Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

Bayesian Optimization with Resource Constraints and Production

Nima Dolatnia
Department of Statistics
Oregon State University
Corvallis, OR, USA

Alan Fern and **Xiaoli Fern**
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR, USA

Abstract

In this paper, we aim to take a step toward a tighter integration of automated planning and Bayesian Optimization (BO). BO is an approach for optimizing costly-to-evaluate functions by selecting a limited number of experiments that each evaluate the function at a specified input. Typical BO formulations assume that experiments are selected one at a time, or in fixed batches, and that experiments can be executed immediately upon request. This setup fails to capture many real-world domains where the execution of an experiment requires setup and preparation time. In this paper, we define a novel BO problem formulation that models the resources and activities needed to prepare and run experiments. We then present a planning approach, based on finite-horizon tree search, for scheduling the potentially concurrent experimental activities with the aim of best optimizing the function within a limited time horizon. A key element of the approach is a novel state evaluation function for evaluating leaves of the search tree, for which we prove approximate guarantees. We evaluate the approach on a number of diverse benchmark problems and show that it produces high-quality results compared to a number of natural baselines.

Introduction

We consider optimizing an unknown function f by running experiments that each take an input x and return a noisy output $f(x)$. In particular, we focus on the setting where experiments are expensive, limiting the number of experiments that can be run. Bayesian Optimization (BO) addresses this setting by maintaining a Bayesian posterior over f to capture uncertainty given prior experiments (Jones 2001; Brochu, Cora, and de Freitas 2010). The posterior is then used to select new experiments that trade-off exploring uncertain experiments and exploiting promising areas.

BO frameworks traditionally assume that experiments can be run immediately upon request, usually at uniform cost. In many real-world domains, experiments require varying amounts of time and resources to setup and run. In such domains, failing to plan for such setup activities may make it impossible to run a potentially useful experiment when desired. Thus, it is critical to reason about both: 1) which experiments should be run (the focus of traditional BO), and 2) experiment setup activities that support running useful future experiments. *This motivates the high-level goal of this paper, which is to take a step toward a more significant integration of automated planning and BO.*

As a motivating example, consider the problem of optimizing the power output of microbial fuel cells (MFCs), which use bacteria to generate electricity from various media (e.g. waste water) (Bond and Lovley 2003). The energy production of an MFC can depend on various parameters including the species and mixture of bacteria, the surface properties of the anode, and nutrient level (Park and Zeikus 2003). Running an MFC experiment requires a number of construction steps, including growing a batch of the desired bacteria from frozen stock, noting that a single batch may be able to support multiple experiments. Bacteria growing times vary widely depending on the species, ranging from days to weeks. This necessitates advanced planning to ensure that a particular experiment can be conducted. This planning problem is further complicated by the fact that a laboratory often has multiple facilities for running concurrent experiments and setup activities.

As another example, consider the problem of using biodegradable polymers to fabricate porous scaffolds in bone tissue engineering. Using these polymers allows bone tissue ingrowth without the need for implant removal operation. However, these polymers lack some of the mechanical properties required for human bones and need to be reinforced. A common strategy to improve the mechanical properties of a polymer, e.g. stiffness, is to incorporate filler particles into the organic matrix of the polymer (Balasundaram and Webster 2007). Moreover, the stiffness of a polymer nanocomposite depends on various factors such as the type of reinforcing nanoparticle, the fiber volume fraction, and fiber aspect ratio (Nairn and Mohammadi 2015). The fabrication costs of these nanocomposites can vary extensively depending on the reinforcing agent. Since these experiments are both expensive and time consuming, the main goal is often to maximize the stiffness given a fixed budget. Currently there are no algorithms or tools that can reason about both experiment selection and the scheduling of such setup activities.

A key contribution of this paper is to introduce an extended BO setting, called Bayesian Optimization with Resources (BOR), that explicitly models experimental resources and activities. In particular, our model specifies the following: 1) resource requirements for experiments, which may vary across different experiments, 2) resource-production actions, which produce the various resources and can require varying amounts of time, and 3) a set of "labs" for running concurrent experiments and a set of "production lines" for concurrent resource production. The problem is then to select and schedule the experiments and resource-production actions in order to optimize the unknown objective function within a specified time horizon.

Our second contribution is to propose an online planning algorithm for BOR, which decides at any time point which experiments and resource-production actions to execute. The algorithm is based on finite-horizon tree search, where the goal is to select an action for the current state that maximizes the finite-horizon expected improvement to the unknown function. Unfortunately, the branching factor of raw search trees for BOR problems is too large for vanilla search algorithms. Our approach addresses this issue in two ways. First, we draw on ideas from traditional BO to describe a practical and effective approximation of the raw search tree with a significantly reduced branching factor. Second, and more importantly, we design a novel state evaluation function for assigning long-term values to leaf nodes, which can significantly improve performance.

Our evaluation on benchmark domains shows that our approach, even with only a small amount of search, performs well compared to a variety of baselines.

Related Work. One practically important extension to basic BO is batch (or parallel) BO, which models the availability of multiple experiment facilities and selects a batch of experiments at each step (e.g. (Azami, Fern, and Fern 2010; Desautels, Krause, and Burdick 2012)). Batch BO was later extended (Azami, Fern, and Fern 2011) to model the duration of experiments and explicitly reason about constraints on the number of experiments and time horizon for running experiments. In both cases, simplistic resource models were used and can be viewed as special cases of our framework where there is a single resource and no action for producing resources. Recent work on sequential BO for spatial monitoring (Marchant, Ramos, and Sanner 2014) integrates path planning with BO, by requiring travel to a particular location in order to run each experiment (collect a measurement). Similarly, our work can be viewed as an integration of BO and resource production planning, which is qualitatively different from path planning. While both their solution and ours is based on tree search, the search trees are significantly different and require different approximations to be effective. In summary, while a number of extensions to the basic BO model have been introduced, we are not aware of prior work that directly models and reasons about experimental resources and production actions.

