Toward Embedding Bayesian Optimization in the Lab: Reasoning about Resource and Actions

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
Bayesian optimization (BO) aims to optimize costly-to-evaluate functions by running a limited number of experiments that each evaluate the function at a selected 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.

1 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 $f$ given prior experiments (Jones 2001; Brochu, Cora, and de Freitas 2010). The posterior is then used to select new experiments that trades-off exploring uncertain areas of the experimental space 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 due to lengthy setups. 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.

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 time 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. 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 derived from BOR problems is too large for vanilla search algorithms to be effective. 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 leaves of the search tree, which can significantly improve performance.

Our evaluation on a number of benchmark domains with qualitatively different resource structures show that our approach, even with only a small amount of search, performs well compared to a variety of baselines and an oracle method.

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.

2 Problem Setup

We consider Bayesian Optimization (BO) problems where each experiment is an element of a d-dimensional space \( X \). 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 \( x \in X \) 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 \( f : X \rightarrow \mathbb{R} \) 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 \( \epsilon \) is a random noise term. The goal of BO is to find an experiment \( x \in X \) 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 \((X, \tau, R, C, A, L_0, L_p, H)\), where \( X \) is the space of experiments and \( \tau \) 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 \( \tau_i \) and produces an amount \( a_i \) of resource \( R_i \) that is added to the resource vector at time \( t + \tau_i \). \( L_0 \) (\( L_p \)) bounds the maximum number of concurrent experiments (production actions) that can be run. For example, \( L_0 \) 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 (\( |E^t| < L_0 \)) or an open production line (\( |P^t| < L_p \)). We say that a decision state \( s \) is an \( e \)-state when there is an available lab, and otherwise \( s \) is an \( r \)-state with an available production line.

A BOR policy \( \pi \) 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 \( \pi \) 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 \( \pi \) when started in \( s_0 \) is then defined as \( \max_{x^*} E[f(x^*)] - E[f(x^* \mid s_0, \pi)] \). The goal of BOR planning is to compute policies that achieve small regret.

It is useful to define the concept of expected improvement of a policy \( \pi \). Running \( \pi \) 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 \( S^h_{x,y}(s) \). Given a reference value \( y^* \) (typically the best output observed before running \( \pi \)), the expected improvement of \( \pi \) relative to \( y^* \) is defined as \( EI_x(s, h, y^*) = E \left[ I \left( \max_{(x,y) \in S^h_{x,y}(s)} y - y^* \right) \right] \), where
\[ I(x) = x \text{ if } x \geq 0 \text{ and 0 otherwise.} \]

**Structured Cost Functions.** The above cost function \( C(x) \) is extremely general and does not explicitly capture structure present in real-world domains. While our algorithm developed in Section 3 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 for any \( x \) and \( i \), either \( C_i(x) = c_i \) or \( C_i(x) = 0 \) for some constant \( c_i > 0 \). This models situations where any experiment that uses a resource uses the same amount of the resource. 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 \( X \) such that for each resource \( R_i \) either 1) \( C(x) > 0 \) 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 \( x \in X - X_j \), \( C(x) = 0 \). 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.

3 Online Tree Search for BOR Planning

Given a BOR domain \((X, \tau, R, C, A, L_o, L_p, H)\) and a decision state \( s \) we now consider how to compute a policy \( \pi \) that will achieve some regret with in the horizon \( H \). As is standard in BO, we assume the availability of a posterior \( P(f | D) \) over the unknown function \( f \) given previous experiments \( D \), which implies a posterior \( P(y | x, D^t) \) over the output \( y \) of experiment \( x \). We will use a Gaussian Process for maintaining this density (details in Section 4).

A BOR domain can be modeled as a Partially Observable Markov Decision Process (POMDP) with hidden state component corresponding to \( f \). Most BOR POMDPs, however, are beyond the scalability of current offline POMDP solvers. Thus, we consider an online planning approach, which constructs a finite-horizon 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). In order to limit attention to discrete trees, throughout we assume that the spaces of experiments \( X \) and their outcomes \( Y \) have been discretized.

**BOR Search Trees.** Given current state \( s' \) and a search horizon \( H_s \), the tree is used to estimate the \( H_s \)-horizon expected improvement (EI) of each action 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 with the best EI is intuitively the one that will lead to the best long-term improvement over \( y^* \).

Tree nodes correspond to either decision states or actions with root \( s' \) and paths alternating between state and action nodes. The children of \( r \)-state and \( e \)-state nodes are resource production actions or feasible experiments respectively. Let \( n_a(s) \) denote the action node corresponding to action \( a \) with parent state \( s = (t_s, D, r, E, P) \). The children of \( n_a(s) \) are the possible next decision states that can be reached after taking \( a \) in \( s \). 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.

