Optimizing Discrete Spaces via Expensive Evaluations: A Learning to Search Framework

Aryan Deshwal[1], **Syrine Belakaria**[1], **Janardhan Rao Doppa**[1], **Alan Fern**[2]
[1]School of EECS, Washington State University
[2]School of EECS, Oregon State University
{aryan.deshwal, syrine.belakaria, jana.doppa}@wsu.edu, alan.fern@oregonstate.edu

Abstract

We consider the problem of optimizing expensive black-box functions over discrete spaces (e.g., sets, sequences, graphs). The key challenge is to select a sequence of combinatorial structures to evaluate, in order to identify high-performing structures as quickly as possible. Our main contribution is to introduce and evaluate a new learning-to-search framework for this problem called L2S-DISCO. The key insight is to employ search procedures guided by control knowledge at each step to select the next structure and to improve the control knowledge as new function evaluations are observed. We provide a concrete instantiation of L2S-DISCO for local search procedure and empirically evaluate it on diverse real-world benchmarks. Results show the efficacy of L2S-DISCO over state-of-the-art algorithms in solving complex optimization problems.

1 Introduction

Many scientific and engineering applications involve optimizing discrete spaces (e.g., sets, sequences, graphs) guided by expensive black-box function evaluations. For example, in the application of finding alloys with high creep-resistance, we need to search over subsets of a given set of candidate metals guided by physical lab experiments. Similarly, for designing application-specific integrated circuits, we need to search over candidate placements of processing elements and communication links to optimize performance as measured by expensive computational simulations.

There is very limited work on optimizing discrete spaces via expensive evaluations as discussed in Section 3. A popular and effective framework for optimizing expensive functions is Bayesian optimization (BO) (Shahriari et al. 2016). The key idea behind BO is to estimate a cheap surrogate model, e.g., a Gaussian Process (Rasmussen and Williams 2006), based on observed outcomes, which can be used as guidance for intelligently selecting the next evaluation points. Despite the huge successes of BO (Snoek, Larochelle, and Adams 2012; Thornton et al. 2013), current approaches focus primarily on continuous optimization spaces and there is little principled work on discrete spaces. The first challenge in moving from continuous spaces to discrete spaces is to define an effective surrogate model over combinatorial structures. The second challenge is, given such a surrogate model, to search through the combinatorial space to identify the most promising next structure to evaluate. Prior methods either employ relatively simple surrogate models that admit tractable optimization solvers for this search or use complex models with highly-heuristic search methods. Ideally, we would like an approach that can work with more complex models while following a more principled and effective search approach.

In this paper, we introduce a new *learning-to-search (L2S) framework*, called L2S-DISCO, for selecting the sequence of combinatorial structures to evaluate during optimization. L2S-DISCO employs a combinatorial search procedure (e.g., local search with multiple restarts) guided by search control knowledge (e.g., heuristic function to select good starting states), and *continuously* improves the control knowledge using machine learning. Our approach can be viewed as online learning for hyper-heuristic search (Burke et al. 2013). From this perspective, a primary contribution of this paper is the first application of hyper-heuristic search to BO.

Our search-based perspective allows us to directly tune the search via learning during the optimization process and has several potential advantages: 1) High flexibility in defining search spaces over structures; 2) Easily handles domain constraints to search over "valid" structures; 3) Allows the incorporation of prior knowledge; and 4) Puts forth a new family of BO-style approaches with many future instantiations to explore in future work. We provide a concrete instantiation of L2S-DISCO for local search based optimization by specifying the form of training data, and a rank learning formulation to update the search heuristic for selecting promising starting states. Experimental results on diverse benchmarks show the efficacy of L2S-DISCO on complex real-world problems.

Contributions. The key contributions of this paper include:

- A novel learning-to-search framework, L2S-DISCO, for optimizing expensive functions over discrete spaces, which integrates combinatorial-search with machine learning.
- A concrete instantiation of L2S-DISCO for local-search–based optimization.
- An evaluation of L2S-DISCO over diverse real-world benchmarks, showing advantages over the state-of-the-art.

2 Problem Setup and Challenges

Combinatorial space of structures. Let $\mathcal{X}$ be a combinatorial space of objects to be optimized over, where each element $x \in \mathcal{X}$ is a discrete structure (e.g., set, sequence, graph). Without loss of generality, let each candidate structure $x \in \mathcal{X}$ be represented using d discrete variables $v_1, v_2, \cdots, v_d$, where each variable v_i take candidate values from a set $C(v_i)$. If each discrete variable takes k candidate values, the size of the combinatorial space is $\mathcal{O}(k^d)$.

Problem definition. We are given a combinatorial space of structures $\mathcal{X}$. We assume an unknown real-valued objective function $\mathcal{F} : \mathcal{X} \mapsto \Re$, which can evaluate each candidate structure $x \in \mathcal{X}$. For example, in alloys optimization application, x is a set corresponding to material design and $\mathcal{F}(x)$ corresponds to running a physical lab experiment using additive manufacturing techniques. Conducting an experiment produces an evaluation $y = \mathcal{F}(x)$ and is expensive in terms of the consumed resources. The main goal is to find a structure $x \in \mathcal{X}$ that approximately optimizes $\mathcal{F}$ by conducting a limited number of evaluations and observing their outcomes.

