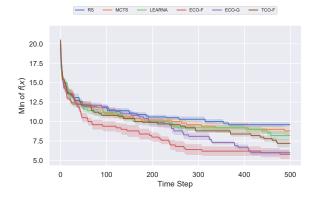Figure 3: Design puzzle #15 with $n = 30$.

Figure 4: Design puzzle #41 with $n = 35$.

5 Conclusions and Future Work

We propose two novel Fourier representations as surrogate models for black box optimization over categorical variables and show performance improvements over existing baselines when combined with search algorithms. We utilize two algorithms to learn such surrogate models. Our ECO algorithm incorporates a computationally-efficient online estimator with strong adversarial guarantees (see (Dadkhahi et al. 2020) for theoretical results in the Boolean case), which can be shown to carry over to our setting as well. Our TCO algorithm uses a Bayesian regression model with a regularized horseshoe prior and selects queries via Thompson sampling to balance exploration-exploitation trade-off in surrogate model learning at the price of a higher computation cost.

Considering the performance variability with respect to different algorithms and representations across different problems, an important research avenue would be to derive an ensemble of such models rather than a single one and investigate potential performance gains. Such an ensemble model would then update and explore all models simultaneously and draw samples from either individual or a combination of models at any given time step.

References

Anderson-Lee, J.; Fisker, E.; Kosaraju, V.; Wu, M.; Kong, J.; Lee, J.; Lee, M.; Zada, M.; Treuille, A.; and Das, R. 2016. Principles for Predicting RNA Secondary Structure Design Difficulty. *Journal of Molecular Biology*, 428(5, Part A): 748 – 757. Challenges in RNA Structural Modeling and Design.

Angermueller, C.; Dohan, D.; Belanger, D.; Deshpande, R.; Murphy, K.; and Colwell, L. 2020. Model-based reinforcement learning for biological sequence design. In *ICLR*.

Angermüller, C.; Belanger, D.; Gane, A.; Mariet, Z.; Dohan, D.; Murphy, K.; Colwell, L. J.; and Sculley, D. 2020. Population-Based Black-Box Optimization for Biological Sequence Design. In *ICML*.

Arnold, F. H. 1998. Design by directed evolution. *Accounts of chemical research*, 31(3): 125–131.

Bergstra, J.; Bardenet, R.; Bengio, Y.; and Kégl, B. 2011. Algorithms for Hyper-Parameter Optimization. In *NeurIPS*.

Bogard, N.; Linder, J.; Rosenberg, A. B.; and Seelig, G. 2019. A deep neural network for predicting and engineering alternative polyadenylation. *Cell*, 178(1): 91–106.

Brookes, D.; Park, H.; and Listgarten, J. 2019. Conditioning by adaptive sampling for robust design. In *ICML*.

Carvalho, C. M.; Polson, N. G.; and Scott, J. G. 2010. The horseshoe estimator for sparse signals. *Biometrika*, 97(2): 465–480.

Colbourn, C. J.; and Dinitz, J. H. 2006. *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC.

Dadkhahi, H.; Shanmugam, K.; Rios, J.; Das, P.; Hoffman, S.; Loeffler, T. D.; and Sankaranarayanan, S. 2020. Combinatorial Black-Box Optimization with Expert Advice. In *KDD*.

Deshwal, A.; Belakaria, S.; and Doppa, J. R. 2021. Mercer Features for Efficient Combinatorial Bayesian Optimization. *AAAI*, 35(8): 7210–7218.

Deshwal, A.; Belakaria, S.; Doppa, J. R.; and Fern, A. 2020. Optimizing Discrete Spaces via Expensive Evaluations: A Learning to Search Framework. In *AAAI*.

Dill, K. A.; and MacCallum, J. L. 2012. The protein-folding problem, 50 years on. *Science*, 338(6110): 1042–1046.

Dixon, N.; Duncan, J. N.; Geerlings, T.; Dunstan, M. S.; McCarthy, J. E.; Leys, D.; and Micklefield, J. 2010. Reengineering orthogonally selective riboswitches. *Proceedings of the National Academy of Sciences*, 107(7): 2830–2835.