Space precludes details and below we consider one case as an example.

We consider the case when \( a \) is a resource production action \( A_i \) and the next decision state corresponds to an experiment \( x \) completing at time \( t' \geq t_s \). The children of \( n_a(s) \) will be \( e \)-states, since a lab will be available for running a new experiment. There will be one child \( e \)-state for each possible outcome \( y \) of \( x \). In particular, for each outcome \( y \in Y \) we will create a new state \( s' = (t', D \cup \{(x,y')\}, r, E - \{x\}, P \cup \{A_i\}) \), where the time is updated to \( t' \), the outcome \( (x, y) \) is added to the set of completed experiments, \( x \) is removed from the set of running experiments, and \( A_i \) is added to the running resource actions. The resource vector \( r \) remains unchanged since no new resource actions ended and no new experiments were started. The probability associated with \( s' \) is given by the posterior \( P(y' | x, D) \) at state node \( s \).

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 \( s' \) will correspond to the \( H_s \)-horizon EI as desired.

Unfortunately, it is impractical to construct the full search tree. First, the action branching factor is enormous, especially at \( e \)-states. Second, the stochastic branching factor of action nodes where experiments complete is very large (i.e. \(|Y|\)) These factors seriously limit 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 \(|Y|\). Unfortunately, even using small values of \( k \) increases the overall tree branching factor by many fold and using small values of \( k \) can introduce large variance into the search results. Thus, we follow an alternative approach, which is to use a deterministic transition model that assigns all exper-
mendment 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 factor at e-nodes (i.e. \( |X| \)), we draw on ideas from BO. In particular, BO has developed a number of effective policies for selecting experiments given prior experiment outcomes. Among the most effective is the maximum expected improvement (MEI) policy \( \pi_{ei} \), which selects the feasible experiment \( x \) that maximizes the EI function, i.e. \( \pi_{ei}(s) = \arg \max_{x \in X_s} E[I(y - y^*) \mid x,D] \), where \( y \sim P(y \mid x,D) \), \( D \) is the set of finished experiments at \( s \), and \( X_s \) is the set of feasible experiments at \( s \). For the GP models used in this work, this can be computed efficiently using a closed form for the EI of any \( x \). We leverage this idea by only considering the action selected by \( \pi_{ei} \), which eliminates action branching due to experiment selection in the tree. While this appears to be an extreme approximation, \( \pi_{ei} \) has consistently shown good performance and is quite effective in our experiments.

After the above simplifications, the remaining branching in the tree corresponds to action branching at \( s \)-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 \( s \)-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 \( s \)-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 \( d \) typically occurs 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 corresponding to 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 up 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 at a leaf. Here long term value corresponds to the potential improvement that could be obtained when 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 as a result of already running resource production actions. In other words, \( X^* \) maximizes \( G(X,y^*) = E(\max_{x \in X \cup X^*} I(y^* - y^*)) \) 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, 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 \( \tilde{X} \) that achieves an EI within a constant factor of \( G(X^*,y^*) \) for a wide class of BOR problems. We compute \( \tilde{X} \) using the following simple greedy algorithm. We initially set \( \tilde{X} = \emptyset \) and at each iteration adds an experiment to \( \tilde{X} \) 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 \( \tilde{X} \) we add an experiment \( x \) that maximizes \( G(\tilde{X} \cup \{x\}, y^*) \), subject to the constraint that \( \tilde{X} \cup \{x\} \) 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 idea of submodular optimization (Nemhauser, Wolsey, and Fisher 1978) via greedy algorithms. Here we only outline the main concepts as space precludes a full introduction to submodularity (see (Nemhauser, Wolsey, and Fisher 1978)). Viewing \( G(X,y^*) \) as a set function of \( X \), it is straightforward to show that \( G \) is a monotonically non-decreasing function of \( X \), meaning that adding an element to \( X \) will never decrease the objective. In addition, it is relatively straightforward to show that \( G(X,y^*) \) is submodular. Intuitively, this means that \( G \) has a diminishing returns property. That is, for any two sets \( X_1 \subseteq X_2 \), adding an experiment to \( X_1 \) will improve \( G \) by at least as much as adding the experiment to the superset \( X_2 \).