Problem Setup

We consider Bayesian Optimization (BO) problems where each experiment is an element of a d-dimensional space $\mathcal{X}$. Each dimension describes an experiment property and can be either real-valued or discrete over a finite domain. For example, in our motivating fuel cell domain, an experiment $x \in \mathcal{X}$ may have a discrete attribute that specifies the bacteria type to use and real-valued attributes that specify properties such as the composition of the supplied nutrients. An unknown real-valued function $f : \mathcal{X} \rightarrow \Re$ represents the expected value of the dependent variable after running an experiment. For example, $f(x)$ might be the expected power output of a fuel cell experiment described by x. Running an experiment x allows us to observe a possibly noisy outcome $y = f(x) + \epsilon$, where ϵ is a random noise term. The goal of BO is to find an experiment $x \in \mathcal{X}$ that approximately maximizes f by requesting a limited number of experiments and observing their outcomes.

Traditional BO assumes that experiments can be run instantaneously upon request at a uniform cost. In reality, experiments often require resources that can vary across experiments. Because producing those resources requires preparation time, it is critical to reason about resource production in order to support the most useful future experiments. This paper extends traditional BO to account for such resource dependencies by introducing a setting we call Bayesian Optimization with Resources (BOR).

BOR Domains. A BOR domain is a tuple $(\mathcal{X}, \tau, R, C, A, L_e, L_p, H)$, where $\mathcal{X}$ is the space of experiments and τ is a constant specifying the time duration of experiments. The *resource set* $R = \{R_1, \ldots, R_n\}$ is a set of n resource types. For example, R_i may represent a certain bacteria type in our fuel cell example. During the experimental process, the resource vector r^t denotes the available amount of each resource at time t (r^t_i is the amount of R_i). The *cost function* $C(x)$ specifies the resources required for experiment x. Experiment x is *feasible* and can be started at time t only if $C_i(x) \leq r^t_i$ for all i, after which the resource vector r^t is updated by subtracting $C(x)$.

To produce additional resources (e.g. growing a batch of bacteria) the *production action set* $A = \{A_1, \ldots, A_n\}$ specifies a production action A_i for each resource R_i. When an action A_i is executed at time t, it runs for a duration of τ_i and produces an amount a_i of resource R_i that is added to the resource vector at time $t + \tau_i$. L_e (L_p) bounds the maximum number of concurrent experiments (production actions) that can be run. For example, L_e may be bound by the number of sensing apparatus available and L_p may be bound by limited laboratory resources and/or personnel. Finally H is the time horizon within which activities must be completed.

BOR States and Policies. The *state* s^t at time t of a BOR problem captures all information available up to time t. Specifically, a state is a tuple $s = (t, D^t, r^t, E^t, P^t)$, where t is the current time, D^t is a set of experiment-outcome pairs that have completed by time t, r^t is the current resource vector, and E^t (P^t) is a set specifying the currently running experiments (production actions) and the time remaining for each. A *decision state* is a state where there is either a free experimental lab ($|E^t| < L_e$) or an open production line ($|P^t| < L_p$). We say that a decision state s is an e-states when there is an available lab, and otherwise s is an r-state with an available production line.

A *BOR policy* π is a mapping from a decision state to an action, which is either a feasible experiment to run for e-states or a production action for r-states. A policy π is executed by using it to select actions in any encountered decision state until reaching the time horizon. At the horizon the policy must output an experiment x^* that is predicted to have the best value of f. Note that when started in an initial state s_0, the eventual predicted optimum x^* is a random variable. The *expected regret* of π when started in s_0 is then defined as $\max_x f(x) - E[f(x^*) \mid s_0, \pi]$. The goal of BOR planning is to compute policies that achieve small regret.

Additionally, to evaluate a policy π we introduce the following concept. Running π from s until horizon h, each run generates a potentially different set of completed experiments (i.e. experiment-outcome pairs (x, y)), denoted by the random variable $S_\pi^h(s)$. Given a reference value y^* (typically the best output observed before running π), the *expected improvement* of a policy π relative to y^* is defined as $\mathrm{EI}_\pi(s, h, y^*) = E\left[I\left(\max_{(x,y)\in S_\pi^h(s)} y - y^*\right)\right]$, where $I(x) = x$ if $x \geq 0$ and 0 otherwise.

Structured Cost Functions. The cost function $C(x)$ is extremely general and does not capture structure present in real-world domains. While our algorithm developed in Section is applicable to arbitrary cost functions, our analysis will apply to the following special cases, which are inspired by our motivating fuel cell application.

We say that $C(x)$ is *r-uniform* if any experiment that uses a resource uses the same amount of the resource. More specifically, for any x and i, either $C_i(x) = c_i$ or $C_i(x) = 0$ for some constant $c_i > 0$. We say that $C(x)$ is a *partition cost function* if it is r-uniform and there is also a partition $\{X_1, \ldots, X_N\}$ of $\mathcal{X}$ such that for each resource R_i either 1) $C_i(x) > 0$ for all x, or 2) There is an X_j such that for all $x \in X_j, C_i(x) = c_i > 0$ and for all $x \in \mathcal{X} - X_j, C_i(x) = 0$. A partition cost function models cases where there are N experiment types and each resource is either dedicated to a particular type of experiment or is used by all experiments. As an example from our fuel cell domain, different types of fuel cell experiments will use different strains of bacteria resources, while all experiments require basic resources needed for fuel cell operation.

Online Lookahead Search for BOR Planning

Given a BOR domain $(\mathcal{X}, \tau, R, C, A, L_e, L_p, H)$ and a decision state s we now consider how to compute a policy π that will achieve small regret within the horizon H. As is standard in BO, we assume the availability of a posterior $P(f \mid D)$ over the unknown function f given previous experiments D, which implies a posterior $P(y \mid x, D^t)$ over the output y of experiment x. We will use a Gaussian Process for maintaining this density (details in Section).

A BOR domain can be modeled as a Partially Observable Markov Decision Process (POMDP) with hidden state component corresponding to the unknown function f. For all but trivial BOR POMDPs, however, current offline POMDP solvers are not a practical option. Thus, we consider an online planning approach, which constructs a look-ahead search tree for each encountered state in order to estimate action values. The highest-valued action is then executed. This is a popular and successful approach for large-scale planning in both POMDPs (Ross et al. 2008) and MDPs (Kearns, Mansour, and Ng 2002; Kocsis and Szepesvári 2006). To simplify our description of the tree construction, we will view the space of experiments $\mathcal{X}$ and their outcomes $\mathcal{Y}$ as very large finite discrete sets as determined by measurement resolution. As we will see, however, our eventual search algorithm does not depend on this assumption.