Eastman, P.; Shi, J.; Ramsundar, B.; and Pande, V. S. 2018. Solving the RNA design problem with reinforcement learning. *PLoS computational biology*, 14(6): e1006176.

Fannjiang, C.; and Listgarten, J. 2020. Autofocused oracles for model-based design. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *NeurIPS*.

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.

Geman, S.; and Geman, D. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on pattern analysis and machine intelligence*, 6: 721–741.

Gerchinovitz, S.; and Yu, J. Y. 2011. Adaptive and Optimal Online Linear Regression on ℓ1-Balls. In *Algorithmic Learning Theory*.

Golovin, D.; Solnik, B.; Moitra, S.; Kochanski, G.; Karro, J.; and Sculley, D. 2017. Google vizier: A service for black-box optimization. In *KDD*, 1487–1495.

Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; and Aspuru-Guzik, A. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276.

Gould, N.; Hendy, O.; and Papamichail, D. 2014. Computational tools and algorithms for designing customized synthetic genes. *Frontiers in bioengineering and biotechnology*, 2: 41.

Gupta, A.; and Zou, J. 2019. Feedback GAN for DNA optimizes protein functions. *Nature Machine Intelligence*, 1(2): 105–111.

Hofacker, I. L. 2003. Vienna RNA secondary structure server. *Nucleic acids research*, 31(13): 3429–3431.

Hofacker, I. L.; Fontana, W.; Stadler, P. F.; Bonhoeffer, L. S.; Tacker, M.; and Schuster, P. 1994. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chemical Monthly*, 125(2): 167–188.

Hoffman, M. D.; and Gelman, A. 2014. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(47): 1593–1623.

Holland, J. H.; and Reitman, J. S. 1978. Cognitive systems based on adaptive algorithms. In *Pattern-directed inference systems*, 313–329. Elsevier.

Hoshika, S.; Leal, N. A.; Kim, M.-J.; Kim, M.-S.; Karalkar, N. B.; Kim, H.-J.; Bates, A. M.; Watkins, N. E.; SantaLucia, H. A.; Meyer, A. J.; et al. 2019. Hachimoji DNA and RNA: A genetic system with eight building blocks. *Science*, 363(6429): 884–887.

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.

Kennedy, J.; and Eberhart, R. 1995. Particle swarm optimization. In *International Conference on Neural Networks (ICNN)*, volume 4, 1942–1948 vol.4.

Killoran, N.; Lee, L. J.; Delong, A.; Duvenaud, D.; and Frey, B. J. 2017. Generating and designing DNA with deep generative models. *arXiv:1712.06148*.

Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983. Optimization by Simulated Annealing. *Science*, 220(4598): 671–680.

Kocsis, L.; and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In *European Conference on Machine Learning (ECML)*, 282–293.

Leprêtre, F.; Verel, S.; Fonlupt, C.; and Marion, V. 2019. Walsh functions as surrogate model for pseudo-boolean optimization problems. In *The Genetic and Evolutionary Computation Conference (GECCO 2019)*, 303–311. ACM.

Linder, J.; and Seelig, G. 2020. Fast differentiable DNA and protein sequence optimization for molecular design. Arxiv:2005.11275.

Liu, G.; Zeng, H.; Mueller, J.; Carter, B.; Wang, Z.; Schilz, J.; Horny, G.; Birnbaum, M. E.; Ewert, S.; and Gifford, D. K. 2020. Antibody complementarity determining region design using high-capacity machine learning. *Bioinformatics*, 36(7): 2126–2133.

Lorenz, R.; Bernhart, S. H.; Höner zu Siederdissen, C.; Tafer, H.; Flamm, C.; Stadler, P. F.; and Hofacker, I. L. 2011. ViennaRNA Package 2.0. *Algorithms for Molecular Biology*, 6(1): 26.

Makalic, E.; and Schmidt, D. F. 2016. A Simple Sampler for the Horseshoe Estimator. *IEEE Signal Processing Letters*, 23(1): 179–182.