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

If there were no resource constraints then the standard \((1 - e^{-1})\) approximation result for submodular optimization would hold. 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 would like to characterize the constraints on the possible feasible sets \( \tilde{X} \) of experiments that could be produced. The simplest constraint is on the total number of experiments. Given knowledge of 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 resource constraint is $\sum_{x \in X} C(x) \leq r^t$. The key approximation result in (Calinescu et al. 2007) states that if constraints on sets $X$ can be defined as an intersection of $p$ matroids (or more generally as a $p$-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 $X^*$. Thus, it remains to characterize $p$ for classes of BOR problems.

**Definition 1 (Matroid)** A matroid is a pair $(\mathcal{X}, \mathcal{J})$ with a nonempty collection $\mathcal{J} \subseteq 2^\mathcal{X}$ of independent sets such that: 1) For all $A \subseteq B \subseteq \mathcal{X}$, if $B \in \mathcal{J}$, then $A \in \mathcal{J}$, and 2) For all $A, B \in \mathcal{J}$ with $|B| > |A|$ there exists $x \in B - A$ such that $A \cup \{x\} \in \mathcal{J}$.

The above cardinality constraint on $X$ 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 $\mathcal{J} \subseteq 2^\mathcal{X}$, contain all experiment subsets $X$ such that $\sum_{x \in X} C(x) \leq r$. If $C$ is $r$-uniform, then $\mathcal{J}$ can be represented as the intersection of $|R|$ matroids. If, in addition, $C$ is a partition cost function then $(\mathcal{X}, \mathcal{J})$ is a matroid.

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 result.

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

### 4 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 $X$ 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 $K(x, x') = \sigma \exp \left(-\frac{1}{2w} \| x - x' \|^2 \right)$, with signal variance $\sigma = 1$ and kernel width $w = 0.05$.

**Benchmarks.** Note that, to compare different BO algorithms, it is not practical to conduct real-world experiments in our motivating fuel-cell application or similar scientific domains, since we cannot afford repeated runs of the BO process. Instead, we designed a set of benchmarks that emulate the structure of our motivating fuel cell problem and other scientific domains. Each benchmark is modeled as having three different types of experiments, e.g. each using a different strain of bacteria. Each experiment type corresponds to one of three well-known BO benchmark functions:

**Rastrigin, Rosenbrock, and Cosines** defined over $[0, 1]^2$, which are qualitatively quite different. Thus, each experiment $x$ is described by two real-valued attributes and one three-valued discrete attribute that indicates its type. The duration of each experiment is $\tau = 6$ days.

**Resource structures** We consider two types of resource structure. The first, **Independent**, requires 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 run and the durations of the actions are specified by a vector $[\tau_1, \tau_2, \tau_3]$, where $\tau_i$ 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 unit 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 $[\tau_1, \tau_2, \tau_3, \tau_4]$. Unless otherwise specified, the time horizon is $H = 90$ days and there are 5 experimental labs ($L_\tau = 5$) and 2 resource production lines ($L_p = 2$). Our results report the average regret (over 50 runs) achieved by each method throughout the 90 day duration.

**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 select 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 first consider the impact of the resource depth $d$ of the search tree and the leaf evaluation function on our method. Fig. 1 shows results of our tree search method 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 the leaf evaluation were almost identical to $d = 1$). We see that when leaf evaluation is not used, increasing the depth $d$ improves performance (especially for the case $[11, 7, 5]$), which is 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 factor in performance is the leaf evaluation function, which provides a computationally cheaper way of evaluating the long-term impact of decisions compared to increasing the depth.

Comparing to Baselines. Figure 2 compares our method for $d = 1$ and leaf evaluation to the baselines. The first row shows results for the Independent resource scenario with production time settings $[5, 7, 11], [11, 7, 5], [8, 8, 8]$, and $[7, 11, 7]$. 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 gives results for the Shared resource setting where the additional shared resource always requires 8 days to produce. 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 4, we consider a version of the Independent scenario where the number of labs and production lines are doubled (i.e. $L_e = 10$ and $L_p = 4$) to observe the impact of increased experimental capacity. 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.
5 Summary
We introduced the new problem of Bayesian Optimization with Resources (BOR), which extends traditional BO to account for experiment resource requirements and the production of those resources. An online planning approach based on depth-limited tree search was introduced for selecting experiments and resource production actions. A key element of the approach was a novel state evaluation function for leaves of the tree, for which approximation results were shown based on the theory of optimizing submodular functions under matroid constraints. Our empirical results on a set of benchmarks with diverse resource requirements shows that the resulting approach significantly outperforms a variety of natural baselines and is even competitive with a policy that uses an optimal resource production oracle.

6 Acknowledgments
This research was supported by NSF grant IIS 1320943.

References
Marchant, R.; Ramos, F.; and Sanner, S. 2014. Sequential bayesian optimisation for spatial-temporal monitoring. In UAI.