Bayesian optimization formulation and challenges. Bayesian optimization (BO) methods (Shahriari et al. 2016) build a surrogate *statistical model* $\mathcal{M}$, e.g., Gaussian Process (GP), from the training data of past function evaluations and employ it to sequentially select a sequence of inputs for evaluation to solve the problem (see Algorithm 1). The selection of inputs is performed by optimizing an *acquisition function* $\mathcal{AF}$ that is parameterized by the current model $\mathcal{M}$ and input $x \in \mathcal{X}$ to score the utility of candidate inputs for evaluation. Some example acquisition functions include expected improvement (EI) (Jones, Schonlau, and Welch 1998) and upper-confidence bound (UCB) (Srinivas et al. 2010). BO methods are mostly studied for continuous spaces $\mathcal{X} \subset \Re^d$. There is very limited work on BO methods to optimize discrete spaces (as discussed in the related work section). There are two key challenges in using BO for discrete spaces.

1. *Surrogate statistical modeling.* GPs are the popular choice for building statistical models in BO over continuous spaces. To handle discrete structures, we need an appropriate kernel that can compute the similarity between any pair of candidate structures $x, x' \in \mathcal{X}$. One choice is to leverage the general recursive convolution framework (Haussler 1999). The key idea is to recursively decompose the structured object into *atomic* sub-structures and define valid local kernels between them. For example, random walk kernels over graphs defined in terms of paths (Vishwanathan et al. 2010). Random forest (RF) models can be used as an alternate generic choice to handle discrete spaces. In this work, we employ RF models as part of our experiments.
2. *Acquisition function optimization.* In each iteration of BO, we need to solve the following optimization problem to select the next candidate structure for evaluation.

$$x_{next} = \arg\max_{x \in \mathcal{X}} \mathcal{AF}(\mathcal{M}, x) \tag{1}$$

The key challenge for discrete spaces is that, Equation 1 corresponds to solving a general combinatorial optimization problem. The effectiveness of BO critically depends on the accuracy of solving this optimization problem. In this paper, our main focus is on addressing this challenge (line 4 in Algorithm 1 given below) using a novel learning to search framework.

Algorithm 1 Bayesian Optimization framework

Input: $\mathcal{X}$ = Discrete space, $\mathcal{F}(x)$ = expensive objective function
Output: $\hat{x}_{best}$, the best uncovered structure from $\mathcal{X}$

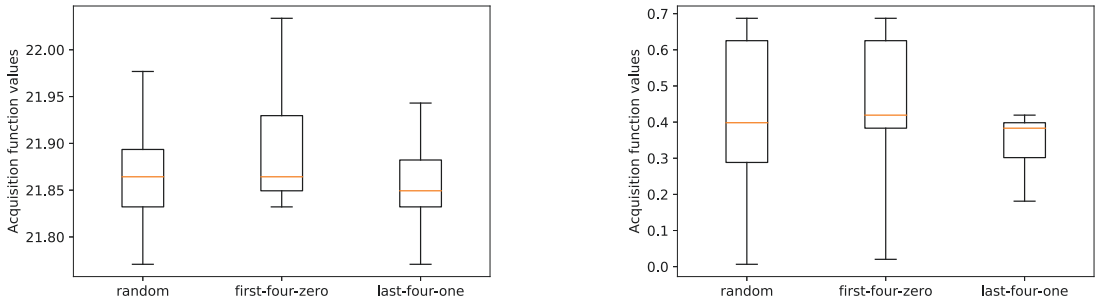
1: Initialize $\mathcal{D}_0 \leftarrow$ small number of input-output pairs; and $t \leftarrow 0$
2: **repeat**
3: Learn the model: $\mathcal{M}_t \leftarrow$ LEARN($\mathcal{D}_t$)
4: Compute the next structure to evaluate via acquisition function optimization: $x_{t+1} \leftarrow \arg max_{x \in \mathcal{X}} \mathcal{AF}(\mathcal{M}_t, x)$
5: Evaluate objective function $\mathcal{F}(x)$ at x_{t+1} to get y_{t+1}
6: Aggregate the data: $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(x_{t+1}, y_{t+1})\}$
7: $t \leftarrow t + 1$
8: **until** convergence or maximum iterations
9: $\hat{x}_{best} \leftarrow \arg max_{x_t \in \mathcal{D}} y_t$
10: **return** the best uncovered structure $\hat{x}_{best}$

3 Related Work

There is very limited work on BO over discrete spaces. SMAC (Hutter, Hoos, and Leyton-Brown 2010; 2011) is one canonical baseline which employs random forest as surrogate model and a *hand-designed* local search procedure for optimizing the acquisition function. BOCS (Baptista and Poloczek 2018) is a state-of-the-art method that employs a linear Bayesian model defined over binary variables as the surrogate model. The model is described as:

$$f_\alpha(x \in \mathcal{X}) = \alpha_0 + \sum_j \alpha_j v_j + \sum_{i,j>i} \alpha_{ij} v_i v_j \tag{2}$$

where $\mathcal{X} = \{0,1\}^d$. The α variables drawn from a sparse prior, quantify the uncertainty of the model. This linear model formulation over binary variables along with the usage of Thompson sampling as the acquisition function allows the acquisition function optimization in BOCS to be amenable to a semi-definite programming (SDP) solution. However, BOCS has several drawbacks. First, the simple model with second-order interactions may not suffice for optimization problems with complex interactions. Additionally, the model is very specific to binary variables. An extension to general categorical variables via one-hot encoding was provided in the supplementary section (Baptista and Poloczek 2018), but this results in significant growth of input dimensions. Second, the SDP based acquisition function optimization solution is very specific to the case of binary variables and Thompson sampling, which severely limits its applicability. Additionally, one-hot encoding to handle categorical variables leads to poor scalability and loss of accuracy for BOCS. Third, the SDP based solver cannot handle complex constraints to select only valid structures. Indeed, we observe these shortcomings of BOCS in our experiments. In contrast, our proposed method can work with any choice of statistical model and acquisition function, and uses advances in machine learning to tune search-based acquisition function optimizers on-the-fly to