BOR Search Trees. Given a current state s^t and a specified *search horizon* H_s, the value for action a defined by our tree is the H_s-horizon expected improvement, with respect to the best experiment outcome y^* observed up to time t. In particular, for action a, this is equal to the EI obtained by executing action a and then following an optimal policy until time H_s. The action that maximizes this notation of value is intuitively the one that we can expect to lead to the most potential for improvement.

Figure 1 illustrates a fragment of an example BOR tree. Paths in a tree alternate between state nodes (circles) and action nodes (squares). We will let $n_a(s)$ denote the action node corresponding to action a with parent state $s = (t, D^t, r^t, E^t, P^t)$. The root node of a BOR tree is the current state s^t, which in our example is an r-state. The children of each state node are action nodes corresponding to the possible actions at the state. As illustrated in the example, for r-states the possible actions are the n resource production actions and for e-states the actions correspond to the possible feasible experiments in $\mathcal{X}$.

The children of an action node $n_a(s)$ are the possible next decision states that can be reached after taking a in s, which is the state occurring when either the next experiment finishes (an e-state) or lab becomes available (an r-state). The times of these events are easily inferable from the action durations in our action models. Further the edges to those children are weighted by the probability of their occurrence. The child generation process depends on whether the next decision state is an r-state or e-state and whether an experiment or resource action ended. Here, we illustrate two cases in detail. The remaining cases are similar but are omitted due to space constraints.

The first case, illustrated by the leftmost action child of the root in Figure 1, is for a resource actions that lead to e-states at time t_1, meaning that an experiment x ended at t_1. There will be one child e-state for each possible outcome y of x, with edge weight given by the posterior $P(y \mid x, D^t)$. In particular, for each y we will create a new child state having tuple $(t_1, D^t \cup \{(x, y)\}, r^t, E^t - \{x\}, P^{t_1})$, where the time is updated to t_1, the outcome (x, y) is added to the set of completed experiments, and x is removed from the set of running experiments, the resource vector is unchanged from r_t since no new resources were consumed or produced at t_1. Finally, P^{t_1} is an update of P^t reflecting that resource action A_1 just started running. Each of these resulting e-states has children (action nodes) corresponding to the possible experiments that could be selected in them.

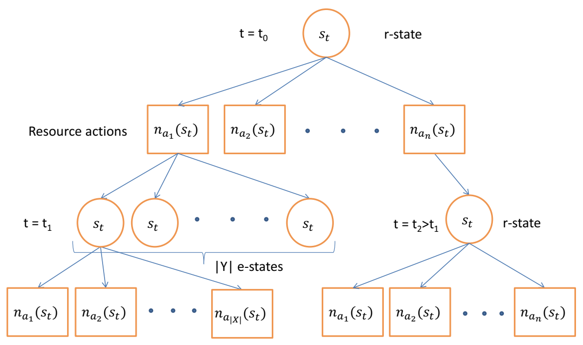


Figure 1: Partial illustration of a search tree

The second case, illustrated by the rightmost action node in Figure 1, is when the next decision state is an r-state occurring at time t_2, meaning that a production line freed up at t_2 due to resource production action A_i finishing. In this case, there is a single child r-state of the action, which has children corresponding to each resource action available at that state. The state description for this new r-state is the tuple $(t_2, D^t, r^{t_2}, E^t, P^t - \{A_i\})$, where the time is updated to t_2, action A_i is removed from the set of running resource actions, and r^{t_2} is updated so that $r_i^{t_2} = r_i^t + a_i$ indicating that an increment to resource R_i occurred. The set of running experiments and completed experiments remain unchanged.

The full BOR tree is defined as above with leaves corresponding to the fringe of state nodes whose time stamps are less than the search horizon H_s and whose children would have time stamps greater than H_s. The EI values assigned to state and action nodes can be defined bottom up starting with the leaves. The value of a leaf node with finished experiment set D is simply $I\left(\max_{(x,y)\in D} y - y^*\right)$, recalling that y^* is the best output observed in the real world so far (i.e. before the search began). The value of an action node is equal to the expectation of values assigned to its child state nodes. The value of a state node is defined as the maximum value of its child action nodes, reflecting that state nodes have a choice over actions. In this way the action values in the tree for root s^t will correspond to the H_s-horizon EI as desired.

Unfortunately, it is impractical to construct the full tree. First, the action branching is enormous, especially at e-states. Second, the stochastic branching of action nodes where experiments complete is very large (i.e. $|\mathcal{Y}|$). This seriously limits the search horizons that can be used. Below we first describe how we address the large branching factors by approximating the tree. Even then the practically attainable depths are quite limited. Thus, next we describe how to compute informative leaf evaluation functions that attempt to provide longer term estimates of leaf node values, which translates into improved action values at the root.

Search Tree Approximation. One approach for dealing with stochastic branching due to completed experiments is to follow the Sparse Sampling approach (Kearns, Mansour, and Ng 2002) and sparsely sample a smaller number of k outcomes for the completed experiment, which results in just k new e-states compared to $|\mathcal{Y}|$. Unfortunately, even for relatively small values of k the feasible search depths are very small since the tree size is exponential in k. Further, small values of k introduce large variance into the search results. Thus, we follow a relatively common alternative approach, which is to use a deterministic transition model that assigns all experiment outcomes to its MAP estimate. That is, the only e-state included under an action node $n_a(s)$ corresponds to the outcome $\arg\max_y P(y \mid x, D)$, where x is the newly completed experiment and D is the set of complete experiments for state s. This MAP approximation is a commonly employed and often successful technique in control and AI planning (e.g. (Yoon, Fern, and Givan 2007; Platt et al. 2010; Marchant, Ramos, and Sanner 2014)). Our initial investigation showed that it is a more effective approximation for BOR problems than Sparse Sampling.