Markham, N. R.; and Zuker, M. 2008. UNAFold. In *Bioinformatics*, 3–31. Springer.

Moss, H.; Leslie, D.; Beck, D.; González, J.; and Rayson, P. 2020. BOSS: Bayesian Optimization over String Spaces. In *NeurIPS*.

Ng, A. H.; Nguyen, T. H.; Gómez-Schiavon, M.; Dods, G.; Langan, R. A.; Boyken, S. E.; Samson, J. A.; Waldburger, L. M.; Dueber, J. E.; Baker, D.; et al. 2019. Modular and tunable biological feedback control using a de novo protein switch. *Nature*, 572(7768): 265–269.

Oh, C.; Tomczak, J.; Gavves, E.; and Welling, M. 2019. Combinatorial Bayesian Optimization using the Graph Cartesian Product. In *NeurIPS*.

Piironen, J.; and Vehtari, A. 2017. Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electronic Journal of Statistics*, 11(2): 5018–5051.

Pincus, M. 1970. Letter to the editor—a Monte Carlo method for the approximate solution of certain types of constrained optimization problems. *Operations Research*, 18(6): 1225–1228.

Portela, F. 2018. An unexpectedly effective Monte Carlo technique for the RNA inverse folding problem. *BioRxiv*, 345587.

Ricardo Baptista, M. P. 2018. Bayesian Optimization of Combinatorial Structures. In *ICML*.

Runge, F.; Stoll, D.; Falkner, S.; and Hutter, F. 2019. Learning to Design RNA. In *ICLR*.

Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R. P.; and De Freitas, N. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175.

Sinai, S.; Wang, R.; Whatley, A.; Slocum, S.; Locane, E.; and Kelsic, E. 2020. AdaLead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint*.

Spears, W. M. 1993. Simulated Annealing for Hard Satisfiability Problems. In *DIMACS Workshop: Cliques, Coloring, and Satisfiability*, 533–557.

Stephens, Z. D.; Lee, S. Y.; Faghri, F.; Campbell, R. H.; Zhai, C.; Efron, M. J.; Iyer, R.; Schatz, M. C.; Sinha, S.; and Robinson, G. E. 2015. Big data: astronomical or genomical? *PLoS biology*, 13(7): e1002195.

Swersky, K.; Rubanova, Y.; Dohan, D.; and Murphy, K. 2020. Amortized Bayesian Optimization over Discrete Spaces. In *UAI*.

Taneda, A. 2015. Multi-objective optimization for RNA design with multiple target secondary structures. *BMC bioinformatics*, 16(1): 280.

Terras, A. 1999. *Fourier Analysis on Finite Groups and Applications*. London Mathematical Society Student Texts. Cambridge University Press.

Thompson, W. R. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4): 285–294.

Thompson, W. R. 1935. On the Theory of Apportionment. *American Journal of Mathematics*, 57: 450.

Tripp, A.; Daxberger, E.; and Hernández-Lobato, J. M. 2020. Sample-Efficient Optimization in the Latent Space of Deep Generative Models via Weighted Retraining. In *Advances in Neural Information Processing Systems*.

Yamagami, R.; Kayedkhordeh, M.; Mathews, D. H.; and Bevilacqua, P. C. 2019. Design of highly active double-pseudoknotted ribozymes: a combined computational and experimental study. *Nucleic acids research*, 47(1): 29–42.

Yang, X.; Yoshizoe, K.; Taneda, A.; and Tsuda, K. 2017. RNA inverse folding using Monte Carlo tree search. *BMC Bioinformatics*, 8: 468.

Zadeh, J. N.; Steenberg, C. D.; Bois, J. S.; Wolfe, B. R.; Pierce, M. B.; Khan, A. R.; Dirks, R. M.; and Pierce, N. A. 2011. NUPACK: Analysis and design of nucleic acid systems. *Journal of computational chemistry*, 32(1): 170–173.

Zuker, M.; and Stiegler, P. 1981. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1): 133–148.