(a) Contamination domain with UCB acquisition function.

(b) Ising domain with EI acquisition function.

Figure 1: Empirical evidence to show how learning can be useful to solve acquisition function optimization. Boxplot shows final acquisition function values resulting from 100 runs of local search based optimization with three different restart strategies.

improve its accuracy in selecting candidate structures for evaluation.

There is also work on solving BO over discrete spaces by reduction to continuous BO (Gómez-Bombarelli et al. 2018). The key idea is to employ an encoder-decoder architecture to learn continuous representation from data and perform BO in this latent space. The main drawback of this method is that it generates a large fraction of invalid structures. This approach also requires a large database of "relevant" structures, for training an auto-encoder, which will not be available in many applications, where *small data* is the norm.

The challenge of optimizing acquisition function for continuous input spaces was tackled in previous work (Wilson, Hutter, and Deisenroth 2018). Since this approach relies on gradients for optimizing the acquisition function, it is specific to continuous spaces and cannot be generalized to the challenging case of discrete spaces. A tangential line of work (Volpp et al. 2019; Perrone et al. 2019) exploiting the idea of learning acquisition function strategies across multiple tasks has also been explored in the context of transfer learning for black-box optimization in the BO framework. However, our problem setting is very different as we focus on learning within a single task when we have not yet solved the task.

4 Learning to Search Framework

In this section, we first motivate learning-to-search (L2S) methods for solving acquisition function optimization (AFO) problems. Subsequently, we describe our proposed learning to search framework, L2S-DISCO, and provide a concrete instantiation for local-search based AFO.

4.1 Motivation

Search-based AFO solvers. In a search-based optimizer, the overall problem-solving can be modeled as a computational search process defined in terms of an appropriate *search space* over candidate solutions, *search procedure* to uncover solutions, and *search control knowledge* to guide the search. For example, a solver, based on local search with multiple restarts, may use control knowledge that biases the restart distribution. Similarly, a solver, based on branch-and-bound search, may use control knowledge corresponding to policies for node expansion and pruning based on the current state of the solver. An important aspect of search-based optimization is that we can potentially improve the search control knowledge during a search by feeding the information about the search progress to machine learning techniques.

Relation between AFO problems. We now give the intuition for why it may be useful to learn control knowledge across the sequence of AFO problems encountered during BO. Recall that the change in acquisition function $\mathcal{AF}(\mathcal{M}, x)$ from iteration i to $i+1$ is due to *only one* new training example (x_i, y_i), where x_i is the selected structure in iteration i and y_i is its function evaluation. Intuitively, even if the acquisition function scores of candidate structures in $\mathcal{X}$ are changing, the search control knowledge can still guide the search towards promising structures and only require small modifications to account for the slight change in the AFO problem from previous BO iteration. This motivates using machine learning to adapt the knowledge in a way that generalizes from prior iterations of AFO to future AFO iterations.

Empirical evidence for the utility of learning. We now provide some empirical evidence on real-world problems to show how machine learning can be potentially useful to improve the accuracy of solving AFO problems. We consider local search with multiple restarts as the AFO solver. In this case, the AFO solver takes as input the objective function $\mathcal{AF}(\mathcal{M}, x)$ and restarting strategy, and returns the local optima $\hat{x} \in \mathcal{X}$ with associated acquisition function value $\mathcal{AF}(\mathcal{M}, \hat{x})$. The accuracy of local search based AFO solver critically depends on the restart strategy. We performed local search based AF optimization using three different restart strategies on optimization problems with binary discrete variables: 1. Completely random (*random*); 2. Assigning the first four discrete variables as zero and remaining randomly (*first-four-zero*); and 3. Assigning the last four discrete variables as one and remaining randomly (*last-four-one*). In figure 1, we show the results of solving a single AFO problem using these three different restart strategies over 100 runs. We plot the distribution of $\mathcal{AF}(\mathcal{M}, \hat{x})$ for these strategies. We can see that different restart strategies give varied solutions (em-

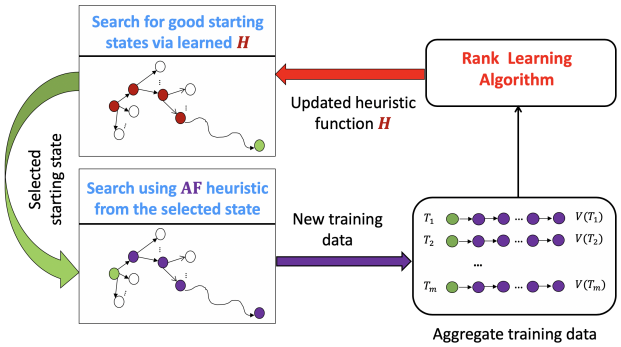


Figure 2: High-level overview of L2S-DISCO instantiation for local search. It repeatedly performs three steps. First, run local search from a random state guided by current heuristic $\mathcal{H}$ to select a good starting state. Second, run local search from this selected starting state guided by acquisition function ($\mathcal{AF}$). Third, use new training data in the form of local search trajectory T and acquisition function value of the local optima $V(T)$ to update the heuristic $\mathcal{H}$ via rank learning.

pirically). This observation can be leveraged to learn a search heuristic to select promising starting states for local search using the training data from local search trajectories.

