Journal of Artificial Intelligence Research 1 (1993) 1-15 Submitted 6/2015; published 9/2016

# Budgeted Optimization with Constrained Experiments

**Javad Azimi** JAAZIMI@MICROSOFT.COM
*Microsoft, Sunnyvale, CA, USA*

**Xiaoli Z. Fern** XFERN@EECS.OREGONSTATE.EDU
*School of EECS, Oregon State University*

**Alan Fern** AFERN@EECS.OREGONSTATE.EDU
*School of EECS, Oregon State University*

## Abstract

Motivated by a real-world problem, we study a novel budgeted optimization problem where the goal is to optimize an unknown function $f(\cdot)$ given a budget by requesting a sequence of samples from the function. In our setting, however, evaluating the function at precisely specified points is not practically possible due to prohibitive costs. Instead, we can only request *constrained experiments*. A constrained experiment, denoted by $\mathbf{Q}$, specifies a subset of the input space for the experimenter to sample the function from. The outcome of $\mathbf{Q}$ includes a sampled experiment $\mathbf{x}$, and its function output $f(\mathbf{x})$. Importantly, as the constraints of $\mathbf{Q}$ become looser, the cost of fulfilling the request decreases, but the uncertainty about the location $\mathbf{x}$ increases. Our goal is to manage this trade-off by selecting a set of constrained experiments that best optimize $f(\cdot)$ within the budget. We study this problem in two different settings, the *non-sequential* (or *batch*) setting where a set of constrained experiments is selected at once, and the *sequential* setting where experiments are selected one at a time. We evaluate our proposed methods for both settings using synthetic and real functions. The experimental results demonstrate the efficacy of the proposed methods.

## 1. Introduction

This work is motivated by the experimental design problem of optimizing the power output of nano-enhanced microbial fuel cells. Microbial fuel cells (MFCs) (Bond & Lovley, 2003; Fan, Hu, & Liu, 2007; Park & ZeikusG, 2003; Reguera, McCarthy, Mehta, Nicoll, Tuominen, & Lovley, 2005) use micro-organisms to break down organic matter and generate electricity. For a particular MFC design, it is critical to optimize the biological energetics and the microbial/electrode interface of the system, which research has shown to depend strongly on the surface properties of the anodes (Park & ZeikusG, 2003; Reguera et al., 2005). This motivates the design of nano-enhanced anodes, where nano-structures (e.g., carbon nano-wire) are grown on the anode surface to improve the MFC's power output. Unfortunately, there is little understanding of the interaction between various possible nano-enhancements and MFC capabilities for different micro-organisms. Thus, optimizing the anode design for a particular application is largely guesswork. Our goal is to develop algorithms to aid this process.

Traditional experimental design, Bayesian optimization and response surface methods (Myers, Montgomery, & Anderson-Cook, 1995; Jones, 2001; Brochu, Cora, & De Freitas, 2010) commonly assume that the experimental inputs can be specified precisely and attempt

to optimize a design by requesting specific experiments. For example, requesting an anode to be tested with nano-wire of specific length and density. However, these parameters are unlike usual experimental control parameters (such as temperature) that can be easily set at precise values. Manufacturing nano-structures is rather an art and it is very difficult to achieve a precise parameter setting. Instead, it is more practical to request constrained experiments, which place constraints on these parameters. For example, we may specify intervals for the length and density of the nano-wire. Given such a request, nano-materials that satisfy the given set of constraints can be produced at some cost, which will typically increase with tighter constraints.

Note that a possible alternative to requesting constrained experiments would be to treat the nano-structure manufacturing parameters as the experimental inputs. Such inputs can be precisely specified, and hence standard methods can be used. However, this approach will not yield a satisfactory solution for our problem. In particular, the mapping between the manufacturing parameters and the produced nano-structures is extremely noisy. This makes it difficult to find the manufacturing parameters that optimize the expected MFC power output. Further, the scientists are primarily interested in learning what nano-structure properties optimize the MFC power output, rather than knowing the specific manufacturing parameters, which can vary significantly from lab to lab. Thus we focus on directly optimizing in the space of nano-structure properties via constrained experiments.

Based on the above motivation, in this paper, we study the associated budgeted optimization problem where, given a budget, our goal is to optimize an unknown function $f(\cdot)$ by requesting a set of constrained experiments. Solving this problem requires careful consideration of the trade-off between the cost and the uncertainty of a constrained experiment: weakening the constraints will lower the cost of an experiment, but increase the uncertainty about the location of the next observation. Prior work on experimental design, stochastic optimization, and active learning do not directly apply to constrained experiments because they typically assume precise experiments.

This problem can be formulated in the theoretical framework of partially observable Markov decision processes (POMDPs), where the optimal solution corresponds to finding an optimal POMDP policy. However, solving for optimal or even near-optimal policies is computationally intractable, even in the case of traditional optimization problems. This has led researchers to develop a variety of myopic policies in the context of traditional optimization, which have been observed to achieve good performance, even in comparison to more sophisticated, less myopic strategies (Moore & Schneider, 1995; Jones, 2001; Madani, Lizotte, & Greiner, 2004; Brochu et al., 2010).

Our problem can be considered in two different settings, *non-sequential* and *sequential*. In the non-sequential setting, which is also referred to as the *batch* setting, all constrained experiments must be selected at once. This setting is appropriate for applications where there are multiple experimental facilities and the experiments are too time consuming to be run sequentially. In contrast, the sequential setting allows us to request one constrained experiment at a time, and wait for the outputs of previous experiments before making the next request. The sequential setting has the advantage that it allows us to use the maximum available information for selecting each experiment, and is generally expected to outperform the non-sequential setting when the total running time is not a concern. In this paper, we study both settings.

For the non-sequential setting, we introduce a *non-decreasing submodular* objective function to select a batch of constrained experiments within the given budget. For a given set of constrained experiments, the objective measures its expected maximum output. We present a computationally efficient greedy algorithm that approximately optimizes the proposed objective.

For the sequential setting, we build on a set of classic myopic policies that have been shown to achieve good performance in traditional optimization (Moore & Schneider, 1995; Jones, 2001; Madani et al., 2004; Brochu et al., 2010) and introduce non-trivial extensions to make them applicable to constrained experiments.

We present experimental evaluations for both settings using synthetic functions and functions generated from real-world experimental data. The results indicate that, in both settings, our proposed methods outperform competing baselines.

The remainder of the paper is organized as follows. We will introduce the background and related work in Section 2. Section 3 describes the problem setup. The non-sequential setting is studied in Section 4. Section 5 introduces our proposed methods for selecting constrained experiments in the sequential setting. Section 6 presents the empirical evaluation of the proposed methods. We conclude the paper and discuss future work in Section 7.

## 2. Background and Related Work

Given an unknown black box function that is costly to evaluate, we are interested in finding the extreme point (minimizer or maximizer) of the function via a small number of function evaluations. To solve this problem, Bayesian Optimization (BO) approaches have been heavily studied (Jones, 2001; Brochu et al., 2010) and demonstrated significant promises. There are two key components in the basic framework of Bayesian Optimization. The first component is a probabilistic *model* of the underlying function that is built based on the prior information (i.e., the existing observed experiments). Gaussian process (GP) regression has been widely used in the literature of BO for this purpose (Brochu et al., 2010). For any unobserved point, a GP models its function output as a normal random variable, with its mean predicting the expected function output of the point and the variance capturing the uncertainty associated with the prediction.

The second key component of BO is the *selection criterion* that is used to determine what experiment to select based on the learned model. In the existing literature, various selection criteria have been proposed and most of them are a combination of exploring the unexplored input space of the function (i.e., areas of high variance) and exploiting the promising area (i.e., area with large mean). A selection criterion can be either sequential (Jones, 2001; Locatelli, 1997; Moore, Schneider, Boyan, & Lee, 1998; Srinivas, Krause, Kakade, & Seeger, 2010) in which only one experiment is asked at each iteration or non-sequential (Schonlau, 1997; Azimi, Fern, & Fern, 2010; Ginsbourger, Riche, & Carrarog, 2010) where a batch of experiments are requested at each iteration.

Below we review some of the most successful sequential selection criteria in the literature of BO. One of the first sequential policies is based on selecting the sample with maximum probability of improving (MPI) the best current observation, $y_{max}$ (assuming we want to maximize $f$), by a given margin $\alpha$ (Elder, 1992; Stuckman, 1988). Let the best current observation be $y_{max}$. The goal of MPI is to select the next experiment that has the highest

probability of producing an output no smaller than $(1+\alpha)y_{max}$. One issue of this approach is that the performance can often be sensitive to the value of the margin parameter $\alpha$ (Jones, 2001). For small values of $\alpha$, MPI will focus on the most promising area at first and then move onto unexplored areas. In contrast, for large values of $\alpha$, MPI will primarily explore and converge very slowly. Selecting a proper value for $\alpha$ can be challenging in practice. The *maximum expected improvement* (MEI) (Locatelli, 1997) criterion avoids this issue and selects the experiment that directly maximizes the expected improvement over the current best observation. Heuristics based on upper-confidence bounds have also been explored (Srinivas et al., 2010), which has been shown to be competitive with MEI given appropriate parameter selection. However, selecting the best parameters for a particular application empirically can be a challenge. An alternative approach that has received attention is Thompson Sampling (Chapelle & Li, 2011), which is a randomized strategy for managing the exploration-exploitation trade-off. This approach first samples the underlying uncertainty, in our case the unknown function $f$, and then returns the experiment that maximizes the sample. In this work, we have focused on extending the above deterministic methods for BO to constrained experiments. Extending alternatives such as Thompson sampling is a potentially interesting future direction.

Recently, researchers have begun to consider non-sequential or batch Bayesian optimization (Azimi et al., 2010; Ginsbourger et al., 2010; Desautels, Krause, & Burdick, 2014), which selects multiple experiments at once. Non-sequential BO is considered more appropriate for applications where there is a need and capability to run multiple experiments simultaneously. One non-sequential approach (Azimi et al., 2010) selects $k > 1$ experiments at once by matching the behavior of executing a given sequential policy (e.g., MEI) for $k$ steps. In another non-sequential approach (Ginsbourger et al., 2010), the authors tried to select a batch of experiments that will lead to the highest expected improvement. However, it was shown that the expected improvement over a set of jointly normal random variables does not have any closed form solution when $k > 2$, nor it can be solved efficiently using numerical methods. Instead, simple heuristics were proposed to approximate the expected improvement and select a batch accordingly. More recently, an algorithm based on upper-confidence bounds has also been introduced (Desautels et al., 2014), which is computationally cheaper than prior work but requires careful parameter selection.

Note that all of the aforementioned approaches assume that the unknown function we aim to optimize can be sampled at precisely specified points, making them unsuitable for tasks such as our motivating nano application, where sampling the function at exact locations is impractical. The proposed sequential approaches in this paper, have been previously presented in less detail (Azimi et al., 2010). In this paper, we provide a more complete and formal description of the sequential approaches with additional empirical results. In addition, we introduce and evaluate a batch selection algorithm that chooses a batch of constrained experiments at each iteration.

## 3. Problem Setup

Let $\mathcal{X} \subseteq R^d$ be a $d$-dimensional input space, where each dimension $i$ is bounded in $[A_i, B_i]$. We often refer to the elements of $\mathcal{X}$ as experiments. We assume there is an unknown real-valued function $f(\mathbf{x}) : \mathcal{X} \rightarrow \Re$, which represents the expected value of the dependent

variable after running experiment $\mathbf{x}$. In our motivating application, $f(\mathbf{x})$ is the expected power output produced by a particular nano-structure $\mathbf{x}$. Conducting an experiment $\mathbf{x}$ produces a noisy outcome $y = f(\mathbf{x}) + \epsilon$, where $\epsilon$ is a noise term.

In traditional optimization settings (Jones, 2001; Brochu et al., 2010), the goal is to find an $\mathbf{x} \in \mathcal{X}$ that approximately optimizes $f(\cdot)$ by requesting a set of experiments and observing their outcomes. Since sampling the function at exactly specified points is prohibitively expensive in our application, we request *constrained experiments*, which define a subset of experiments in $\mathcal{X}$. Specifically, we define a constrained experiment as a *hyper-rectangle* in $\mathcal{X}$, denoted by $\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_d)$, where $\mathbf{q}_i = (a_i, b_i)$ with $A_i \leq a_i < b_i \leq B_i$ defines the range of values that is considered admissible for each input dimension $i$. Note that for computational reasons, in this work we consider a discretized input space, where each input dimension is divided into equal-sized intervals. As such, a constrained experiment $\mathbf{Q}$ will indicate for each dimension $i$ the first (specified by $a_i$) and the last (specified by $b_i$) intervals to be included in the hyper-rectangle. For the remainder of this paper, we will interchangeably use the terms *constrained experiment*, *hyper-rectangle* and *query*.