In order to deal with the large action branching at e-nodes (i.e. $|\mathcal{X}|$), we draw on ideas from BO. BO has developed a number of policies for selecting experiments given prior experiment outcomes. Among the most effective is the *maximum expected improvement (MEI) policy* π_{ei}, which selects the feasible experiment x that maximizes the EI function, i.e. $\pi_{ei}(s) = \arg\max_{x\in\mathcal{X}_s} E\left[I(y - y^*) \mid x, D\right]$, where $y \sim P(y \mid x, D)$, D is the set of finished experiments at s, and $\mathcal{X}_s$ is the set of feasible experiments at s. EI can be computed efficiently for the GP models used in our work. We leverage this idea by only considering the action selected by π_{ei}, which eliminates action branching due to experiment selection. While this appears to be an extreme approximation, π_{ei} has consistently shown good performance in BO and is quite effective in our experiments.

After the above simplifications, the remaining branching in the tree corresponds to action branching at r-states, which is equal to the number of resource actions $|R|$. Thus, the computation time of constructing our approximate search tree will be $O(d^{|R|})$, where d is the maximum number of r-states encountered on a path from the root to a leaf. For this reason we will parameterize our search trees by not only the search horizon H_s but also by the *r-depth* d, which specifies the maximum number of r-states allowed on any tree path. During tree construction, whenever an r-state is encountered that goes beyond the r-depth, it is converted into a leaf node of the tree, which for small values of d will typically occur before reaching the search horizon. Thus, while using small values of d allow for efficiency, the price is that the value computed for such early terminating leaf nodes can be quite short sighted compared to the search horizon, which can hurt overall performance. This is accounted for by our leaf evaluation function described next.

State Evaluation Function. Given a leaf node for state $s = (t', D, r, E, P)$ with $t' < H_s$ the default leaf evaluation from above is based only on the experiment outputs in D. This ignores the potential experiments that could be run and completed between time t' and H_s using resources that were produced in the tree leading to s. Thus, the leaf and resource production decision above it can be severely undervalued.

The purpose of the state evaluation function is to estimate the potential long term value that could be obtained from a leaf. Long term value corresponds to the potential improvement of continuing to select and run experiments until the horizon. One measure of this potential would be to compute a set of experiments X^* that achieves the maximum EI under the constraint that X^* consumes resources that are either available at the leaf or that will become available in the future. In other words, X^* maximizes $G(X, y^*) = E\left[\max_{e \in X \cup X'} I(y_e - y^*)\right]$, where the expectation is with respect to $P(y \mid D)$, D contains the observed data, and X' is the set of ongoing experiments at the leaf.

Unfortunately, even computing X^* is a computationally hard problem, which follows from the computational hardness of BO. Fortunately, however, we are able to compute a set of experiments $\hat{X}$ that achieves an EI within a constant factor of $G(X^*, y^*)$ for a wide class of BOR problems. We compute $\hat{X}$ using the following simple greedy algorithm. We initially set $\hat{X} = \emptyset$ and at each iteration add an experiment to $\hat{X}$ that produces the maximum increase in EI of the resulting set while ensuring that the resulting set satisfies the resource constraints. More formally, at each iteration given the current $\hat{X}$ we add an experiment x that maximizes $G(\hat{X} \cup \{x\}, y^*)$, subject to the constraint that $\hat{X} \cup \{x\}$ can be produced using resources available at the leaf or that are currently being produced at the leaf. Unfortunately there is no closed form expression for $G(X, y^*)$, however, it is straightforward to estimate the expectation G to an arbitrary accuracy via Monte-Carlo sampling.

Approximation Bound. We now draw on the theory of greedy submodular optimization (Nemhauser, Wolsey, and Fisher 1978), which provides approximation bounds for greedy optimization of sets as is done for our heuristic leaf evaluation function. We only outline the main concepts due to space limits (see (Nemhauser, Wolsey, and Fisher 1978)). Our results partly rely on viewing $G(X, y^*)$ as a set function of X and showing that it is a monotone increasing, submodular set function. A set function is *monotone increasing* if adding elements to X never decrease the value. Submodularity is a type of diminishing returns property. A set function G is *submodular* if for any two sets $X_1 \subseteq X_2$, adding an element x to X_1 will improve G by at least as much as adding x to the superset X_2.

Proposition 1 *For any value of y^*, $G(\cdot, y^*)$ is a monotone increasing and submodular set function for a Gaussian Process prior over the unknown function f.*

Proof. We first introduce some notations and definitions. Any mapping from all experiments to their possible outcomes is called a realization ϕ. We say that outcomes of a set of experiments A (i.e. $Y_A = \{y_{e_i}\}$ where $e_i \in A$) are a partial realization consistent with a ϕ if they are equal over the set A and we write $Y_A \sim \phi$. And lastly, Y_B is a subrealization of Y_A if and only if $B \subseteq A$ and Y_A and Y_B are both consistent with some realization ϕ and we write $Y_B \subseteq Y_A$.

Monotonicity: Given a partial realization Y_X and any experiment e', we have

$$max\{I_{e \in X}\left(y_e - y^*\right)\} \leq max\{I_{e \in X}\left(y_e - y^*\right), I\left(y_{e'} - y^*\right)\}$$
$$\leq max\{I_{e \in X \cup \{e'\}}\left(y_e - y^*\right)\}$$

where y^* is any arbitrary reference point and $y_{e'}$ is any realization for e'. Since this inequality holds for any realizations we get,

$$E\left[max\{I_{e \in X}\left(y_e - y^*\right)\}\right] \leq E\left[max\{I_{e \in X \cup \{e'\}}\left(y_e - y^*\right)\}\right]$$

which implies that $G(X, y^*) \leq G(X \cup \{e'\}, y^*)$, for all e'. Therefore, G is monotone increasing. □