4.2 L2S-DISCO and Key Elements

L2S-DISCO integrates machine learning techniques and combinatorial search in a principled manner for accurately solving AFO problems to select combinatorial structures for evaluation. This framework allows us to employ surrogate statistical models of arbitrary complexity and can work with any acquisition function. The key insight behind L2S-DISCO is to directly tune the search via learning during the optimization process to select the next structure for evaluation. The search-based perspective has several advantages: 1) High flexibility in defining search spaces over structures; 2) Easily handles domain constraints that determine which structures are "valid". For example, when designing an optimized network on the chip to facilitate data transfer between multiple cores, we need to make sure that there is a viable path between any pair of cores; 3) Allows to incorporate prior knowledge in the form of heuristic rules to explore promising regions of the search space; and 4) Provides additional points for learning within the search framework to improve the effectiveness of search in uncovering better structures.

Overview of L2S-DISCO. We build a surrogate model $\mathcal{M}$ using a small number of experiments and their outcomes to guide our search process to select the sequence of combinatorial structures to perform experiments. L2S-DISCO is parameterized by a search space $\mathcal{S}$ over structures, a learned function $\mathcal{AF}(\mathcal{M}, x \in \mathcal{X})$ to score the utility of structures for evaluation, a search strategy $\mathcal{A}$ (e.g., local search), and a learned search control knowledge $\mathcal{H}$ to guide the search towards high-scoring structures. In each BO iteration, we perform the following two steps repeatedly until the maximum time-bound is exceeded or a termination criteria is met. **Step 1:** Execute search strategy $\mathcal{A}$ guided by the current search control knowledge to uncover promising structures. **Step 2:** Update the parameters of search control knowledge $\mathcal{H}$ using the online training data generated from the recent search experience. Fig 2 illustrates the instantiation of L2S-DISCO for local search. Each structure $x \in \mathcal{X}$ uncovered during the entire search is scored according to $\mathcal{AF}(\mathcal{M}, x)$ and we select the highest scoring structure x_{next} for function evaluation. We perform experiment using the selected structure x_{next} and observe the outcome $\mathcal{F}(x_{next})$. The statistical model $\mathcal{M}$ is updated using the new training example $(x_{next}, \mathcal{F}(x_{next}))$. We repeat the next iteration of BO via L2S-DISCO initialized with the current search control knowledge.

Key Elements. There are two key elements in L2S-DISCO that need to be specified to instantiate it for a given search procedure. **1)** The form of training data to learn search control knowledge $\mathcal{H}$; and **2)** The learning formulation and associate learning algorithm to update the parameters of search control knowledge $\mathcal{H}$ using online training data. These elements vary for different search procedures and forms of search control knowledge. We provide a high-level example to illustrate these elements for branch-and-bound search.

Branch-and-bound search is a widely used search procedure to solve combinatorial optimization problems. It employs a search space over partial structures, where each state corresponds to partial assignment of variables. The states with complete assignment for all variables are referred as terminals. Variable selection strategy for successive assignment is one of the main components of branch-and-bound search. Therefore, $\mathcal{H}$ corresponds to the policy that selects the variable on which to branch on for the next assignment. In this case, the training data is generated by the trajectories obtained by a strong branching (SB) strategy (Khalil et al. 2016) which exhaustively tests each variable for assignment. A learning-to-rank formulation is natural for inducing the variable selection policy, since the reference strategy (SB) effectively ranks variables at a node by a score, and picks the highest-scoring variable, i.e., the score itself is not important.

Below we provide a concrete instantiation of L2S-DISCO for local search based acquistion function optimization that will be employed for our empirical evaluation.

4.3 Instantiation of L2S-DISCO for Local Search

Recall that local search based AFO solver performs multiple runs of local search guided by the acquisition function $\mathcal{AF}(\mathcal{M}, x)$ from different random starting states. The search space is defined over complete structures, where each state corresponds to a complete structure $x \in \mathcal{X}$. The successors of a state with structure x referred as $\mathcal{N}(x)$, is the set of all structures $x' \in \mathcal{X}$ such that the hamming distance between x and x' is one. The effectiveness of local search depends critically on the quality of starting states. Therefore, we instantiate L2S-DISCO for local search and learn a search heuristic $\mathcal{H}(\theta, x)$ to select good starting states that will allow local search to uncover high-scoring structures from $\mathcal{X}$ according to $\mathcal{AF}(\mathcal{M}, x)$.

To instantiate L2S-DISCO for local search, we need to specify the two key elements: 1) The training data for learning

the heuristic $\mathcal{H}(\theta, x)$?; and 2) The learning formulation to induce $\mathcal{H}(\theta, x)$ from online training data.