Given a constrained experiment $\mathbf{Q}$, the experimenter will first construct an experiment $\mathbf{x}$ (we assume that $\mathbf{x}$ can be precisely measured after being produced) that satisfies the given constraints of $\mathbf{Q}$, run the experiment, and return the noisy observation of $f(\mathbf{x})$. Note that $\mathbf{x}$ is a random variable given $\mathbf{Q}$, and we assume this conditional distribution, $p_{\mathbf{x}}(\mathbf{x}|\mathbf{Q})$, is known *a priori* as part of the problem inputs. More precisely, for any query $\mathbf{Q}$, the experimenter will return a 2-tuple $(\mathbf{x}, y)$, where:

- $\mathbf{x} = (x_1, x_2, \cdots, x_d)$ is an experiment that satisfies the constraints of $\mathbf{Q}$,
- $y$ is the noisy observation of the function $f(\cdot)$ at $\mathbf{x}$, $y = f(\mathbf{x}) + \epsilon$.

In practice, the cost $c$ of fulfilling a constrained experiment can be variable depending on the size of the hyper-rectangle. In particular, higher cost will be associated with tighter constraints or smaller hyper-rectangles. We assume that this cost is modeled by a deterministic function $f_c(\cdot)$, which is provided to us as part of the inputs. For example, in our motivating application, $f_c(\cdot)$ is dominated by the time required to produce the nano material that satisfies the given constraints, which is inversely correlated with the size of the constraints. In addition, we must operate within a total budget $B$. Thus, the objective is to find a set of queries within budget $B$ that leads to the best estimate of the maximizer of the function over the input space $\mathcal{X}$.

To summarize, the inputs to our problem include a set of prior experiments $\mathcal{D}$ (which could potentially be empty), a budget $B$, a deterministic cost function $f_c(\cdot)$ of fulfilling a constrained experiment $\mathbf{Q}$, and a conditional probability density function $p_{\mathbf{x}}(\mathbf{x}|\mathbf{Q})$ of the specific experiment $\mathbf{x}$ generated for any given constrained experiment $\mathbf{Q}$.

Given the inputs, our task is to select a set of constrained experiments $\mathcal{Q} = \{\mathbf{Q}_1, \mathbf{Q}_2, \cdots, \mathbf{Q}_k\}$ whose total cost is within budget $B$. Running the selected constrained experiments will result in a set of $k$ tuples $(\mathbf{x}_i, y_i)_{i=1}^k$, with which we must determine a final output $\mathbf{x}^* \in \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, which is our prediction of the maximizer of $f(\cdot)$ among all *observed* experiments. Note that we restrict ourselves to returning an experiment that was *actually observed*, even in cases where we might predict some other non-observed experiment to be better. This formulation matches the objective of our motivating application to produce a

good nano-structure $\mathbf{x}^*$ using the given budget, rather than to make a prediction of what nano-structure might be good.

We study this problem in two different settings, *non-sequential* (or *batch*) and *sequential*. In the non-sequential setting, we must decide the entire set of queries at the same time. In contrast, the sequential setting requests constrained experiments sequentially one at a time: only after receiving the result of the previous request, another query is selected and presented to the experimenter. This procedure is repeated until we reach the budget limit. In the following two sections, we will introduce our proposed solutions for both settings.

## 4. Non-Sequential Approach

In this section, we consider the non-sequential setting, in which we must select the entire batch of queries $\mathcal{Q}$ within the given budget $B$ at once. Note that in general these batches can be multi-sets that have repeated queries, which may be desirable in noisy settings. This is also called the *non-adaptive* (Goel et al., 2006; Krause et al., 2008) or *Batch* (Azimi et al., 2010) setting. This setting is commonly used in applications where we must start multiple experiments at once and cannot wait for the outputs of the previous queries to decide the next queries (Tatbul et al., 2003).

### 4.1 The Objective Function

Let $\mathcal{Q}_B$ be the set of feasible solutions such that for any $\mathcal{Q} \in \mathcal{Q}_B$ the total cost of $\mathcal{Q}$ is no greater than the budget $B$. Our goal is to find the optimal multi-set of queries $\mathcal{Q}^* = \{\mathbf{Q}_1^*, \mathbf{Q}_2^*, \cdots, \mathbf{Q}_k^*\} \in \mathcal{Q}_B$. To define what we mean by optimal, let us consider the outcome of the queries, which are a set of tuples: $(\mathbf{x}_i, y_i), i = 1, 2, \ldots, k$. The $\mathbf{x}_i$'s are the experiments produced by the experimenter given the queries and the $y_i$'s represent their experimental output (i.e., the noisy observation of $f(\mathbf{x}_i)$). We will then select a final output $\mathbf{x}^* \in \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ that is believed to achieve the maximal $f(\cdot)$ value. As such, for any $\mathcal{Q} \in \mathcal{Q}_B$, we can measure how good $\mathcal{Q}$ is based on the maximal $y$ value resulting from this set of queries. Specifically, this is captured by:

$$J(\mathcal{Q}) = \mathbb{E}_{(\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{Q}|})} \left[ \mathbb{E}_{(y_1, \cdots, y_{|\mathcal{Q}|})} \left[ \max \left\{ y_1, \ldots, y_{|\mathcal{Q}|} \right\} \middle| \mathcal{D}, (\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{Q}|}) \right] \right], \tag{1}$$

where the first expectation is taken over all possible values of the $\mathbf{x}_i$'s, which represent the individual experiments created for each query in $\mathcal{Q}$, and the second expectation is taken over all possible $y_i$'s, which represents the experimental outcomes of the $\mathbf{x}_i$'s. As mentioned previously, the $\mathbf{x}_i$'s are distributed according to $p_{\mathbf{x}}(\mathbf{x}_i|\mathbf{Q}_i)$, which is part of the inputs. The distribution of $y_i$'s given the $\mathbf{x}_i$'s depends on the posterior distribution of $f(\cdot)$ given $\mathcal{D}$. In our work, we use Gaussian processes to model the distribution of $f(\cdot)$. Consequently, the set of $y_i$'s are jointly normal conditioned on all $\mathbf{x}_i$'s and $\mathcal{D}$.

Since our input space is discretized, we can enumerate all possible constrained experiments and denote them as $\mathcal{Q}_M = \{\mathbf{Q}_1, \mathbf{Q}_2, ..., \mathbf{Q}_M\}$, where $M$ is the total number of possible constrained experiments, and let $c_1, \ldots, c_M$ be their corresponding cost (i.e., $c_i = f_c(\mathbf{Q_i})$). Let $S \subseteq \mathcal{S} = \{1, ..., M\}$ be a subset of indices and $\mathcal{Q}_S$ denote the collection of queries indexed by $S$, i.e., $\mathcal{Q}_S = \{\mathbf{Q}_i : i \in S\}$. Our goal can then be stated as selecting an $S$ such that the corresponding $\mathcal{Q}_S$ maximizes the objective (Equation 1) subject to the constraint

$\sum_{i \in S} c_i \leq B$. Unfortunately, optimizing this objective is intractable due to the combinatorial nature of the problem and exponentially many possible solutions to consider. Below we will reformulate the objective to demonstrate that it is a *non-decreasing submodular* set function and introduce an algorithm with an approximation guarantee.

Specifically, we will consider a slightly different but equivalent view of the querying process. So far our view is that after $S$ is chosen, each query $\mathbf{Q} \in \mathcal{Q}_S$ will result in an experiment $\mathbf{x}$, which can be viewed as a random sample drawn from the distribution $p_{\mathbf{x}}(\mathbf{x}|\mathbf{Q})$ (note that in this work $p_{\mathbf{x}}$ is uniform within the hyper-rectangle defined by the query). From the process point of view, it clearly does not matter whether this random draw happens after $\mathbf{Q}$ is chosen, or at the very beginning of the whole process before $\mathbf{Q}$ is chosen. Following this reasoning, we could assume that for every possible query in $\mathcal{Q}_M$, a random experiment is drawn at the very beginning of the process and the results are stored and used later when $S$ is selected. Let $\mathcal{X}_M = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ denote the random variables representing the outcome of the random draw for $\mathbf{Q}_1, ..., \mathbf{Q}_M$ respectively. The objective can be then reformulated as:

$$J(S) = \mathbb{E}_{\mathcal{X}_S}\left[\mathbb{E}_{(y_1,\ldots,y_{|S|})}\left[\max\left\{y_1, \ldots, y_{|S|}\right\} \middle| \mathcal{D}, \mathcal{X}_S\right]\right], \tag{2}$$

where $\mathcal{X}_S = \{\mathbf{x}_i : i \in S\}$ is the subset of $\mathcal{X}_M$ defined by $S$, and the $y_i$'s are the noisy outcomes of the $\mathbf{x}_i$'s in $\mathcal{X}_S$.

### 4.2 Approximation Algorithm

Since standard batch Bayesian Optimization is a special case of optimizing $J(S)$, the hardness of optimizing $J(S)$ follows from NP-hardness of Bayesian Optimization. Thus, below we will show that $J(S)$ is a non-decreasing submodular set function and present an algorithm with a bounded approximation factor.

**Definition 1.** *Suppose $\mathcal{S}$ is a finite set, $g : 2^{\mathcal{S}} \to \mathbb{R}^+$ is a submodular set function if for all $S_1 \subseteq S_2 \subset \mathcal{S}$ and $\mathbf{x} \in \mathcal{S} \setminus S_2$, it holds that $g(S_1 \cup \{\mathbf{x}\}) - g(S_1) \geq g(S_2 \cup \{\mathbf{x}\}) - g(S_2)$.*

Thus, a set function is submodular if adding an element to a smaller set provides no less improvement than adding the element to a larger superset. Also, a set function is non-decreasing if for any set $S$ and element $\mathbf{x}$ we have $g(S) \leq g(S \cup \{\mathbf{x}\})$.

To show that $J(S)$ is submodular, we will rewrite the objective function by defining $J_{\mathcal{X}_M}(S)$ to be the inner expectation of Equation 2 for a fixed realization of random variable $\mathcal{X}_M$:

$$J_{\mathcal{X}_M}(S) = \mathbb{E}_{(y_1,\ldots,y_{|S|})}\left[\max\left\{y_1, \ldots, y_{|S|}\right\} \middle| \mathcal{D}, \mathcal{X}_S\right].$$

**Lemma 1.** *For any given $\mathcal{X}_M$, $J_{\mathcal{X}_M}(S)$, which returns the expected maximum over a set of jointly distributed random variables, is a monotonically non-decreasing submodular set function.*

The proof is in the Appendix.

The proposed objective function, $J(S) = \mathbb{E}_{\mathcal{X}_M}\left[J_{\mathcal{X}_M}(S)\right]$ takes the expectation of $J_{\mathcal{X}_M}(S)$ over all possible values of $\mathcal{X}_M$. Because $J_{\mathcal{X}_M}(S)$ is non-decreasing, it is easy to verify that $J(S)$ is also non-decreasing. Further note that the set of submodular functions is

closed under expectation, we can thus conclude that the proposed objective, $J(S)$, is a non-decreasing submodular function.

We now present our proposed algorithm for optimizing $J(S)$. The inputs to our algorithm include the set of all possible constrained experiments, $\mathcal{Q}_M = \{\mathbf{Q}_1, ..., \mathbf{Q}_M\}$, their associated costs $c_1, ..., c_M$, and a total budget $B$, and the output is a subset $S \subseteq \mathcal{S} = \{1, ..., M\}$ such that $\sum_{i \in S} c_i \leq B$. We first introduce a simple greedy algorithm, which begins with an initial empty set $S = \varnothing$ and greedily adds one constrained experiment (its index) at a time until the total cost of $S$ reaches $B$. In each step, let $S$ be the current set and $C$ be the total cost of $S$, the greedy algorithm selects an index $i^* \in \mathcal{S}$ such that:

$$i^* = \underset{i \notin S; c_i \leq B - \mathcal{C}}{\operatorname{argmax}} \frac{J(S \cup i) - J(S)}{c_i}. \tag{3}$$

In other words, at each step, the algorithm greedily selects a new constrained experiment that is within the budget and leads to the largest cost-normalized improvement of the objective.