Submodularity: We need to show that

$$G\left(X_2 \cup \{e'\}, y^*\right) - G\left(X_2, y^*\right)$$
$$\leq G\left(X_1 \cup \{e'\}, y^*\right) - G\left(X_1, y^*\right)$$

for all $X_1 \subseteq X_2 \subseteq E$ and $e' \in E \setminus X_2$ where E is the set of all experiments. For the case that $X_1 = X_2$ the marginal benefits of adding an experiment to each set is clearly the same. Now consider the more interesting case where $|X_1| = m < |X_2| = n$. Since G is the expected improvement of a set, the order of the elements of the set does not affect the objective. Therefore, for any experiment e' without loss of generality we can write $X_2 \cup \{e'\} = \{e_1, e_2, \ldots, e_m, e_{m+1}, \ldots, e_n, e'\}$ where $X_1 = \{e_1, e_2, \ldots, e_m\}$. Now for any partial realization $Y_{X_2 \cup \{e'\}}$, $Y_{X_1} \subseteq Y_{X_2}$ and we have:

$$max\left(I_{e \in X_2}(y_e - y^*), I(y_{e'} - y^*)\right) =$$
$$max\left(I_{e \in X_1}(y_e - y^*), I_{e \in X_2 \setminus X_1}(y_e - y^*), I(y_{e'} - y^*)\right)$$

To calculate the marginal benefit of adding e' to X_2, only the cases count where $I(y_{e'} - y^*)$ is greater than both $I_{e \in X_1}(y_e - y^*)$ and $I_{e \in X_2 \setminus X_1}(y_e - y^*)$. Whereas to calculate the marginal benefit of adding e' to X_1, $I(y_{e'} - y^*)$ only needs to be greater than $I_{e \in X_1}(y_e - y^*)$. Note that a Gaussian prior is assumed over the underlying function and that adding new elements to a set can only decrease the variance on the other elements. Therefore, the existence of $\{e_{m+1}, \ldots, e_n\}$ in X_2 not only might result in $I(y_{e'} - y^*)$ getting dominated by $I_{e \in X_2 \setminus X_1}(y_e - y^*)$ but it may also decrease the variance at the point e' which means lowering the probability of observing larger outcomes for $y_{e'}$. Also note that $p(\{y_{e_1}, y_{e_2}, \ldots, y_{e_m}\} | Y_{X_1} \sim \phi) = p(\{y_{e_1}, y_{e_2}, \ldots, y_{e_m}\} | Y_{X_2} \sim \phi)$. Combining these observations implies that:

$$E\left[I_{e \in X_2 \cup \{e'\}}(y_e - y^*) - I_{e \in X_2}(y_e - y^*)\right] \leq$$
$$E\left[I_{e \in X_1 \cup \{e'\}}(y_e - y^*) - I_{e \in X_1}(y_e - y^*)\right]$$

This implies that $G(X_2 \cup \{e'\}, y^*) - G(X_2, y^*) \leq G(X_1 \cup \{e'\}, y^*) - G(X_1, y^*)$, which completes the proof. □

If there were no resource constraints then the standard $(1 - e^{-1})$ approximation result for greedy submodular optimization would hold. That is, our greedy approach for optimizing the set would be within a factor of $1 - e^{-1})$ of optimal. In our case, however, the sets being optimized over must satisfy the resource constraints, and the greedy algorithm can only add experiments subject to those constraints. Thus, this standard result does not apply. Instead we draw on work that considers submodular optimization under matroid constraints (Calinescu et al. 2007).

Consider a leaf state s with time stamp t' and available resource vector r. We aim to characterize the constraints on the possible feasible sets X of experiments that could be produced. The simplest constraint is on the total number of experiments. Given the currently running experiments in s, the number of labs, and the experiment duration, we can calculate the number k of additional experiments could be run between time t' and H_s. Thus, the first constraint on X is the cardinality constraint $|X| \leq k$. Next, the resources specified by r must be sufficient to support all experiments in X. Thus, the second constraint is $\sum_{x \in X} C(x) \leq r^t$. An approximation result in (Calinescu et al. 2007) states that if constraints on sets X can be defined as an intersection of p matroids (more generally p-independence system), then the greedy algorithm, in our case selecting $\hat{X}$, achieves a $1/(p+1)$ factor approximation guarantee compared to the best possible set of experiments X^*. It remains to characterize p for classes of BOR problems.

Definition 1 (Matroid) *A matroid M is an ordered pair $(\mathcal{X}, \mathcal{I})$ consisting of a finite set $\mathcal{X}$ and a collection $\mathcal{I} \subseteq 2^{\mathcal{X}}$ such that: 1) $\emptyset \in \mathcal{I}$ 2) For $A \subset B \subseteq \mathcal{X}$ and $B \in \mathcal{I}$, then $A \in \mathcal{I}$, and 3) If A and B are in $\mathcal{I}$ and $|B| > |A|$, then there exists $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$.*

The cardinality constraint on $\mathcal{X}$ defines the well-known uniform matroid. In general, however, the resource constraint will not be a single matroid. For r-uniform and partition cost functions we can show the following.

Proposition 2 *For any resource vector r, let $\mathcal{I} \subseteq 2^{\mathcal{X}}$, contain all experiment subsets X such that $\sum_{x \in X} C(x) \leq r$. If C is r-uniform, then $\mathcal{I}$ can be represented as the intersection of $|R|$ matroids. If, in addition, C is a partition cost function then $(\mathcal{X}, \mathcal{I})$ is a matroid.*

Proof. For any $i = \{1, \ldots, |R|\}$, let $\mathcal{I}_i$ be all the experiments that have the required amount of resource i i.e. $\{x \mid 0 < C_i(x) \leq r_i\}$. We show that $(\mathcal{X}, \mathcal{I}_i)$ forms a matroid. Given a vector r, if there is not enough of resource i available, then no corresponding experiment can be run. Therefore, $\emptyset \in \mathcal{I}_i$. Consider any two arbitrary sets A and B where $A \subset B \subseteq \mathcal{X}$ and $B \in \mathcal{I}_i$. We need to show that $A \in \mathcal{I}_i$. Assume for contradiction that $A \notin \mathcal{I}_i$. This means that there is an experiment in A that does not have the required amount of resource i. But this is not possible since all the experiments in A are also elements of B and $B \in \mathcal{I}_i$. So, A must be in $\mathcal{I}_i$ as well.