1) Training data. The set of search trajectories $\mathcal{T}$ obtained by performing local search from different starting states and acquisition function scores for local optima correspond to the training data. Each search trajectory $T \in \mathcal{T}$ consists of the sequence of states from the starting state x_{start} to the local optima x_{end}. Suppose $V(T)$=$\mathcal{AF}(\mathcal{M}, x_{end})$ represents the acquisition function score of the local optima for local search trajectory T.

2) Rank learning formulation. The role of the heuristic $\mathcal{H}(\theta, x)$ is to rank candidate starting states according to their utility in uncovering high-scoring structures from $\mathcal{X}$ via local search. Recall that if we perform local search guided by $\mathcal{AF}(\mathcal{M}, x)$ from any state x on a search trajectory $T \in \mathcal{T}$, we will reach the same local optima with acquisition function score $V(T)$. In other words, every state on the trajectory $T \in \mathcal{T}$ has the same utility. Therefore, we formulate the problem of learning the search heuristic as an instance of bipartite ranking (Agarwal and Roth 2005). Specifically, for every pair of search trajectories $T_1, T_2 \in \mathcal{T}$, if $V(T_1) > V(T_2)$, then we want to rank every state on the trajectory T_1 better than every state on the trajectory T_2. We will generate one ranking example for every pair of states (x_1, x_2), where x_1 is a state on the trajectory T_1 and x_2 is a state on the trajectory T_2. The aggregate set of ranking examples are given to an off-the-shelf rank learner to induce $\mathcal{H}(\theta, x)$, where θ are the parameters of the ranking function. In our experiments, we employed RankNet (Burges et al. 2005) as the base rank learner. We leveraged existing code[1] for our purpose.

[1]https://github.com/shiba24/learning2rank

Algorithm 2 L2S-DISCO for local search

Input: $\mathcal{X}$= space of combinatorial structures, $\mathcal{AF}(\mathcal{M}, x)$= acquisition function, $\mathcal{H}(\theta, x)$= search heuristic from previous BO iteration, RANKLEARN= rank learner

Output: $\hat{x}_{next}$, the selected structure for function evaluation

Initialization: $\mathcal{T} \leftarrow \emptyset$ (training data of local search trajectories) and $\mathcal{S}_{start} \leftarrow \emptyset$ (set of starting states)
repeat
  Perform local search from a random state $x \in \mathcal{X}$ guided by heuristic $\mathcal{H}(\theta, x)$ to reach a local optima $x_{restart}$
  if $x_{restart} \in \mathcal{S}_{start}$ **then**
    $x_{start} \leftarrow$ random structure from $\mathcal{X}$
  else
    $x_{start} \leftarrow x_{restart}$
  end if
  Perform local search from x_{start} guided by $\mathcal{AF}(\mathcal{M}, x)$
  Add the new search trajectory and $\mathcal{AF}(\mathcal{M}, x_{end})$ to $\mathcal{T}$
  Update heuristic $\mathcal{H}(\theta, x)$ via rank learner using $\mathcal{T}$
  $\mathcal{S}_{start} \leftarrow \mathcal{S}_{start} \cup x_{start}$
until convergence or maximum iterations
$\hat{x}_{next} \leftarrow$ best scoring structure as per $\mathcal{AF}(\mathcal{M}, x)$ found during the entire search process
return the selected structure for evaluation $\hat{x}_{next}$

L2S-DISCO for local search based optimization. Figure 2 illustrates L2S-DISCO instantiation for local search based acquisition function optimization. At a high-level, each iteration of L2S-DISCO consists of two alternating local search runs. First, local search guided by heuristic $\mathcal{H}$ to select the starting state. Second, local search guided by $\mathcal{AF}$ from the selected starting state. After each local search run, we get a new local search trajectory, and the heuristic function $\mathcal{H}$ is updated to be consistent with this new search trajectory.

Algorithm 2 shows the pseduo-code for learning based local search to solve AFO problems arising in BO iterations. It reuses the learned search heuristic from the previous BO iteration and updates it in an online manner using the new training data generated during AF optimization. In each iteration, we perform the following sequence of steps. First, we perform local search from a random state guided by the search heuristic $\mathcal{H}(\theta, x)$ until reaching the local optima $x_{restart}$ to select the next starting state. Second, if $x_{restart}$ was not explored as a starting state in previous local search iterations, we select $x_{restart}$ as the starting state to perform local search guided by $\mathcal{AF}(\mathcal{M}, x)$ and add the local search trajectory to our training data. Third, we update the search heuristic $\mathcal{H}(\theta, x)$ using the newly added training example via rank learner. We repeat the above three steps until convergence or maximum iterations. This instantiation of L2S-DISCO is similar in spirit to the STAGE algorithm (Boyan and Moore 2000). At the end, we return the best scoring structure uncovered during the search $\hat{x}_{next}$ for function evaluation.

5 Experiments and Results

In this section, we first describe our experimental setup and then discuss the results of L2S-DISCO and baseline methods.