It is known that this simple greedy algorithm does not have any bounded approximation factor (Khuller, Moss, & Naor, 1999). Previous work (Khuller et al., 1999; Krause & Guestrin, 2005) introduced a small change to the greedy algorithm that provides us with a bounded approximation factor. In particular, one just needs to consider, as an alternative to the output of the greedy algorithm, the single query that is within the budget and achieves the best objective value (denoted by $S_a$). By comparing this alternative with the output of the greedy algorithm, we are guaranteed to achieve a bounded approximation factor. The complete algorithm is summarized in Algorithm 1. The approximation bound for this algorithm follows from the following theorem.

**Theorem 1.** *(Khuller et al., 1999) Let $J(\cdot)$ be a monotonically non-decreasing submodular set function such that $J(\varnothing) = 0$, and $S^*$ is the output of our Algorithm 1. Suppose* OPT *is the optimal solution, the following bound is guaranteed*

$$\begin{aligned} J(S^*) &\geq \frac{1}{2}\left[1 - \left(1 - \frac{1}{|S^*| + 1}\right)^{|S^*|+1}\right] J(\text{OPT}) \\ &\geq \frac{1}{2}\left(\frac{e-1}{e}\right) J(\text{OPT}). \end{aligned} \tag{4}$$

The dominating factor of the run time is the linear dependence on $M$, the number of possible queries. Note that in the discretized setting that we consider, $M$ will be exponential in the number of dimensions. In the scientific application domains that motivate our work, the number of dimensions is typically small (e.g., 2 or 3). However, when working with a fine resolution discretization, the computation time can still be significant. To address this issue, in the next section we describe a simple strategy for soundly pruning candidate queries from consideration, which yields significant speedups. Problems with significantly higher dimensions, however, will require continuous rather than discretized optimization.

### 4.3 Accelerated Greedy Algorithm

In this section, following prior applications of submodular optimization (e.g. (Krause & Guestrin, 2005)), we describe an accelerated greedy algorithm, which yields significant gains in computational efficiency. At each greedy step, let $S$ represent the set of queries that have been selected so far. To make another greedy selection, we need to compute the cost normalized *incremental difference* $\delta(i) = \frac{J(S \cup i) - J(S)}{c_i}$ for each candidate query $i$, such that $i \notin S$ and $c_i \leq B - \mathcal{C}$. This computation can be expensive because the number of candidate queries is often very large. Fortunately, by carefully maintaining the normalized incremental differences calculated in the first greedy step, we can avoid recomputing a large majority of them in later iterations.

Specifically, the first iteration will compute the $\delta(i)$ values for all $i \in \mathcal{S}$. We then sort them in decreasing order based on their $\delta$ values, and select the first query and remove it from the list. For the next iteration, we move on to the next query in the sorted list and recompute its $\delta$ value. If the value remains the largest, we will immediately select this query and remove it from the list, and proceed to the next iteration without recomputing any other $\delta$ values. Otherwise, we proceed to evaluate the next query in the sorted list until finding one whose recomputed $\delta$ value is greater than the other stored values and select that query. The submodular property of our objective guarantees that this approach makes the same choices as the full greedy algorithm, but effectively avoids a large number of computations of the $\delta$ values in practice. The proposed accelerated algorithm is summarized in Algorithm 2.

**Algorithm 1** The Greedy Non-Sequential Algorithm

**Input:** $\mathcal{D}, B > 0, \{\mathbf{Q}_1, ..., \mathbf{Q}_M\}, \{c_1, ..., c_M\}$
**Output:** A set of indices $S \subseteq \mathcal{S} = \{1, ..., M\}$ such that $\sum_{i \in S} c_i \leq B$

- $i^* = \text{argmax}_{i \in \mathcal{S}, c_i \leq B} J(\{i\})$
$S_a \leftarrow \{i^*\}$
- $S \leftarrow \varnothing, \mathcal{C} \leftarrow 0$
**while** $(\mathcal{C} < B)$ **do**
  Select $i^*$ such that: $\quad i^* = \underset{i \notin S, c_i \leq B - \mathcal{C}}{\text{argmax}} \frac{J(S \cup \{i\}) - J(S)}{c_i}$
  - $\mathcal{C} \leftarrow \mathcal{C} + c_{i^*}$
  - $S \leftarrow S \cup \{i^*\}$
**end while**
**if** $J(S) \geq J(S_a)$ **then**
  - **return** $S$
**else**
  - **return** $S_a$
**end if**

### 4.4 Computation of the Expected Maximum

For any given set $S$, to compute $J(S)$, we need to compute the expected maximum value of a set of jointly distributed random variables $(y_1, ..., y_{|S|})$. Unfortunately, the expected maximum of a set of dependent random variables generally does not have a closed-form

**Algorithm 2** Accelerated Greedy Algorithm

**Input:** $\mathcal{D}, B > 0, \{\mathbf{Q}_1, ..., \mathbf{Q}_M\}, \{c_1, ..., c_M\}$

**Output:** A set of indices $S \subseteq \mathcal{S} = \{1, ..., M\}, s.t., \sum_{s \in S} c_i \leq B$

- $i^* = \text{argmax}_{i \in \mathcal{S}, c_i \leq B} J(\{i\})$
- $S_a \leftarrow \{i^*\}$
-$S \leftarrow \varnothing, \mathcal{C} = 0, \delta(i) = J(\{i\})/c_i$, for $i = 1, \ldots, M$

**while** $(\mathcal{C} < B)$ **do**

  **while** true **do**

    Set $z = \text{argmax}_{i: i \in \mathcal{S} \setminus S} \delta(i), c_i \leq B - \mathcal{C}$, then re-calculate $\delta(z)$ such that

$$\delta(z) = \frac{J(S \cup \{z\}) - J(S)}{c_z}$$

    **if** $\delta(z) \geq \max_{i \in \mathcal{S} \setminus \{S \cup z\}} \delta(i)$ **then**

      **Break**

    **end if**

  **end while**

  - $\mathcal{C} \leftarrow \mathcal{C} + c_z$, $S \leftarrow S \cup \{z\}$

**end while**

**if** $J(S) \geq J(S_a)$ **then**

  - **return** $S$

**else**

  - **return** $S_a$

**end if**

solution (Ross, 2008). Instead, we use a Monte-Carlo simulation approach for computing the expected maximum value. Specifically, given $S$, to compute $J(S)$, we first sample one experiment for each $\mathbf{Q} \in \mathcal{Q}_S$, resulting in $\{\mathbf{x}_1, ..., \mathbf{x}_{|S|}\}$. We then sample the $y_i$'s from their posterior distribution $p_y(y_1, ..., y_{|S|}|\mathbf{x}_1, ..., \mathbf{x}_{|S|}, \mathcal{D})$ and take the maximum of the sampled $y_i$'s. This is repeated $l$ independent times and the expected maximum is then obtained by averaging across the $l$ results. Note that our computation of the expected maximum value with simulation will not be exact. Denoting the simulated results by $\hat{J}$, standard Chernoff bounds can be used to bound the error of $\hat{J}(S)$ with high probability. Assuming a bounded error, that is $|J(S) - \hat{J}(S)| \leq \epsilon$ for some $\epsilon \geq 0$, the following theorem holds for non-decreasing submodular objective functions:

**Theorem 2.** *(Krause & Guestrin, 2005) Let $J(\cdot)$ be a non-decreasing submodular function and $S^* = \max_{S:c(S)\leq B} J(S^*)$ be the cost constrained optimizer of $J$. For any $\hat{J}(\cdot)$ such that $|J(S) - \hat{J}(S)| \leq \epsilon$ for all $S$, if Algorithm 1 is run using $\hat{J}$ in place of $J$, then the returned set $\hat{S}$ satisfies the following approximation bound, where $c_{min} = \min_i c_i$:*

$$J(\hat{S}) \geq \frac{1}{2}\left(\frac{e-1}{e}\right) J(S^*) - \frac{1}{2}\left(\frac{c(S^*)}{c_{min}} + |S^*|\right)\epsilon. \tag{5}$$

## 5. Sequential Approach

In this section, we consider the sequential setting (Deshpande et al., 2004; Silberstein et al., 2006) where each query is selected one at a time after the result for the previous query becomes available. This is the most commonly studied setting for Bayesian Optimization and is appropriate for many applications where there is only a single experimental facility to process the queries.

The inputs to our problem remain the same, which include $B$, the total budget, $f_c(\cdot)$ the cost function, $p_{\mathbf{x}}(\mathbf{x}|Q)$, the distribution of the constructed experiment $\mathbf{x}$ given query $\mathbf{Q}$, and $\mathcal{D}$, the set of observed experiments. In the sequential setting, given the inputs we must request a sequence (one at a time) of constrained experiments whose total cost is within the budget.

Leveraging the extensive body of research on traditional Bayesian Optimization, we design our sequential selection policies by extending a number of well-established myopic sequential selection policies from the literature. Most existing policies for traditional Bayesian Optimization can be viewed as defining a greedy heuristic that assigns a score to each candidate experiment $\mathbf{x}$ based on the current *experimental state*, which we denote by $(\mathcal{D}_c, B_c)$, where $\mathcal{D}_c$ represent the current set of prior experiments, and $B_c$ represents the current remaining budget. As reviewed in Section 2, many of the existing heuristics have been observed to perform well for traditional Bayesian Optimization problems. Unfortunately they cannot be directly used for our problem since they select individual specific experiments rather than constrained experiments, as we require.

### 5.1 Model-Free Policies

Model-free policies do not consider statistical models of the data in making the selection. In this work we consider two model-free policies, *Round Robin* (RR) and *Biased Round Robin*

(BRR), which are motivated by previous work on budgeted multi-armed bandit problems (Lizotte et al., 2003; Madani et al., 2004).

#### 5.1.1 Round Robin (RR)

In the multi-armed bandit setting, the RR policy seeks to keep the number of pulls of each arm as uniform as possible. In the context of constrained experiments, we apply the same principle to keep the experiments as uniformly distributed as possible in the experimental space $\mathcal{X}$. Given the current experimental state $(\mathcal{D}_c, B_c)$, we define the RR policy to return the largest hyper-rectangle or the least costly query $\mathbf{Q}$ that does not contain any previous experiment in $\mathcal{D}_c$. If the cost $\mathbf{Q}$ exceeds the current budget $B_c$, we return the constrained experiment with cost $B_c$ that contains the fewest experiments in $\mathcal{D}_c$. Ties are broken randomly. Note that in RR the outputs of previous queries do not have any effect in selecting the next queries. However, the exact location of the previous experiments do have a significant effect in the next query selection. Therefore, we can not consider RR as a non-sequential approach.

#### 5.1.2 Biased Round Robin (BRR)

*BRR* policy behaves identically to *RR*, except that it repeats the previously selected constrained experiment as long as the outcome of the constrained experiment has improved the performance and it does not exceed the budget. In particular, given the current experimental state $(\mathcal{D}_c, B_c)$, the query $\mathbf{Q}$ is repeated as long as it results in an outcome that improves over the current best outcome in the set $\mathcal{D}_c$, and $f_c(\mathbf{Q}) \leq B_c$. Otherwise, the *RR* policy is followed. This policy is analogous to *BRR* in multi-armed bandit problems (Madani et al., 2004) where an arm is pulled as long as it has a positive outcome.

### 5.2 Model-Based Policies

For model-based policies, it is assumed that a conditional posterior distribution $p(y|\mathcal{D}_c, \mathbf{x})$ over the outcome $y$ of each individual experiment $\mathbf{x} \in \mathbf{Q}$ is learned from the set of currently observed experiments $\mathcal{D}_c$. Existing model-based myopic policies for traditional experimental design typically select the experiment $\mathbf{x}$ that maximizes certain heuristics computed from statistics of the posterior (Jones, 2001). The heuristics provide different mechanisms for trading off between exploration (probing unexplored regions of the experimental space) and exploitation (probing areas that appear promising) given $\mathcal{D}_c$. Note that the experiment $\mathbf{x}$ is a fixed and known point in the experimental design literature before running the real experiment in the lab since it is assumed that we can ask for a particular fixed point.

However, in our constrained experiment application, we ask for a hyper-rectangle query $\mathbf{Q}$ rather than a fixed experiment point $\mathbf{x}$. Therefore the conditional posterior distribution for each constrained experiment $\mathbf{Q}$ is defined as $\bar{p}(y|\mathbf{Q}, \mathcal{D}_c) \triangleq \mathbb{E}_{\mathbf{x}|\mathbf{Q}}\left[p(y|\mathbf{x}, \mathcal{D}_c)\right]$. This definition corresponds to the process of drawing an experiment $\mathbf{x}$ from $\mathbf{Q}$ and then drawing an outcome for $\mathbf{x}$ from $p(y|\mathbf{x}, \mathcal{D}_c)$. $\bar{p}(\cdot)$ effectively allows us to treat constrained experiments as if they were individual experiments in a traditional optimization problem. Thus, we can define heuristics for constrained experiments by computing the same statistics of the posterior $\bar{p}(\cdot)$, as used in traditional optimization. In this work we consider four such heuristics.