Now consider any two arbitrary sets A and B in $\mathcal{I}_i$ for which $|B| > |A|$. If the cost function is r-uniform, any experiment that uses resource i uses the same amount. This means that there exists enough of resource i for at least $|B| - |A|$ more experiments after A. Since $|B| > |A|$, we know that there exist an experiment $x \in \{B \setminus A\}$ such that $A \cup \{x\} \in \mathcal{I}_i$. Therefore, $(\mathcal{X}, \mathcal{I}_i)$ is a matroid for any i. Note that $\bigcap_{i=1}^{|R|} \mathcal{I}_i$ gives us $\{x \mid 0 < C_i(x) \leq r_i, \forall i\}$ i.e. all feasible experiments. This means $(\mathcal{X}, \bigcap_{i=1}^{|R|} \mathcal{I}_i)$ is indeed the intersection of $|R|$ matroids providing us with the feasible experiment space.

Now we show that partition cost would result in a single matroid. To do so, first let $\mathcal{I}$ be all feasible sets of experiments. Clearly, as above, $\emptyset \in \mathcal{I}$ and for any two sets A and B, such that $A \subset B \subseteq \mathcal{X}$ and $B \in \mathcal{I}$, A must be in $\mathcal{I}$.

Now consider arbitrary sets A and B in $\mathcal{I}$ where $|B| > |A|$. Recall that for a partition cost function, we have a partition $\{X_1, \ldots, X_N\}$ of $\mathcal{X}$ and each resource is used either by one particular partition or by all the experiments. Therefore, if both A and B are feasible and $|B| > |A|$, there must exist a partition j in which the number of elements in B is greater than the number of elements in A (i.e. $|B_j| > |A_j|$). In other words, there exist an experiment $e_{B_j} \notin A_j$. Since all the experiments in $|B_j|$ are feasible and we know that the required resources for $|B_j|$ cannot be consumed by any other partition, e_{B_j} can be added to A and $A \cup \{e_{B_j}\}$ would still remain feasible. This completes the proof. □

Thus, we see that for r-uniform and partition cost functions the feasible experiment sets can be represented by $|R| + 1$ and 2 matroids respectively, which combined with Proposition 1 gives the following main result.

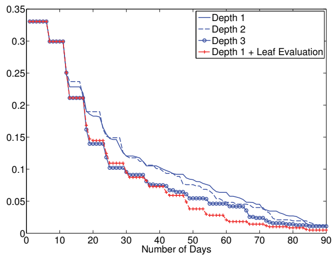
Theorem 1 *For any leaf state s and y^*, it holds that $G(\hat{X}, y^*) \geq \alpha \cdot G(X^*, y^*)$, where $\alpha = \frac{1}{|R|+2}$ for r-uniform cost functions and $\alpha = \frac{1}{3}$ for partition cost functions.*

Empirical Results

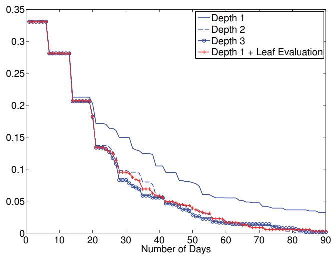
GP Model. Our approach and baselines require a posterior over f, for which we use zero-mean Gaussian Process (GP) priors. The space of experiments $\mathcal{X}$ in our benchmarks include both real-valued and discrete attributes (indicating the type of an experiment). We handle discrete attributes by maintaining one GP over real-valued attributes for each combination of discrete attribute values. More sophisticated GP models could be used for larger numbers of discrete attributes. The co-variance function of each GP is given by a Gaussian kernel $K(x, x') = \sigma \exp\left(-\frac{1}{2w} \parallel x - x' \parallel^2\right)$, with signal variance $\sigma = 1$ and kernel width $w = 0.05$. Depending on the application in hand, one can use any parameter selection approach desired. We found these fixed parameter values work just fine for the purpose of this work.

Unless otherwise specified, the time horizon is $H = 90$ days in all experiments and there are 5 experimental labs ($L_e = 5$) and 2 resource production lines ($L_p = 2$). Four different production time settings [5, 7, 11], [11, 7, 5], [8, 8, 8] and [7, 11, 7] are used to evaluate the performance of the introduced algorithms. Production time [5, 7, 11] means that producing resources 1 through 3 takes 5, 7 and 11 days respectively. These settings roughly have the average time of 8 days but are distributed in different patterns to cover a vast range of possible scenarios. Our results report the average regret (over 50 runs) achieved by each method throughout the 90 day duration.

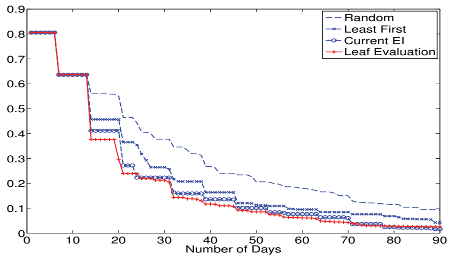
Benchmarks. Quantitatively comparing BO algorithms in real-world laboratories is not practical. In particular, the real laboratory experimentation process can only be done once due to time and resource constraints, which does not allow for repeated BO algorithm runs that would be required for comparison. We address this problem in two ways.



(a) [11, 7, 5]



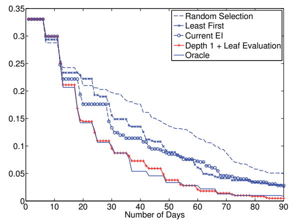
(b) [7, 11, 7]



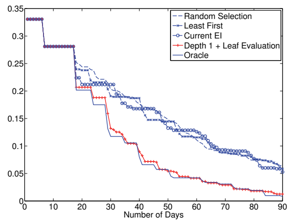
(c) Nitrogen Production

Figure 2: (a and b) Results varying resource depth and leaf evaluation. (c) Nitrogen production experiments.