5.1 Experimental Setup

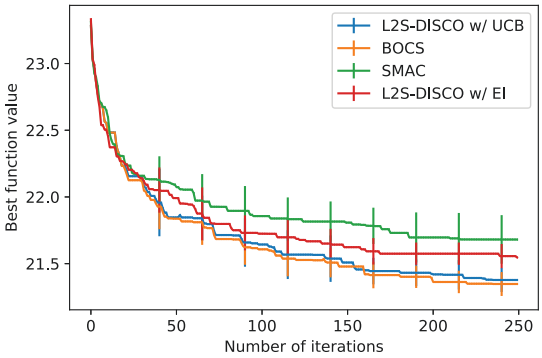
Benchmark Domains. We employ five diverse benchmark domains for our empirical evaluation.

1. Contamination. The problem considers a food supply with d stages, where a binary {0,1} decision must be made at each stage to prevent the food from being contaminated with pathogenic micro-organisms (Hu et al. 2010; Baptista and Poloczek 2018). Each prevention effort at stage i can be made to decrease the contamination by a given random rate Γ_i and incurring a cost c_i. The contamination spreads with a random rate Λ_i if no prevention effort is taken. The overall goal is to ensure that the fraction of contaminated food at each stage i does not exceed an upper limit U_i with probability at least $1-\epsilon$ while minimizing the total cost of all prevention efforts. Following (Baptista and Poloczek 2018), the lagrangian relaxation based problem formulation is:

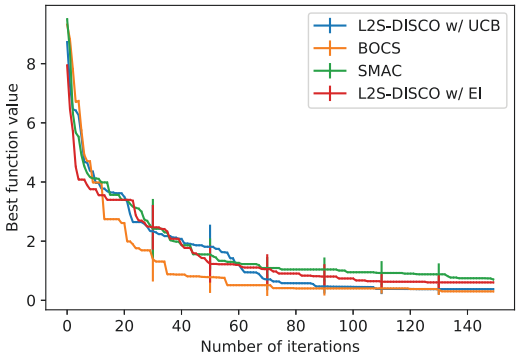
$$\arg\min_{x} \sum_{i=1}^{d} \left[c_i x_i + \frac{\rho}{T} \sum_{k=1}^{T} 1_{\{Z_k > U_i\}} \right] + \lambda \|x\|_1$$

where λ is a regularization coefficient, Z_i is the fraction of contaminated food at stage i, violation penalty coefficient ρ=1, and T=100.

2. Sparsification of zero-field Ising models. The distribution of a zero field Ising model $p(z)$ for $z \in \{-1, 1\}^n$ is



(a) Contamination domain with no. of stages $d = 25$ and $\lambda = 10^{-4}$ over 250 iterations.



(b) Ising domain with number of nodes $d = 24$ and $\lambda = 10^{-2}$ over 150 iterations.

Figure 3: Results for contamination and ising domain (**minimization**).

characterized by a symmetric interaction matrix J^p whose support is represented by a graph $G^p = ([n], E^p)$ that satisfies $(i, j) \in E^p$ if and only if $J^p_{ij} \neq 0$ holds (Baptista and Poloczek 2018). The overall goal is to find a close approximate distribution $q(z)$ while minimizing the number of edges in E^q. Therefore, the objective function in this case is a regularized KL-divergence between p and q as given below:

$$D_{KL}(p||q_x) = \sum_{(i,j)\in E^p} (J^p_{ij} - J^q_{ij}) E_p[z_i z_j] + log(Z_q/Z_p)$$

where Z_q and Z_p are partition functions corresponding to p and q respectively, and $x \in \{0, 1\}^{E^q}$ is the decision variable representing whether each edge is present in E^q or not.

3. Low auto-correlation binary sequences (LABS). The problem is to find a binary {+1,-1} sequence $S = (s_1, s_2, \cdots, s_n)$ of given length n that maximizes *merit factor* defined over a binary sequence as given below:

$$\text{Merit Factor(S)} = \frac{n^2}{E(S)}$$

$$\text{where } E(S) = \sum_{k=1}^{n-1} \left(\sum_{i=1}^{n-k} s_i s_{i+k} \right)^2$$

The LABS problem has multiple applications in diverse scientific disciplines (Packebusch and Mertens 2015).

4. Network optimization in multicore chips. With Moore's law aging quickly, multicore architectures are considered very promising for parallel computing (Ceze, Hill, and Wenisch 2016). A key challenge in multicore research is to reduce the performance bottleneck due to data movement. One promising solution is to optimize the placement of communication links between cores to facilitate efficient data transfer. This optimization is typically guided by expensive simulators that mimics the real hardware. The network optimization problem is part of the rodinia benchmark (Che et al. 2009) and uses the gem5-GPU simulator (Power et al. 2014). There are 12 cores whose placements are fixed and the goal is to place 17 links between them to optimize performance: *66 binary variables*. There is one *constraint* to determine valid structures: existence of a viable path between any pair of cores. We report the performance improvement with respect to the provided baseline network.

5. Core placement optimization in multicore chips. This is another multicore architecture optimization problem from rodinia benchmark (Che et al. 2009). In this problem, we are given 64 cores of three types (8 CPUs, 40 GPUs, and 16 memory units) and they are connected by a mesh network (every core is connected to its four neighboring cores) to facilitate data transfer. The goal is to place the three types of cores to optimize performance: *64 categorical variables* with each taking three candidate values. We need to make sure that the *cardinality constraints* in terms of the number of cores of each type are satisfied. We report the performance improvement w.r.t the provided baseline placement.