#### 5.2.1 Maximum Mean (MM)

In the context of traditional optimization with individual experiments, the MM heuristic, also known as PMAX (Moore & Schneider, 1995; Moore et al., 1998; Schneider & Moore, 2002), simply selects the experiment that has the largest expected outcome according to the current posterior. In our constrained experiments, the MM heuristic computes the expected outcome of a given query according to the current posterior $\bar{p}(\cdot)$. The MM of any arbitrary query $\mathbf{Q}$ is computed as follows:

$$\text{MM}(\mathbf{Q}|\mathcal{D}_c) = \mathbb{E}\left[y|\mathbf{Q}, \mathcal{D}_c\right], \text{ where } y \sim \bar{p}(y|\mathbf{Q}, \mathcal{D}_c). \tag{6}$$

MM is purely an exploitative heuristic and has the weakness that it can often be too greedy and get stuck in a poor local maximum point before exploring the rest of the experimental space.

#### 5.2.2 Maximum Upper Interval (MUI)

The MUI heuristic, also known as IEMAX in previous work (Moore & Schneider, 1995; Moore et al., 1998; Schneider & Moore, 2002), attempts to overcome the greedy nature of MM by exploring areas with non-trivial probability of achieving a good result as measured by the upper bound of the 95% confidence interval of output prediction. Thus, the MUI heuristic for any arbitrary constrained experiments $\mathbf{Q}$ is calculated as follow (assuming that Gaussian process is used for estimating the posterior distribution of $f(\cdot)$):

$$\text{MUI}(\mathbf{Q}|\mathcal{D}_c) = \mathbb{E}\left[y|\mathbf{Q}, \mathcal{D}_c\right] + 1.96\sqrt{Var[y|\mathbf{Q}, \mathcal{D}_c]}, \text{ where } y \sim \bar{p}(y|\mathbf{Q}, \mathcal{D}_c). \tag{7}$$

Intuitively, MUI will aggressively explore untouched regions of the experimental space since the outcomes in such regions will have high posterior variance. However, as experimentation continues for a long time and uncertainty decreases, MUI will focus on the most promising areas and behaves like MM. Note that MUI is a specific case of a more general heuristic GP-UCB (Cox & John, 1992, 1997), where the constant 1.96 is replaced by a varying parameter that results in certain theoretical guarantees. Empirically GP-UCB has been observed to perform comparatively to the MEI heuristic which we introduce later in this section.

#### 5.2.3 Maximum Probability of Improvement (MPI)

In the context of individual experiments, the MPI heuristic corresponds to selecting the experiment that has the highest probability of generating an outcome $y$ that outperforms the best current outcome in $\mathcal{D}_c$. An issue with the basic MPI strategy is that it has a tendency to behave similar to MM and focuses on the areas that currently look promising, rather than exploring unknown areas. The reason for this behavior is that the basic MPI does not take into consideration the amount of improvement over the current best outcome. In particular, it is typical for the posterior to assign small amounts of variances to the outcomes in well explored regions. It means while there might be a good probability of observing a small amounts of improvement, the probability of a substantial improvement is small. Hence, it is common to consider the use of a margin $\alpha$ when using MPI, which we will refer to as MPI($\alpha$). Let $y_c^*$ represent the current best outcome that was observed

in $\mathcal{D}_c$, then $\text{MPI}_\alpha(\mathbf{Q}|\mathcal{D}_c)$ is equal to the probability that the outcome of the constrained experiment $\mathbf{Q}$ will be greater than $((1+\alpha)y_c^*)$ (assuming non-negative $y_c^*$ values). The MPI heuristic is given by:

$$\text{MPI}_\alpha(\mathbf{Q}|\mathcal{D}_c) = p\left(y \geq (1+\alpha)y_c^*|\mathbf{Q}, \mathcal{D}_c\right), \qquad \text{where } y \sim \bar{p}(y|\mathbf{Q}, \mathcal{D}_c). \tag{8}$$

The MPI($\alpha$) heuristic is sensitive to the $\alpha$ margin parameter. Adjusting the margin $\alpha$ from small to large causes the heuristic to change its behavior from more exploitive to more explorative.

#### 5.2.4 Maximum Expected Improvement (MEI)

The *maximum expected improvement* (Locatelli, 1997) heuristic seeks to improve on the basic MPI heuristic without requiring a margin parameter $\alpha$. Rather than focus on the probability of improvement, it considers the expected amount of improvement according to the current posterior. In particular let $I(y, y_c^*) = \max(y - y_c^*, 0)$. Then, the MEI heuristic is defined as:

$$\text{MEI}(\mathbf{Q}|\mathcal{D}_c) = \mathbb{E}_y\left[I(y, y_c^*)|\mathcal{D}_c, \mathbf{x}\right], \text{ where } y \sim \bar{p}(y|\mathbf{Q}, \mathcal{D}_c). \tag{9}$$

### 5.3 Cost-Sensitive Policies

The above introduced sequential heuristics do not take the variable cost of a constrained experiment into account. If only the heuristic value is used as our selection criterion, the most costly constrained experiment might be selected. In fact, the nature of our heuristics will typically assign the highest score to the constrained experiments that are maximally constrained and centered around the individual experiment that maximizes the heuristic. Unfortunately, such constrained experiments are also maximally costly. More generally, assume that the cost of a constrained experiment $\mathbf{Q}$ monotonically decreases with the size of its region of support, which is the most natural case to consider. It is easy to show that for all of our heuristics, the value of a constrained experiment $\mathbf{Q}$ is monotonically non-decreasing with respect to the cost of $\mathbf{Q}$. This is true when reducing the size of a constrained experiment would remove the points which have the heuristic values less than the constrained experiment value.

Thus, maximizing the above defined heuristics leads to the selection of the most costly experiments, which might consume more budget than is warranted. This suggests that there is a fundamental trade-off between the heuristic values and the cost of the constrained experiments that we must address. Below, we introduce two approaches that attempt to address this trade off by defining cost-sensitive policies from the cost insensitive heuristics.

#### 5.3.1 Cost Normalized (CN) Policies

Cost normalized policies have been widely used in budgetd optimization settings where the costs are non-uniform across experiments, for example, (Krause et al., 2008; Snoek, Larochelle, & Adams, 2012). It simply selects the constrained experiment that achieves the highest expected improvement *per unit cost*, or rate of the improvement.

Suppose $H$ is our heuristic. We can define a corresponding CN policy for any heuristic on constrained experiment $\mathbf{Q}$ given the current experimental state $\{\mathcal{D}_c, B_c\}$ as follows:

$$\mathrm{CN}_H(\mathcal{D}_c, B_c) = \underset{\mathbf{Q}: f_c(Q) < B_c}{\operatorname{argmax}} \left\{ \frac{H(\mathbf{Q}|\mathcal{D}_c)}{f_c(Q)} \right\}, \tag{10}$$

where $H(\mathbf{Q}|\mathcal{D}_c)$ assigns a score to constrained experiment $\mathbf{Q}$ given a set of observed experiments $\mathcal{D}_c$.

This cost normalization approach is a natural baseline and has been suggested in the context of other optimization problems, for example, Krause et al., (2008). However, in most such prior work, the actual empirical evaluations involved uniform cost models and thus there is little empirical data regarding the performance of normalization. In our setting of constrained experiments, a uniform cost model is not an option, since selecting among constrained experiments of varying variable cost is a fundamental aspect of the problem. Thus, our empirical evaluation, in Section 6, necessarily provides a substantial evaluation of this normalization principle.

Unfortunately, experimental results show that the proposed cost normalized approach can be outperformed by random policy in some cases. This prompts us to introduce a *constrained minimum cost*(CMC) approach which will only select a constrained experiment if it is expected to perform better than a random policy when spending the same amount of budget.

#### 5.3.2 Constrained Minimum Cost (CMC) Policies

For any heuristic on constrained experiments $H(\mathbf{Q}|\mathcal{D}_c)$, which assigns a score to constrained experiment $\mathbf{Q}$ given a set of observed experiments $\mathcal{D}_c$, we can define an associated CMC policy. The principle behind CMC is to select the least cost constrained experiment that satisfies the following two conditions:

- **Condition 1:** It *approximately optimizes* the heuristic value,
- **Condition 2:** It has an expected improvement (EI) that is no worse than the random policy after spending the same amount of budget.

The first condition encourages the selection of constrained experiments that look promising according to $H$, but it might result in the selection of an overly costly experiment. The second condition helps to place a limit on how much we are willing to pay to achieve a good heuristic value. Specifically, we will only be willing to pay a cost of $c$ for a single constrained experiment $\mathbf{Q}$ if its expected improvement is no worse than that achieved by a set of random experiments whose total cost is $c$.

To formalize this policy, we first make the notion of *approximately optimize* more precise by introducing a parameterized version of the CMC policy and then show how the parameter will be automatically selected via condition 2. For a given heuristic $H$, let $h^*$ be the value of the highest scoring constrained experiment that fits within the current budget. Note that this will necessarily be one of the most constrained (i.e. most expensive) experiments within the budget. For a given parameter $\alpha \in [0, 1]$, the $\mathrm{CMC}_{H,\alpha}$ policy selects the least-cost constrained experiment that achieves a heuristic value of at least $\alpha \cdot h^*$. Formally, this is defined as

$$\mathrm{CMC}_{H,\alpha}(\mathcal{D}_c, B_c) = \underset{\mathbf{Q}: f_c(Q) \leq B_c}{\operatorname{argmin}} \{ f_c(\mathbf{Q}) \mid H(\mathbf{Q}|\mathcal{D}_c) \geq \alpha h^* \} \tag{11}$$

The value of $\alpha$ controls the trade off between the cost of $\mathbf{Q}$ and its heuristic value. Smaller/larger values of $\alpha$ will result in less/more costly experiments, but smaller/larger heuristic values. In our preliminary work, we experimented with the $\text{CMC}_{H,\alpha}$ policy and found that it was difficult to select a value of $\alpha$ that worked well across a wide range of optimization problems, cost structures, and budgets. This motivated the introduction of condition 2 above to help us adaptively select an appropriate value of $\alpha$ at each decision point.

We now formalize the CMC class of policies. The objective is to select the largest value of $\alpha$ such that the experiment suggested by $\text{CMC}_{H,\alpha}$ satisfies condition 2. This will guarantee that the selected constrained experiment will: 1) achieve a heuristic value as close as possible to $h^*$, and 2) outperforms the random policy given the same cost allocation. In the following, we define $\text{EIR}(\mathcal{D}_c, C)$ to be the expected improvement of a set of random experiments that have a total cost of $C$ and $\mathbf{Q}_\alpha$ to be the constrained experiment returned by $\text{CMC}_{H,\alpha}$. Also, let $\text{EI}(\mathbf{Q}_\alpha)$ be the expected improvement of constraint experiment $\mathbf{Q}_\alpha$ and $c_\alpha$ be its cost. Our parameter-free CMC policy is now defined as:

$$\begin{gathered}\text{CMC}_H(\mathcal{D}_c, B_c) = \text{CMC}_{H,\alpha^*}(\mathcal{D}_c, B_c) \\ \alpha^* = \arg\max\{\alpha \in [0,1] \mid \text{EI}(\mathbf{Q}_\alpha|\mathcal{D}_c) \geq \text{EIR}(\mathcal{D}_c, \lceil c_\alpha \rceil)\}.\end{gathered} \tag{12}$$

In practice we compute $\text{EIR}(\mathcal{D}_c, C)$ and $\text{EI}(\mathbf{Q}_\alpha)$ via Monte Carlo simulation. This is a straightforward process in both cases. For EIR to compute one EIR sample, we randomly select a set of experiments within the budget $C$ and then sample outcomes for those experiments via the Gaussian Process conditioned on $\mathcal{D}_c$. The improvement of the best outcome is taken to be the result of the EIR sample. The estimate of EIR is the average of $L$ EIR samples. A similar process is used for EI, except that rather than drawing random experiments for each sample we use the experiments in $\mathbf{Q}_\alpha$.

The following steps summarize the overall computational process of CMC-MEI.