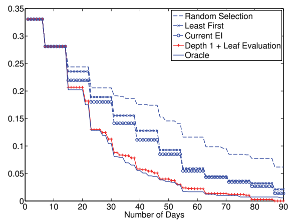
Synthetic Functions (Independent Resource Structure)



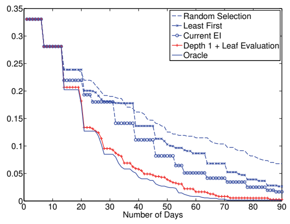
(a) [11, 7, 5]



(b) [5, 7, 11]

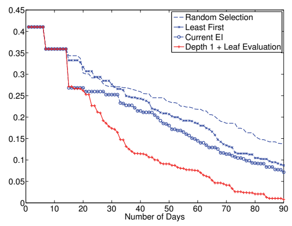


(c) [8, 8, 8]

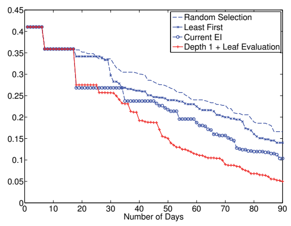


(d) [7, 11, 7]

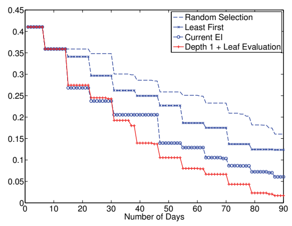
Synthetic Functions (Shared Resource Structure)



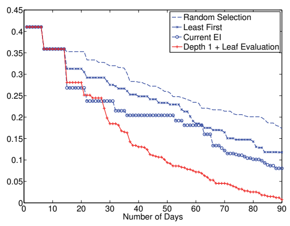
(e) [11, 7, 5, 8]



(f) [5, 7, 11, 8]

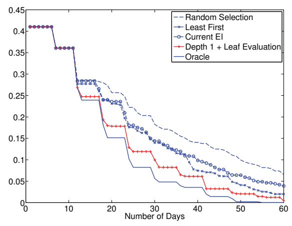


(g) [8, 8, 8, 8]

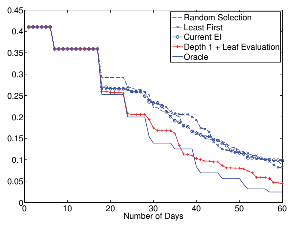


(h) [7, 11, 7, 8]

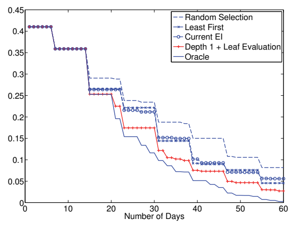
Synthetic Functions (Double Capacity)



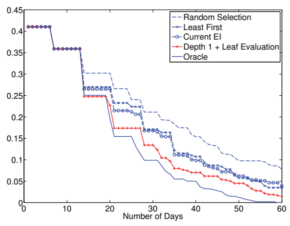
(i) [11, 7, 5]



(j) [5, 7, 11]

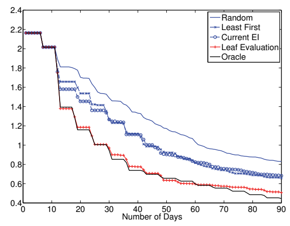


(k) [8, 8, 8]

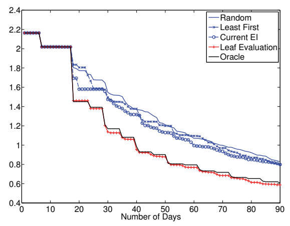


(l) [7, 11, 7]

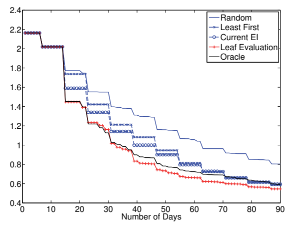
Synthetic Functions (6 Dimensional)



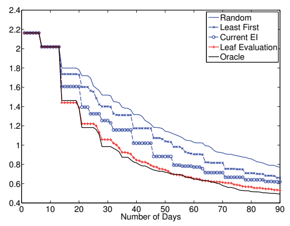
(m) [11, 7, 5]



(n) [5, 7, 11]



(o) [8, 8, 8]



(p) [7, 11, 7]

Figure 3: Synthetic function experiments using different resource models. (First Row) Independent resource structure. (Second Row) Shared resource structure. (Third Row) Double capacity structure. (Forth Row) 6 dimensional structure

Synthetic Benchmarks: We designed benchmarks that emulate the resource structure of our motivating fuel cell problem and other domains, but use synthetic functions to simulate experimental outcomes. Each benchmark is modeled as having three types of experiments, e.g. each using a different strain of bacteria. Each experiment type corresponds to one of three functions. In particular, we consider both 2-d and 6-d function sets. For the 2-d functions we use the well-known benchmarks: Rastriging, Rosenbrock, and Cosines defined over $[-1, 1]^2$. For the 6-d case we use: Rosenbrock, Hartman6 and Styblinski defined over $[0, 1]^6$. Thus, each experiment x has one discrete dimension, indicating the type of experiment and either an additional 2 or 6 real-valued dimensions. The experiment duration is uniformly $\tau = 6$ days.

We consider two types of resource structure. The first, *Independent*, which includes one resource for each of the three experiments types. Each experiment consumes a unit amount of its corresponding resource. The production action for each resource produces three units each time it is run and the durations of the actions are specified by a vector $[\tau_1, \tau_2, \tau_3]$, where τ_i gives the number of days required to produce resource R_i. The second resource structure, *Shared*, is similar to Independent, but includes a fourth resource, which is shared by all experiments. Each experiment consumes 0.5 units of the fourth resource in addition to the single unit of its type specific resource. Again, each resource production action produces three units of a resource and the durations are parameterized by a vector $[\tau_1, \tau_2, \tau_3, \tau_4]$.

Nitrogen Production Benchmark: We developed a benchmark that simulates the maximization of nitrogen production of the bacterium, nitrosmonas europaea, by varying the amount of CO and HCO_3 intake. We used a genome-scale mathematical model to simulate the metabolic network and biochemical activity in order to generate simulated experimental outcomes. The discrete experiment variable indicates which form of nitrogen is being considered for optimization, either NO, N_2O, or N_2. The resource structure is based on the preparation of different measuring devices for each nitrogen form. We model this by having one resource per form of nitrogen. The setup for N_2 experiments is estimated to be approximately 1/3 more costly than for NO and N_2O which are roughly equal. Our evaluation reflects this structure.

Baseline Policies. Our baselines use the MEI policy for selecting experiments for free labs. The baselines differ in how resource actions are selected for free production lines. We evaluate the following baselines: *Random* selects random resource actions. *Least First (LF)* selects the resource with the least amount. *Current EI* identifies the experiment with the current highest EI and selects among its required resources the one that currently has the least amount. *Oracle* knows which of the three functions achieves the global optimum and always produces that resource.