Baseline Methods. We compare the local search instantiation of L2S-DISCO with two state-of-the-art methods: SMAC (Hutter, Hoos, and Leyton-Brown 2011) and BOCS (Baptista and Poloczek 2018). We employed open-source python implementations of both BOCS [2] and SMAC [3]. Since SMAC implementation does not support handling domain constraints to search over valid structures[4], we could not run SMAC for network optimization and core placement optimization benchmarks. Similarly, SDP based solver for BOCS cannot handle constraints, so we employed simulated annealing based solver available in the BOCS code for those two benchmarks. We initialize the surrogate of all the methods by evaluating 20 random structures. For L2S-DISCO. we employed random forest model with 20 trees (tried two standard settings of scikit-learn library, namely, 10 and 20 trees, and got similar results) and two different acquisition functions (EI and UCB). For UCB, we use the adaptive rate recommended by (Srinivas et al. 2010) to set the exploration and exploitation trade-off parameter β_i value depending on the iteration num-

[2] https://github.com/baptistar/BOCS
[3] https://github.com/automl/SMAC3
[4] https://github.com/automl/SMAC3/issues/403

ber i. We ran L2S-DISCO (Algorithm 2) for a maximum of 60 iterations.

Evaluation Metric. We use the best function value achieved after a given number of iterations as a metric to evaluate all methods: SMAC, BOCS, and L2S-DISCO. The method that uncovers high-performing structures with less number of function evaluations is considered better. LABS is a maximization problem, but the remaining four benchmarks require the objective to be minimized. We use the total number of iterations similar to BOCS (Baptista and Poloczek 2018).

5.2 Results and Discussion

We discuss the results of L2S-DISCO and baseline methods on the five benchmarks below. All the reported results are averaged over 10 random runs (except for BOCS in cores placement optimization due to its poor scalability).

Contamination and Ising. Figure 3 shows the comparison of L2S-DISCO with SMAC and BOCS baselines. We make the following observations. 1) Both L2S-DISCO variants that use EI and UCB acquisition functions perform better than SMAC. 2) L2S-DISCO with UCB performs better than the variant with EI. We observed a similar trend for the remaining three benchmarks also. Therefore, to avoid clutter, we only show the results of L2S-DISCO with UCB for the remaining benchmarks. 3) Results of L2S-DISCO are comparable to BOCS on the contamination problem. However, BOCS has a better anytime profile for ising domain. L2S-DISCO eventually matches the performance of BOCS after 90 iterations. The main reason BOCS performs slightly better in these two domains is that they exactly match the modeling assumptions of BOCS, which allows the use of SDP based solver to select structures for evaluation. Below we will show how the performance of BOCS degrades when the assumptions are not met, whereas L2S-DISCO peforms robustly across optimization problems of varying complexity.

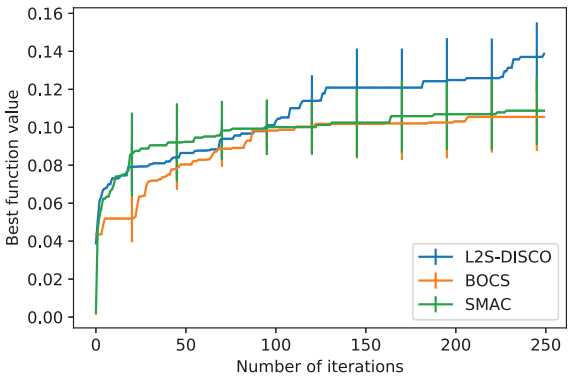


Figure 4: Results for LABS domain (**maximization**) with input sequence length n=30 over 250 iterations.

LABS. Figure 4 shows the comparison of L2S-DISCO with SMAC and BOCS baselines. We can see that L2S-DISCO clearly outperforms both BOCS and SMAC on this domain. BOCS has the advantage of SDP based solver, but its statistical model that accounts for only pair-wise interactions is limiting to account for the complexity in this problem. SMAC and L2S-DISCO both employ random forest model, but L2S-DISCO does better in terms of acquisition function optimization by integrating learning with search.

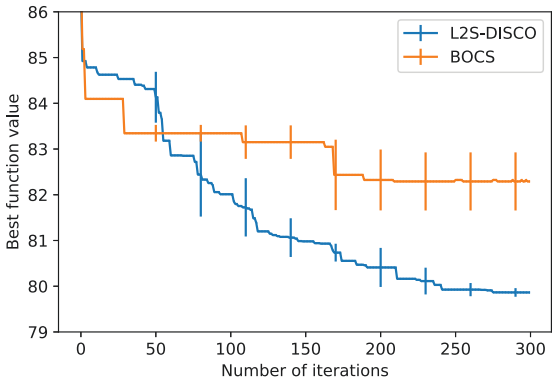


Figure 5: Results for network optimization in multicore chips (**minimization**) over 300 iterations.

Network optimization in multicore chips. As mentioned earlier, we could not run SMAC for this problem as SMAC library does not allow to incorporate complex domain constraints. The SDP solver of BOCS is also not applicable due to complex constraints. Hence, we employ simulated annealing based solver for acquisition function optimization. Figure 5 shows the comparison of L2S-DISCO with BOCS baseline. We can see that L2S-DISCO performs significantly better than BOCS in this domain. BOCS seems to get stuck for long periods, whereas L2S-DISCO shows consistent improvement in uncovering high-performing structures. This behavior of BOCS can be partly attributed to the limitations of both surrogate model and acquisition function optimizer.