1. Compute $h^*$ by maximizing $H$ over constrained experiments that fall within the current budget. Note that this only requires optimizing over the set of minimum-sized constrained experiments that have a cost less than the budget.

2. Perform a discretized line search to find $\alpha^*$ according to Equation 12.

3. Return $\text{CMC}_{H,\alpha^*}(\mathcal{D}_c, B_c)$ according to Equation 11.

## 6. Experimental Results

Our goal is to evaluate the performance of the proposed policies in scenarios that resemble typical real-world scientific applications. In particular, the experimental domains that motivate our work in this paper focus on low-dimensional optimization problems. This choice is based on two reasons. First, with typical budgets the number of total experiments is often limited, which makes optimizing over many dimensions impractical. In practice, the scientists often have to carefully select a few key dimensions to consider. Second, in real-world applications, such as our motivating problem, it is prohibitively difficult to satisfy constraints on more than a couple of experimental variables. Thus, the most relevant

scenarios for us and many other problems are moderate numbers of experiments and small dimensionality.

### 6.1 Experimental Setup

Below we describe the set up of our experiments.

**Test Functions.** We evaluate our policies using five 2-dimensional functions in $[0, 1]^2$. The first three functions: *Cosines*, *Rosenbrock*, and *Discontinuous* are benchmarks that have been widely used in previous studies on stochastic optimization (Anderson, Moore, & Cohn, 2000; Brunato, Battiti, & Pasupuleti, 2006; Azimi et al., 2010). The mathematical expressions of the functions are listed in Table 1 and their contour plots are given in Figure 1.

The two remaining functions are derived from real-world experimental data sets involving *hydrogen production* and our motivating *fuel cell* application. For the former we utilize data collected as part of a study on biosolar hydrogen production (Burrows, Wong, Fern, Chaplen, & Ely, 2009), where the goal was to maximize hydrogen production of the cyanobacteria Synechocystis sp. PCC 6803 by optimizing the pH and Nitrogen levels of the growth media. The data set contains 66 samples uniformly distributed over the 2-d input space. This data is used to simulate the true function by fitting a Gaussian Process (GP) model with RBF kernel, picking kernel parameters via standard validation techniques such as cross validation. With this model we then simulated the experimental design process by sampling from the posterior of this GP to obtain noisy outcomes for requested experiments. For this purpose, we used a zero-mean Gaussian noise model with variance equal to 0.01. See Figure 1 for the contour plot.

For our motivating application (described in the introduction), we utilize data from a set of initial microbial fuel cell experiments using anodes with different nano-enhancements. In particular, each anode was coated with gold nano-particles under different fabrication conditions leading to varying particle densities, shapes, and sizes. The construction of each anode required approximately two days.[1] Each anode was then installed in a microbial fuel cell and run using pure *Shewanella oneidensis* bacterial cultures grown in fed-batch mode for one week while recording the current density at regular intervals. The temporally averaged current density was taken to be the dependent variable to be optimized by modifying the nano-structure. To characterize the nano-structure on each anode, we captured images using scanning electron microscopy and used standard image processing software to compute two features: average area of individual particles, and average circularity of individual particles. Those features were selected to be the independent variables of the design since they can be roughly controlled during the fabrication process and appear to influence the current density. Unfortunately, due to the high cost of running these experiments, which is precisely the motivation for this paper, our data set currently consists of just 16 data points, which are relatively uniformly distributed over the experimental space. Due to the sparse data, we utilize polynomial Bayesian regression with degree 4, rather than Gaussian processes with RBF kernels, to simulate the true function. See Figure 1 for the contour plot.

1. For this first round of experiments no constraints were provided to the scientist constructing the anodes. Rather the goal was to generate a diverse set of anodes to provide a good set of data for seeding the experimental design process. The construction time would likely be more than two days in the presence of constraints since a number of growth conditions and trials would be necessary.

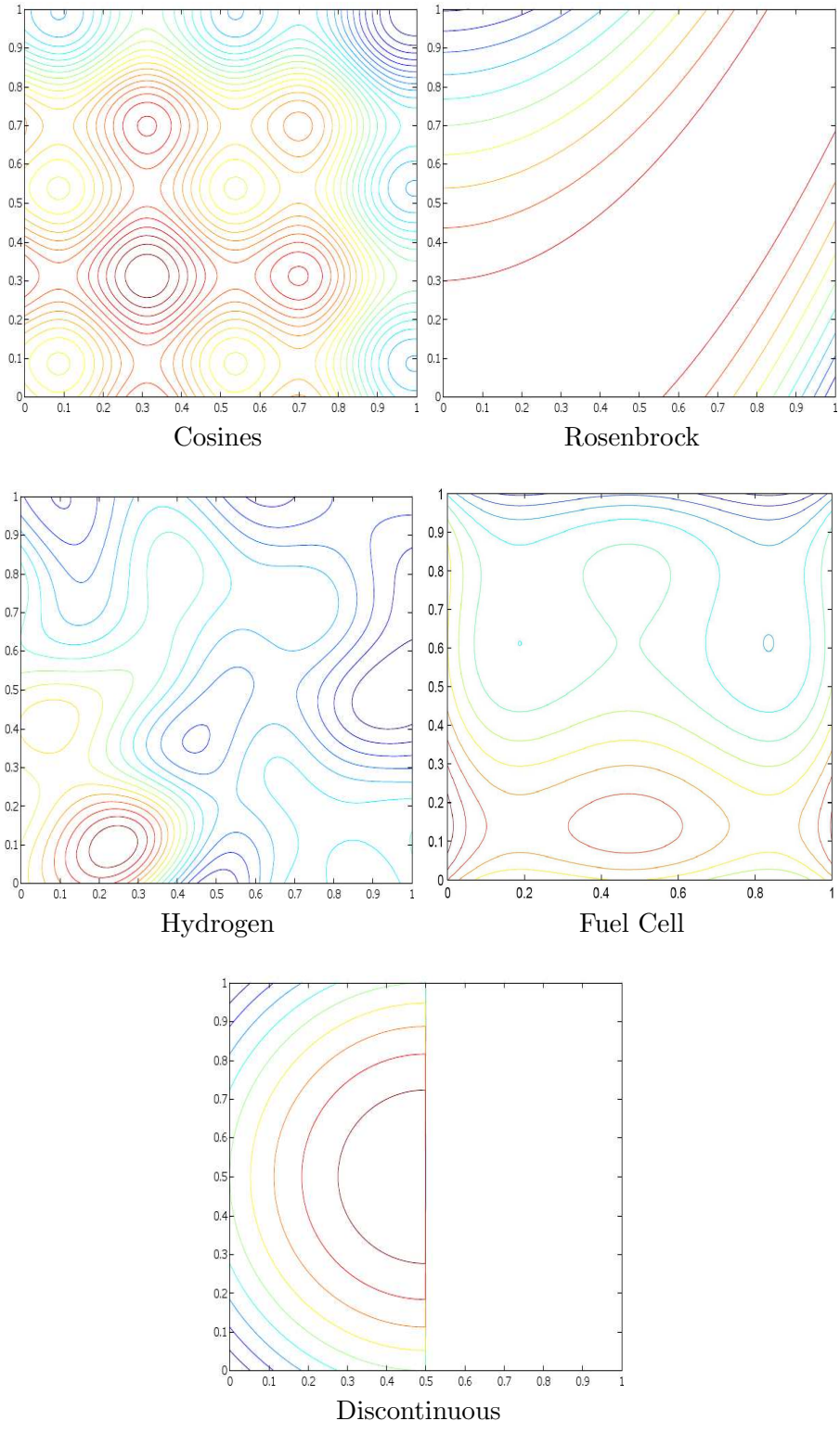


Figure 1: Contour plots of the test functions.

Table 1: Benchmark Functions.

| **Function** | **Mathematical representation** |
|---|---|
| Cosines | $1-(u^2+v^2-0.3cos(3\pi u)-0.3\cos(3\pi v)), \qquad u=1.6x-0.5, v=1.6y-0.5$ |
| Rosenbrock | $10-100(y-x^2)^2-(1-x)^2$ |
| Discontinuous | $1-2((x-0.5)^2+(y-0.5)^2)$ if $x<0.5, \qquad 0$ otherwise |

**Model Definitions.** In this work, we assume that the density $p_{\mathbf{x}}(\mathbf{x}|\mathbf{Q})$ over experiments given a query $\mathbf{Q}$ is uniform over the ranges specified by $\mathbf{Q}$.

To compute the conditional posterior $p(y|\mathbf{x},\mathcal{D})$ required by our non-sequential approach and model-based sequential heuristics, we use Gaussian process (Rasmussen & Williams, 2006) with zero mean prior and covariance specified by RBF kernel function:

$$cov(\mathbf{x}_i,\mathbf{x}_j)=k(\mathbf{x}_i,\mathbf{x}_j)=\sigma_f \exp(-\frac{1}{2\kappa}|\mathbf{x}_i-\mathbf{x}_j|^2), \tag{13}$$

where $\kappa$ is the length scale parameter that can be considered as the distance we have to move in the input space before the function value changes significantly, and $\sigma_f$ is the signal variance which specifies the maximum possible variance at each point. In this paper we set $\kappa=0.02$ and signal variance $\sigma_f=y_{\max}^2$, where $y_{\max}$ is an upper bound on the output values (this is typically easy to elicit from scientists and serves to set our prior uncertainty so that there is non-trivial probability assigned to the expected range of the output). We did not optimize these parameters, but did empirically verify that the GP behaved reasonably for our test functions.

**Cost Function.** In our motivating application, the cost of setting up and running a fuel cell experiment given a constrained experiment request can be roughly considered to have two components. The first component corresponds to the cost of setting up an experiment (producing a nano-structure) that satisfies the given constraints, which is variable depending on the size of the constraints. The tighter the constraints, the more costly they will be. The second component corresponds to the cost of running the constrained experiment, which is generally constant. This two-component cost structure is very common in real-world applications where a portion of the experimental process can be controlled more precisely and has uniform costs across different queries, while other portions are less controllable and have a cost that is inversely proportional to the size of the constraints we place on them. To capture this structure, we define the following cost function $f_c(\cdot):\mathbf{Q}\rightarrow\Re^+$:

$$f_c(\mathbf{Q})=1+\prod_{i=1}^{d}\frac{slope}{(b_i-a_i)}. \tag{14}$$

In this formulation, the constant of one captures the stationary part of the cost, and the second term captures the variable portion that is inversely proportional to the size of the constrains of query $\mathbf{Q}$. The value of the *slope* parameter dictates how quickly the cost increases as the size of a constrained experiment decreases. We evaluate our proposed

approaches considering three different slope values; $slope = 0.1, 0.15, 0.30$. Note that all of our proposed approaches can be readily applied to other cost functions.

**Discretizing the Input Space.** As mentioned previously, our policies assume that the input space is discretized. In particular, we divide each input dimension into 100 equal-length subintervals. Note that this implementation is most appropriate for low dimensional optimization problems, which as described previously is the situation we often encounter in real-world applications.

**Evaluation Settings.** In our evaluation, we test all of the proposed policies in comparison to a random policy (i.e., a policy that always selects the entire input space as the constrained experiment).

Given a budget $B$ and a function $f(\cdot)$ to be optimized, each run of a policy results in a set of observed experiments $\mathcal{D}_c$. Let $\mathbf{x}^*$ be the experiment in $\mathcal{D}_c$ that is predicted to have the maximum expected outcome $y^*$. The *regret* of the policy for a particular run is then defined to be $y_{max} - y^*$, where $y_{max}$ is the maximum value of $f(\cdot)$. For each test function and choice of budget and cost structure (i.e., choice of slope), we evaluate each policy by averaging the regret over 200 runs. Each run starts with $n = 5$ randomly selected initial points $\mathcal{D} = \{(\mathbf{x}_0, y_0), \cdot, \cdot, \cdot(\mathbf{x}_5, y_5)\}$, and then the policies are used to select constrained experiments until the budget runs out, at which point the regret is measured. In order to ease the comparison of the regret values across different functions, we report normalized regret values, which are computed by dividing the regret of each policy by the mean regret achieved by the random policy. A normalized regret less than one indicates that an approach outperforms random, while a value greater than one indicates that an approach is worse than random. In the first round of experiments, we fixed the total budget to $B = 15$ and examine the effect of the cost-model slope parameter over values 0.1, 0.15 and 0.3. In later experiments, we will consider larger budgets.

Note that our non-sequential policy can be used to consume all the experimental budget at once. However, in practice there is typically a limit on the number of constrained experiments that can be processed simultaneously due to limited resources. As such, in the non-sequential setting our policy is used to select up to five simultaneous queries subject to the budget constraint. We will repeat this process until the budget is consumed.