Impact of Depth and Leaf Evaluation. We consider the impact of the resource depth d of the search tree and the leaf evaluation function on our method. Figs. 2a and b shows results of tree search for two production time settings for $d = 1, 2, 3$ without the leaf evaluation function, and for $d = 1$ with the leaf evaluator ($d = 2, 3$ with leaf evaluation are almost identical to $d = 1$). When leaf evaluation is not used, increasing the depth d improves performance, as expected. Interestingly, $d = 1$ with leaf evaluation is never worse and sometimes better than even $d = 3$ without leaf evaluation. Further, $d = 1$ achieves this performance at a much smaller computational cost. These results suggest that the dominating performance factor is the leaf evaluation function.

Comparing to Baselines. Figure 3 (first row) compares our method for $d = 1$ and leaf evaluation to the baselines for the Independent resource scenario. While not shown, the results for $d = 2, 3$ with leaf evaluation are nearly the same as for $d = 1$, but much more computationally expensive. We first observe that our approach outperforms all non-oracle baselines, noting that the relative ordering of baselines varies with scenario. We also see that our approach nearly equals Oracle.

Figure 3 (second row) gives results for the Shared resource setting where the additional shared resource always requires 8 days to produce. For this scenario we considered $d = 2$ since $d = 1$ would not allow both resources necessary for each experiment to be produced. For the Shared scenario there is no obvious Oracle policy, since experiments depend on multiple resources and thus Oracle is not shown in these results. Again, we see that our approach significantly outperforms all baselines in this more complex setting.

In Figure 3 (third row), we double the number of labs and production lines in the independent scenario in order to observe the impact of increased experimental capacity. In this scenario doubling the capacity would allow us to run more experiments and as a result decrease the regret in a shorter period of time. For this reason, we only show the result for the time horizon of $H = 60$ days. Our method outperforms non-oracle baselines, though with less margin than with smaller capacities. We believe that this is due to the availability of more concurrency. Here, however, we see that Oracle outperforms our approach by a small margin. It is unclear why our relative performance to Oracle varies with the amount of concurrency.

All the problems that we have considered so far had two real-valued attributes. Note that increasing the dimensionality would increase the experiment space exponentially and the problem soon gets intractable. For this reason, we use the DIRECT optimization package (Finkel 2003) to maximize our functions of interest such as the expected improvement. This allows us to run our algorithm on higher dimensional problems in a reasonable amount of time without losing much accuracy. Figure 3 (fourth row) shows the comparison between our method and other baselines. As we can see our method outperforms the non-oracle baselines consistently. Comparing between the baselines, we observe that the *Current EI* dominates the *Least First* which in turn outperforms *Random*, except for the time setting [11, 7, 5] where *Least First* and *Current EI* are intertwined,

Figure 2c shows the results for the nitrogen domain. We see that our approach outperforms all baselines, except for

Current EI, which achieve similar performance. We note that the margin of improvement compared to the non-random baselines is smaller than for some of the synthetic experiments. We hypothesize that this is due to the fact that some of the synthetic functions have significantly more complex response surfaces with more local optima. Overall the results are encouraging and suggest that there may be benefits to using our method in real laboratory settings.

Summary

We introduced the new problem of Bayesian Optimization with Resources (BOR), which extends BO to account for experiment resource requirements and productions. An online planning approach based on depth-limited tree search was introduced for selecting experiments and resource production actions. Our empirical results on a set of benchmarks with diverse resource requirements show that the proposed approach significantly outperforms a variety of natural baselines and is even competitive with a policy that uses an optimal resource production oracle.

Acknowledgments

This research was supported by NSF grant IIS 1320943.

References

Azami, J.; Fern, A.; and Fern, X. 2010. Batch bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems*.

Azami, J.; Fern, A.; and Fern, X. 2011. Budgeted optimization with concurrent stochastic-duration experiments. In *Advances in Neural Information Processing Systems*.

Balasundaram, G., and Webster, T. J. 2007. An overview of nano-polymers for orthopedic applications. *Macromolecular bioscience* 7(5):635–642.

Bond, D., and Lovley, D. 2003. Electricity production by geobacter sulfurreducens attached to electrodes. *Applications of Environmental Microbiology* 69:1548–1555.

Brochu, E.; Cora, V. M.; and de Freitas, N. 2010. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. eprint arXiv:1012.2599, arXiv.org.

Calinescu, G.; Chekuri, C.; Pál, M.; and Vondrák, J. 2007. Maximizing a submodular set function subject to a matroid constraint. In *Integer programming and combinatorial optimization*. Springer. 182–196.

Desautels, T.; Krause, A.; and Burdick, J. 2012. Parallelizing exploration-exploitation tradeoffs with gaussian process bandit optimization. In *Proceedings of the 29th International Conference on Machine Learning*.

Finkel, D. E. 2003. Direct optimization algorithm user guide. *Center for Research in Scientific Computation, North Carolina State University* 2.

Jones, D. 2001. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization* 21:345–383.

Kearns, M.; Mansour, Y.; and Ng, A. Y. 2002. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning* 49(2-3):193–208.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European Conference on Machine Learning*.

Marchant, R.; Ramos, F.; and Sanner, S. 2014. Sequential bayesian optimisation for spatial-temporal monitoring. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*.

Nairn, J. A., and Mohammadi, M. S. 2015. Numerical and analytical modeling of aligned short fiber composites including imperfect interfaces. *Composites Part A: Applied Science and Manufacturing* 77:26–36.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming* 14(1):265–294.

Park, D., and Zeikus, J. 2003. Improved fuel cell and electrode designs for producing electricity from microbial degradation. *Biotechnol.Bioeng.* 81(3):348–355.

Platt, R.; Tedrake, R.; Kaelbling, L.; and Lozano-Perez, T. 2010. Belief space planning assuming maximum likelihood observations. In *Proceedings of Robotics: Science and Systems*.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-Draa, B. 2008. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research* 663–704.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. Ff-replan: A baseline for probabilistic planning. In *International Conference on Automated Planning and Scheduling*, volume 7, 352–359.