Core placement optimization in multicore chips. Figure 5 shows the comparison of L2S-DISCO with BOCS. L2S-DISCO significantly outperforms BOCS on this benchmark also. Additionally, BOCS scales poorly on this domain, where the discrete variables are non-binary. Recall that BOCS model was developed for binary variables and authors suggested the use of one-hot encoding to handle categorical variables. However, this transformation excessively increases the no. of dimensions. For example, we have 64 dimensions for L2S-DISCO, but it grows to 192 for BOCS due to one-hot encoding and makes its execution extremely slow. BOCS took one hour per single BO iteration on a machine with Intel Xeon(R) 2.5Ghz CPU and 96 GB memory. This is the main reason we could only perform one run of BOCS.

6 Summary and Future Work

We introduced the L2S-DISCO framework that integrates machine learning with search-based optimization for optimizing expensive black-box functions over discrete spaces.

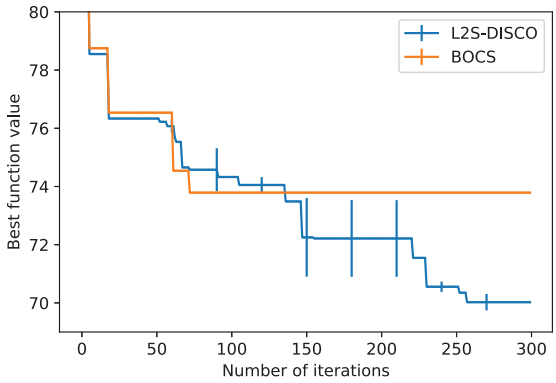


Figure 6: Results for core placement optimization in multi-core chips (**minimization**) over 300 iterations.

We showed that instantiation of L2S-DISCO for local search based optimization yields significantly better performance than state-of-the-art methods on complex optimization problems. Future work includes studying instantiations of L2S-DISCO for other search procedures to further improve the performance, and applying L2-DISCO on important real-world applications by leveraging domain knowledge.

Acknowledgements. We thank the anonymous reviewers for their feedback. This research is supported by National Science Foundation grants IIS-1845922, OAC-1910213, and IIS-1619433.

References

Agarwal, S., and Roth, D. 2005. Learnability of bipartite ranking functions. In *Proceedings of Annual Conference on Learning Theory (COLT)*, 16–31.

Baptista, R., and Poloczek, M. 2018. Bayesian optimization of combinatorial structures. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 462–471.

Boyan, J., and Moore, A. W. 2000. Learning evaluation functions to improve optimization by local search. *Journal of Machine Learning Research* 1(Nov):77–112.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. N. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning (ICML)*, 89–96.

Burke, E. K.; Gendreau, M.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; and Qu, R. 2013. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society* 64(12):1695–1724.

Ceze, L.; Hill, M. D.; and Wenisch, T. F. 2016. Arch2030: A vision of computer architecture research over the next 15 years. *CoRR* abs/1612.03182.

Che, S.; Boyer, M.; Meng, J.; Tarjan, D.; Sheaffer, J. W.; Lee, S.; and Skadron, K. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, 44–54.

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.

Haussler, D. 1999. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz.

Hu, Y.; Hu, J.; Xu, Y.; Wang, F.; and Cao, R. Z. 2010. Contamination control in food supply chain. In *Proceedings of the Winter Simulation Conference*, WSC '10, 2678–2681.

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2010. Sequential model-based optimization for general algorithm configuration (extended version). Technical Report TR-2010-10, University of British Columbia, Department of Computer Science. Available online: http://www.cs.ubc.ca/~hutter/papers/10-TR-SMAC.pdf.

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.

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

Khalil, E. B.; Le Bodic, P.; Song, L.; Nemhauser, G.; and Dilkina, B. 2016. Learning to branch in mixed integer programming. In *Proceedings of 30th AAAI Conference on Artificial Intelligence*.

Packebusch, T., and Mertens, S. 2015. Low autocorrelation binary sequences. *Journal of Physics A: Mathematical and Theoretical, 49 (2016) 165001*.

Perrone, V.; Shen, H.; Seeger, M.; Archambeau, C.; and Jenatton, R. 2019. Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning.

Power, J.; Hestness, J.; Orr, M.; Hill, M.; and Wood, D. 2014. gem5-gpu: A heterogeneous cpu-gpu simulator. *Computer Architecture Letters* 13(1).

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

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

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 26th Annual Conference on Advances in Neural Information Processing Systems.*, 2960–2968.

Srinivas, N.; Krause, A.; Kakade, S.; and Seeger, M. W. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 1015–1022.

Thornton, C.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph kernels. *Journal of Machine Learning Research* 11(Apr):1201–1242.

Volpp, M.; Fröhlich, L.; Fischer, K.; Doerr, A.; Falkner, S.; Hutter, F.; and Daniel, C. 2019. Meta-learning acquisition functions for transfer learning in bayesian optimization.

Wilson, J.; Hutter, F.; and Deisenroth, M. 2018. Maximizing acquisition functions for bayesian optimization. In *Advances in Neural Information Processing Systems*, 9884–9895.