The run time for selecting a single experiment in the sequential setting is on the order of minutes (generally under five) in our experiments with an un-optimized matlab implementation. The run time for selecting a batch of five or fewer queries was never more than 30 minutes.

### 6.2 Results and Discussions

Our results for individual functions are shown in Table 2, and their corresponding standard deviations are shown inside the parentheses. The first row of each table presents the results of our non-sequential greedy algorithm (NS-Greedy). Rows 2 to 6 show the performance of the model-based sequential policies for both CMC and CN cost policies. Note that, for the CN policy, we report the results of CN-MEI, as it performed the best among all CN policies. In addition, it has a nice interpretation as maximizing the rate of expected improvement per unit cost. Finally, the last row shows the performance of our model-free sequential policies.

Table 2: Normalized regrets on individual functions for varying cost models (i.e., slopes)

| | **slope = 0.1** | **slope = 0.15** | **slope = 0.30** |
|---|---|---|---|
| **Cosines, Normalized Regret** (95% **Confidence Interval)** | | | |
| NS-Greedy | 0.767 (±0.04) | 0.838 (±0.05) | 0.841 (±0.05) |
| CN-MEI | 0.569 (±0.05) | 0.714 (±0.06) | 0.826 (±0.06) |
| CMC-MEI | **0.417** (±0.04) | **0.514** (±0.06) | 0.794 (±0.06) |
| CMC-MPI(0.2) | 0.535 (±0.05) | 0.584 (±0.06) | **0.616** (±0.06) |
| CMC-MUI | 0.797 (±0.06) | 0.804 (±0.06) | 0.817 (±0.06) |
| CMC-MM | 0.708 (±0.07) | 0.767 (±0.07) | 0.736 (±0.06) |
| RR/BRR | 0.842(±0.06) / 0.833(±0.06) | 0.866(±0.06) / 0.862(±0.06) | 0.897(±0.06) / 0.889(±0.06) |
| **Discontinuous, Normalized Regret** (95% **Confidence Interval)** | | | |
| NS-Greedy | 0.528 (±0.06) | 0.690 (±0.06) | 0.748 (±0.05) |
| CN-MEI | **0.527** (±0.06) | **0.497** (±0.06) | **0.626** (±0.08) |
| CMC-MEI | 0.564 (±0.06) | 0.677 (±0.08) | 0.779 (±0.09) |
| CMC-MPI(0.2) | 0.954 (±0.11) | 0.940 (±0.10) | 0.951 (±0.11) |
| CMC-MUI | 0.710 (±0.10) | 0.709 (±0.11) | 0.693 (±0.09) |
| CMC-MM | 1.289 (±0.15) | 1.225 (±0.16) | 1.116 (±0.16) |
| RR/BRR | 0.602(±0.07) / 0.587(±0.07) | 0.617(±0.07) / 0.602(±0.07) | 0.634(±0.08) / 0.634(±0.08) |
| **Rosenbrock, Normalized Regret** (95% **Confidence Interval)** | | | |
| NS-Greedy | 0.650 (±0.05) | 0.877 (±0.06) | 0.930 (±0.06) |
| CN-MEI | 0.602 (±0.06) | 0.665 (±0.07) | 0.736 (±0.08) |
| CMC-MEI | 0.547 (0.35) | 0.**556 (0.39)** | 0.630 (0.47) |
| CMC-MPI(0.2) | **0.503** (±0.05) | 0.594 (±0.06) | **0.608** (±0.07) |
| CMC-MUI | 0.805 (±0.11) | 0.974 (±0.16) | 0.913 (±0.14) |
| CMC-MM | 0.721 (±0.09) | 0.740 (±0.10) | 0.662 (±0.08) |
| RR/BRR | 0.897(±0.12) / 0.881(±0.12) | 0.930(±0.12) / 0.927(±0.12) | 0.967(±0.14) / 0.955(±0.14) |
| **Hydrogen, Normalized Regret** (95% **Confidence Interval)** | | | |
| NS-Greedy | 0.879 (±0.06) | 0.969 (±0.08) | 0.993 (±0.09) |
| CN-MEI | 0.176 (±0.04) | 0.354 (±0.06) | 0.852 (±0.09) |
| CMC-MEI | **0.129** (±0.04) | **0.233** (±0.06) | **0.420** (±0.07) |
| CMC-MPI(0.2) | 0.408 (±0.09) | 0.449 (±0.10) | 0.613 (±0.10) |
| CMC-MUI | 0.716 (±0.08) | 0.695 (±0.08) | 0.868 (±0.09) |
| CMC-MM | 0.728 (±0.11) | 0.605 (±0.10) | 0.691 (±0.11) |
| RR/BRR | 1.107(±0.09) / 1.065(±0.09) | 1.161(±0.10) / 1123(0.10) | 1.173(±0.09) / 1.148(±0.09) |
| **Fuel Cell, Normalized Regret** (95% **Confidence Interval)** | | | |
| NS-Greedy | 0.980 (±0.02) | 0.985 (±0.02) | 0.995 (±0.03) |
| CN-MEI | **0.929** (±0.02) | 0.950 (±0.02) | 0.986 (±0.03) |
| CMC-MEI | 0.931 (±0.02) | **0.908** (±0.02) | **0.940** (±0.02) |
| CMC-MPI(0.2) | 0.932 (±0.02) | 0.930 (±0.03) | 0.943 (±0.03) |
| CMC-MUI | 0.971 (±0.03) | 0.973 (±0.03) | 0.995 (±0.03) |
| CMC-MM | 0.945 (±0.03) | 0.963 (±0.04) | 0.963 (±0.05) |
| RR/BRR | 1.038(±0.03) / 1.029(±0.03) | 1.049(±0.03) / 1.044(±0.03) | 1.046(±0.03) / 1.041(±0.03) |

Table 3: Normalized Overall Regrets.

| | **slope = 0.1** | **slope = 0.15** | **slope = 0.30** |
|---|---|---|---|
| NS-Greedy | 0.760 | 0.871 | 0.901 |
| CN-MEI | 0.560 | 0.636 | 0.805 |
| CMC-MEI | **0.517** | **0.578** | **0.712** |
| CMC-MPI(0.2) | 0.666 | 0.698 | 0.746 |
| CMC-MUI | 0.800 | 0.831 | 0.857 |
| CMC-MM | 0.874 | 0.889 | 0.834 |
| RR | 0.897 | 0.925 | 0.944 |
| BR | 0.879 | 0.911 | 0.934 |

In order to provide an assessment of the overall performance of different methods, Table 3 presents the normalized regrets for each policy averaged across our five functions. The different columns of the table correspond to different slope values for the cost function. Below we discuss the results of different methods in detail.

#### 6.2.1 Non-Sequential

We will first examine the performance of our non-sequential greedy policy (NS-Greedy). Recall that we present the normalized regret in our results, thus smaller value indicates better performance. Further, a policy outperforms random whenever its normalized regret is less than 1.

From Table 2, we observe that the proposed greedy algorithm (NS-Greedy) performs consistently better than the random policy for all functions. Among these functions, it can be seen that the performance advantage of NS-Greedy is more significant when the slope parameter of the cost function is smaller. This is consistent with our expectation: with a smaller slope, the cost of our query grows slower as we tighten the constraints. This will allow our algorithm to more aggressively select tighter constraints based on the posterior model of the function. In fact, if the slope is large enough, one would expect the optimal policy to be completely random.

Comparing with sequential approaches, we first observe that NS-Greedy compared favorably to the two model-free methods. This is not surprising because RR/BRR do not consider the posterior model of the function in selecting queries. On the other hand, we also observe that the NS-Greedy algorithm performs significantly worse than the best model-based sequential policies, such as CMC-MEI. This result is expected because sequential policies allow us to update and improve the model of the function with each query. Therefore, we generally expect sequential policies to perform better than non-sequential methods which is a common phenomenon in the active learning literature (Azimi, Fern, Fern, Borradaile, & Heeringa, 2012).

#### 6.2.2 Sequential

In this section we examine the performance of the sequential policies, including both model-free and model-based methods.

**Model-Free Policies.** From Table 3 we see that RR and BRR achieve an improvement over random by approximately 10% across the different slopes. This shows that the heuristic of trying to evenly cover the space pays off compared to random. BRR is also observed to perform slightly better than RR, which indicates that the additional exploitive behavior of BRR pays off overall. Looking at the individual results in Table 2, we see that for the Hydrogen and Fuel Cell functions, both BRR and RR perform worse than random. Further investigation reveals that the reason for this poor performance is that RR/BRR have a bias toward experiments near the center of the input space. This bias is a result of the fact that constrained experiments (hyper-rectangles) are required to fall completely within the experimental space and there are fewer such hyper-rectangles that contain points near the edges and corners. The Hydrogen and Fuel Cell functions have their optimal points near corners of the space, explaining the poor performance.

**Model-Based Policies.** We now focus on the performance of the proposed model-based sequential policies. From the averaged overall results (Table 3), our first observation is that the model-based policies in general perform better than the random policy. Specifically, looking at the results of individual functions, we see that all model-based policies outperform random, with the exception of CMC-MM on the Discontinuous function. This shows that the two proposed approaches for considering cost are able to avoid catastrophic choices that expend the budget more quickly than is warranted.

Our analysis of the poor performance of CMC-MM on the Discontinuous function revealed that CMC-MM would often get stuck in poor local optima and cease to explore the space adequately. Although at each step the CMC-MM policy determined that its selection was better than random in the near term, this did not translate to long term improvement over random due to the lack of exploration. The Discontinuous function is particularly prone to elicit this behavior due to the fact that it has a large sub-optimal and nearly uniform region, which is difficult for CMC-MM to escape from. This overly greedy performance is consistent with prior observations of the MM heuristic and is largely addressed by the other heuristics that provide some measure of exploratory value. In fact, CMC-MM is highly dependent on its initial given random points. For example, if the initial given points $\mathcal{D}$ have been chosen from a non-optimal region, which is more than 50% of the input space for the Discontinuous function, the CMC-MM approach cannot give a satisfactory performance. This can be seen by the standard deviation of CMC-MM, which is higher than other model-based and model-free methods. It shows that the performance of CMC-MM changes significantly in each iteration which is because of its initial points.

In addition, from Table 3, it can be seen that all of the model-based approaches outperform the model free approaches. This indicates that the heuristics we are considering and our GP probabilistic model are providing useful information for effectively guiding the constrained experiment selection.

Comparing different model-based heuristics, we see that the MEI-based methods (CN-MEI and CMC-MEI) are the top contenders among all methods. Examining the results for individual functions, we can see that this holds for all functions except for Rosenbrock, where the CMC-MPI is slightly better than MEI-based methods. Upon closer examination of the behavior of the MPI and MEI heuristics, we found that MPI often selects slightly fewer experiments than MEI, which we believe is due to fact that the MEI heuristic tends

to be smoother than MPI over the experimental space. The smoothness of MEI allows the CMC policy to select less constrained queries and but still achieve "approximately" optimal heuristic value, leading to more constrained experiments. In general we recommend CMC-MEI as the most preferable heuristic to use based on its consistently superior performance and the fact that it is parameter free.

We are also interested in comparing the performance of the two proposed schemes for handling the cost, namely CN and CMC. Focusing on CMC-MEI and CN-MEI, we can see that CMC-MEI generally outperforms CN-MEI. While the differences in the behavior of these two policies appear subtle, experimental investigation show that CN-MEI tends to be overly conservative toward selecting costly experiments in comparison with CMC-MEI, especially at the later stages of the experimental process.

### 6.3 Varying Budget

In this round of experiments, we fixed the cost model slope to 0.1 and varied the budget from 10 to 60 units in increments of 10. We are interested in examining the relative performance of different model-based policies (including both sequential and non-sequential) compared to the random policy as we increase the budget.

Figure 2 plots the *absolute* regret (rather than the normalized regret) versus budget for the best sequential policies including CMC-MEI, CN-MEI and CMC-MPI, and the proposed non-sequential policy (NS-Greedy). We have also plotted the performance of random policy as a reference baseline. We use the same experimental setting as used previously. Specifically, for sequential methods, in each iteration we select one query until the budget is completely consumed. For the proposed non-sequential approach, we select up to five queries in each iteration until the budget is consumed.

First, we observe that the performance of NS-Greedy continues to dominate Random as we increase the budget. This suggests that the performance advantage of NS-Greedy over Random is robust to the amount of experimental budget. We also observe that NS-Greedy is generally outperformed by the lead sequential policies, such as CMC-MEI, and CMC-MPI. This is consistent with our previous observations with fixed budget and varying slope. Finally, we see that polices based on the MEI and MPI heuristics generally achieve the best performance across a wide range of budgets. In particular, they consistently maintain a significant advantage over Random. The MEI-based and CMC-MPI policies are roughly comparable for all functions except for the Fuel Cell function. In that case CMC-MPI slightly outperforms CMC-MEI for large budgets.

Overall, given the results from the previous experiments, CMC-MEI can still be considered as a recommended method, due to its combination of good performance, smoothness and robustness. CMC-MEI is also preferable in that it does not require the selection of a margin parameter.

### 6.4 Comparison with Precise Experiments

In this section, we compare our performance using constrained experiments to the performance achieved using precisely specified experiments. In particular, we compare CMC-MEI with its "precise" counterpart MEI. To do this, we use CMC-MEI to select up to fifteen constrained experiments (with infinite budget) for each function, and at each step evaluate

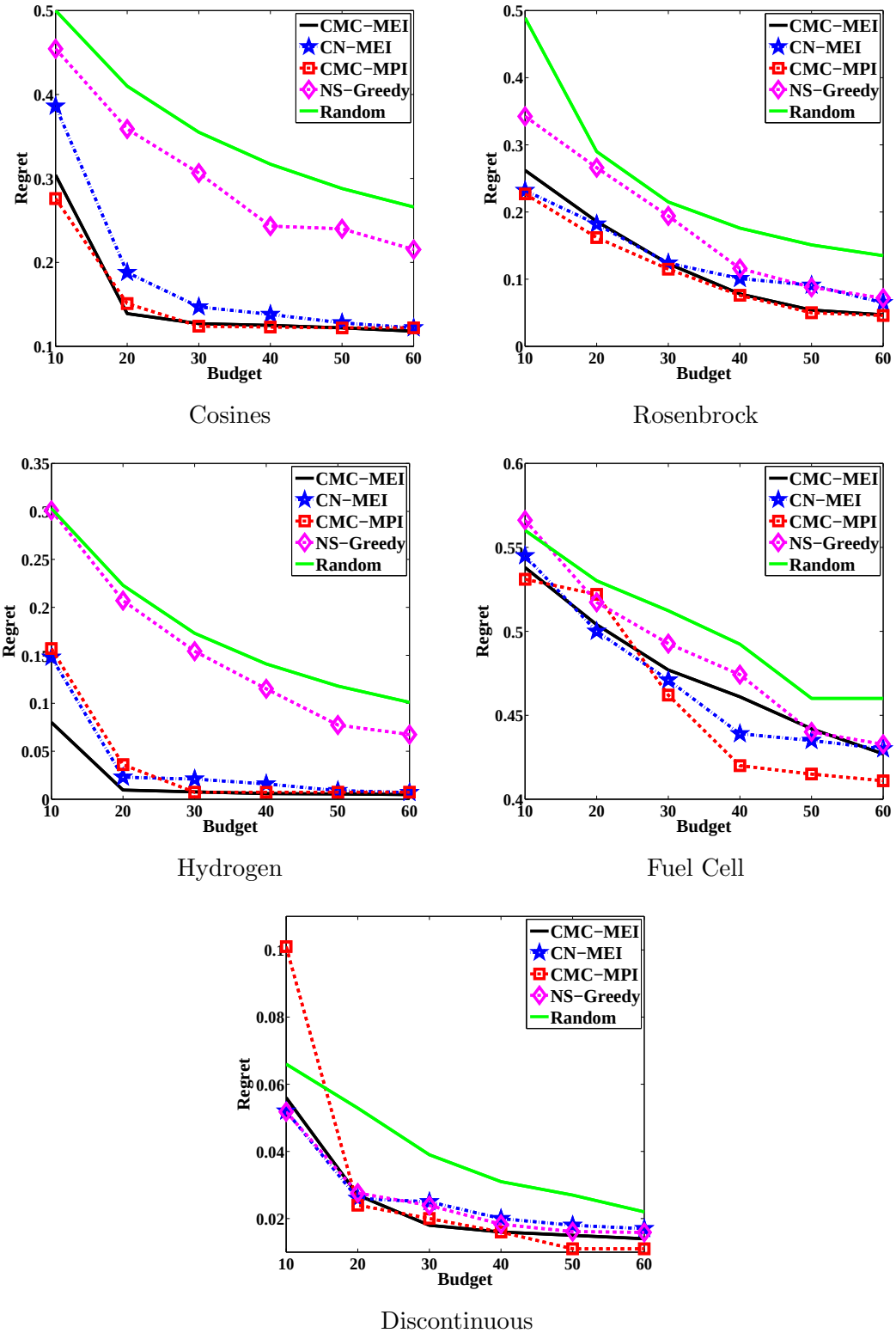


Figure 2: *Un-normalized* regret as a function of the budget (slope=0.1).

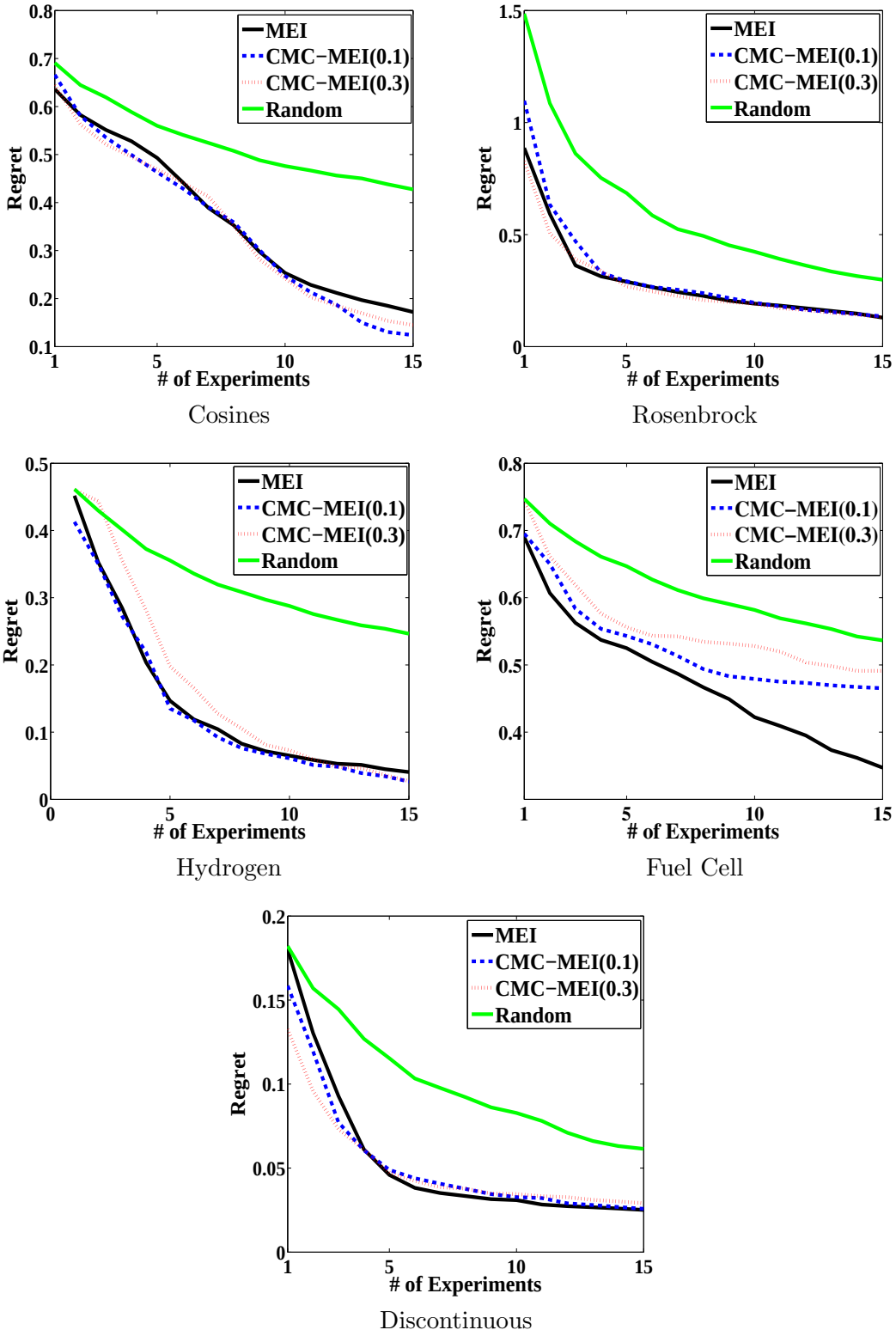


Figure 3: *Un-normalized* regret as a function of the number of experiments.

the regret. This is repeated for 100 times to generate an average performance curve for CMC-MEI as a function of the number of constrained experiments. This is done for two different cost models with slope set to 0.1 and 0.3 respectively, resulting in two curves for CMC-MEI. Similarly, we use MEI to select a sequence of fifteen precisely specified experiments and generate the same average performance curve (over 100 random runs). Finally, as a reference point, we also plot the performance when experiments are selected randomly. Figure 3 shows the performance curves of MEI, CMC-MEI (with $slope = 0.1$ and 0.3 respectively) and random.

From the figure we can see that in most cases CMC-MEI performed comparably to MEI. For these cases, we observe no detrimental effective from the use of constrained experiments. If we compare the efficiency of CMC-MEI with slopes 0.3 and 0.1 (larger slopes yield higher experimental costs), we see that in most cases they are comparable. However, for the Fuel Cell and Hydrogen functions, the smaller slope is consistently better (by a small margin). Further, these are also the two functions where precise experiments show the most significant advantage over CMC-MEI (in particular for slope =0.3). A likely explanation for this is that the optimal regions for these two functions are fairly small, highly peaked and near the boundaries. This can make it difficult to effectively explore this region using constrained experiments, especially with larger slopes.

### 6.5 Comparison with Constant Window Experiments

In our final experiments, we compare the performance of the CMC-MEI approach with a Constant Window (CW) approach, where constant constraint sizes are used throughout the optimization process. The goal is to understand the importance of dynamically selecting the window size as done by CMC-MEI. We consider three different window sizes, denoted by CW5, CW20, CW50, which correspond to constraint sizes of 5%, 20% and 50% in each dimension respectively. Thus, the cost of CW5 is significantly more than CW50 while it has more precision and control over the final selected samples. We compare these CW approaches against CMC-MEI. The cost model parameter is set to $slope = 0.1$ and the budget is varied from 10 to 60 in denomination of 10. The results are provided in Figure 4.

First, we observe that the best performing CW approach varies significantly across benchmarks and budgets. This indicates that choosing the window size for a particular application is non-trivial. Second, we see that CMC-MEI, which adaptively selects the window size, performs the best or is competitive with the best CW approach. This is another indication that CMC-MEI is an effective strategy for choosing window sizes. Further analysis of these experiments indicates that CMC-MEI tends to select experiments close to CW50 at the beginning and then decreases the window size after several experiments.

## 7. Summary and Future Directions

Motivated by a real-world application, this paper introduced a novel framework for budgeted Bayesian optimization with constrained experiments. In this framework, instead of asking for samples of the unknown function at precisely specified inputs, we ask for a constrained experiment and the cost of the constrained experiments is variable depending on the tightness of the constraints. We studied this problem in two different settings.

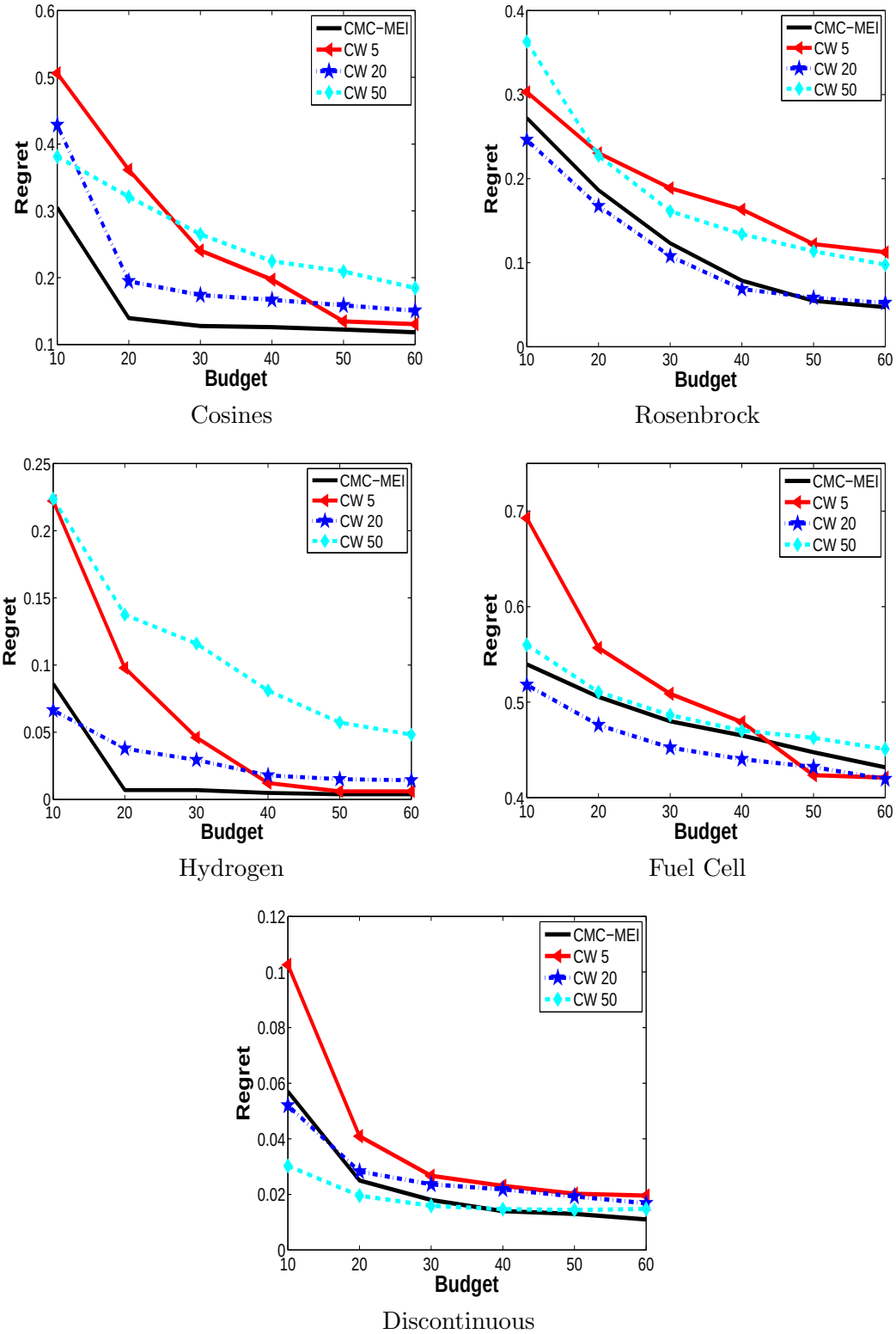


Figure 4: CMC-MEI performance versus constant window size experiments.

In the non-sequential setting, multiple constrained experiments are selected at once. For this setting, we introduced a non-decreasing submodular objective function and presented a greedy algorithm for approximately optimizing the proposed objective. Empirical evaluation indicates that the proposed non-sequential algorithm consistently outperforms a baseline random policy across different budget and cost configurations.

In the sequential setting of the problem, one constrained experiment is selected in each iteration. We extended a number of classic Bayesian optimization and experiment design heuristics for constrained experiments. Direct use of such heuristics to select constrained experiments will select overly tight constraints and consume all of the budget at once. Thus, we introduced two general cost policies, namely CN and CMC, to achieve a balance between moderating the cost of the experiments and optimizing the heuristics. The experiments show that sequential policies generally outperform the non-sequential policy, and the proposed CN and CMC cost policies are effective at dispensing the budget rationally. Overall we found that CMC used with the MEI heuristic (CMC-MEI) demonstrated robust performance and is parameter-free, making it a recommended method.

The work described here focused on methods for optimizing low-dimensional functions, which is typical of the types of scientific and engineering applications that motivated this work. Extending our methods to higher dimensions requires optimizing the selection criteria over continuous rather than discrete input spaces. There are a number of straightforward approaches to doing this and future work could include evaluating those approaches and designing more sophisticated ones. The most interesting direction for future work is to continue enriching the cost and action models supported by Bayesian Optimization methods to more closely match the needs of real-world applications. Solutions for these extended models will require a tighter integration of planning and scheduling techniques with the ideas developed so far for traditional Bayesian Optimization.

## 8. Acknowledgments

This research was supported by NSF grant IIS 1320943.

## Proof of Lemma 1

**Lemma 1**. Let $\mathcal{X}_M = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ denote the random variables representing the outcome of the random draw for $\mathcal{Q}_M = \{\mathbf{Q}_1, ..., \mathbf{Q}_M\}$ respectively where $\mathcal{Q}_M$ is the set of all possible queries. For any given $\mathcal{X}_M$, $J_{\mathcal{X}_M}(S)$, which returns the expected maximum over a set of jointly distributed random variables, is a monotonically non-decreasing submodular set function.

*Proof.* Suppose $\mathcal{S}$ is a finite set. Then $g : 2^{\mathcal{S}} \rightarrow \mathbb{R}^+$ is a submodular set function if for all $S_1 \subseteq S_2 \subset \mathcal{S}$ and $\mathbf{x} \in \mathcal{S} \setminus S_2$, it holds that $g(S_1 \cup \{\mathbf{x}\}) - g(S_1) \geq g(S_2 \cup \{\mathbf{x}\}) - g(S_2)$. In addition a set function $g(\cdot)$ is called monotonically non-decreasing if $g(S_1) \leq g(S_2)$.

We first prove that $\mathbb{E}[\max(\cdot)]$ is monotonic function and then we show that it is a submodular objective function.

Assume that $S_1 = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_p\}$ with $p \leq k$. We need to prove that

$$\mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) \middle| \mathcal{D}\right] \geq \mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p\right) \middle| \mathcal{D}\right]. \tag{15}$$

We use the definition of the expectation to prove the result.

$$\begin{aligned}
&\mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) \middle| \mathcal{D}\right] \\
&\quad = \int \cdots \int \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) p_{y_1, y_2, \cdots, y_p, \cdots, y_k | \mathcal{D}} dy_1 dy_2 \cdots dy_p \cdots dy_k \\
&\quad \geq \int \cdots \int \max\left(y_1, y_2, \cdots, y_p\right) p_{y_1, y_2, \cdots, y_p, \cdots, y_k | \mathcal{D}} dy_1 dy_2 \cdots dy_p \cdots dy_k \\
&\quad = \int \cdots \int \max\left(y_1, y_2, \cdots, y_p\right) \left( \int \cdots \int p_{y_1, y_2, \cdots, y_p, \cdots, y_k | \mathcal{D}} dy_{p+1} \cdots dy_k \right) dy_1 dy_2 \cdots dy_p \\
&\quad = \int \cdots \int \max\left(y_1, y_2, \cdots, y_p\right) p_{y_1, y_2, \cdots, y_p | \mathcal{D}} dy_1 dy_2 \cdots dy_p \\
&\quad = \mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p\right) \middle| \mathcal{D}\right].
\end{aligned} \tag{16}$$

This shows that $\mathbb{E}[\max(\cdot)]$ is a nondecreasing monotonic function.

To prove the submodularity property, We need to show that

$$\begin{aligned}
&\mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p, y^*\right) \middle| \mathcal{D}\right] - \mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p\right) \middle| \mathcal{D}\right] \\
&\quad \geq \mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*\right) \middle| \mathcal{D}\right] - \mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) \middle| \mathcal{D}\right].
\end{aligned} \tag{17}$$

To prove this, we start from the right hand side of the inequality and the basic definition of the expectation.

$$
\begin{aligned}
&\mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*\right) \middle| \mathcal{D}\right] - \mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) \middle| \mathcal{D}\right] \\
&\quad = \int \cdots \int \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*\right) p_{y_1, y_2, \cdots, y_p, \cdots, y_k, y^* | \mathcal{D}} dy_1 dy_2 \cdots dy_p \cdots dy_k dy^* \\
&\qquad - \int \cdots \int \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) p_{y_1, y_2, \cdots, y_p, \cdots, y_k | \mathcal{D}} dy_1 dy_2 \cdots dy_p \cdots dy_k \\
&\quad = \int \cdots \int \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*\right) p_{y_1, y_2, \cdots, y_p, \cdots, y_k, y^* | \mathcal{D}} dy_1 dy_2 \cdots dy_p \cdots dy_k dy^* \\
&\qquad - \int \cdots \int \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) p_{y_1, y_2, \cdots, y_p, \cdots, y_k, y^* | \mathcal{D}} dy_1 dy_2 \cdots dy_p \cdots dy_k dy^* \\
&\quad = \int \cdots \int \left[\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*\right) - \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right)\right] \\
&\qquad\qquad p_{y_1, y_2, \cdots, y_p, \cdots, y_k, y^* | \mathcal{D}} dy_1 dy_2 \cdots dy_p \cdots dy_k dy^* \\
&\quad \leq \int \cdots \int \left[\max\left(y_1, y_2, \cdots, y_p, y^*\right) - \max\left(y_1, y_2, \cdots, y_p\right)\right] \\
&\qquad\qquad p_{y_1, y_2, \cdots, y_p, \cdots, y_k, y^* | \mathcal{D}} dy_1 dy_2 \cdots dy_p \cdots dy_k dy^* \\
&\quad = \int \cdots \int \left[\max\left(y_1, y_2, \cdots, y_p, y^*\right) - \max\left(y_1, y_2, \cdots, y_p\right)\right] p_{y_1, y_2, \cdots, y_p, y^* | \mathcal{D}} dy_1 dy_2 \cdots dy_p dy^* \\
&\quad = \int \cdots \int \max\left(y_1, y_2, \cdots, y_p, y^*\right) p_{y_1, y_2, \cdots, y_p, y^* | \mathcal{D}} dy_1 dy_2 \cdots dy_p dy^* \\
&\qquad\qquad - \int \cdots \int \max\left(y_1, y_2, \cdots, y_p\right) p_{y_1, y_2, \cdots, y_p | \mathcal{D}} dy_1 dy_2 \cdots dy_p \\
&\quad = \mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p, y^*\right) \middle| \mathcal{D}\right] - \mathbb{E}\left[\max\left(y_1, y_2, \cdots, y_p\right) \middle| \mathcal{D}\right]
\end{aligned}
\tag{18}
$$

Notice that the inequality holds if we can prove:

$$
\begin{aligned}
&\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*\right) - \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) \\
&\leq \max\left(y_1, y_2, \cdots, y_p, y^*\right) - \max\left(y_1, y_2, \cdots, y_p\right)
\end{aligned}
\tag{19}
$$

There are two possible cases as follows:

$$
\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*\right) = \begin{cases} y^* \\ \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right). \end{cases}
\tag{20}
$$

1. In the first case, if $\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*\right) = y^*$, then we also have $\max\left(y_1, y_2, \cdots, y_p, y^*\right) = y^*$. Hence,

$$
\begin{aligned}
&\max\left(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*\right) - \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) \\
&\qquad = y^* - \max\left(y_1, y_2, \cdots, y_p, \cdots, y_k\right) \\
&\qquad \leq y^* - \max\left(y_1, y_2, \cdots, y_p\right) \\
&\qquad = \max\left(y_1, y_2, \cdots, y_p, y^*\right) - \max\left(y_1, y_2, \cdots, y_p\right)
\end{aligned}
\tag{21}
$$

2. In the second case, if $\max(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*) = \max(y_1, y_2, \cdots, y_p, \cdots, y_k)$, then we have

$$\begin{aligned} &\max(y_1, y_2, \cdots, y_p, \cdots, y_k, y^*) - \max(y_1, y_2, \cdots, y_p, \cdots, y_k) \\ &\quad = 0 \\ &\quad \leq \max(y_1, y_2, \cdots, y_p, y^*) - \max(y_1, y_2, \cdots, y_p) \\ &\quad = \max(y_1, y_2, \cdots, y_p, y^*) - \max(y_1, y_2, \cdots, y_p) \end{aligned} \tag{22}$$

Notice that $\max(y_1, y_2, \cdots, y_p, y^*) - \max(y_1, y_2, \cdots, y_p)$ is always non-negative.

□

## References


Anderson, B., Moore, A., & Cohn, D. (2000). A nonparametric approach to noisy and costly optimization. In *ICML*.

Azimi, J., Fern, A., & Fern, X. (2010). Batch bayesian optimization via simulation matching. In *NIPS*, pp. 109–117.

Azimi, J., Fern, A., Fern, X. Z., Borradaile, G., & Heeringa, B. (2012). Batch active learning via coordinated matching. In *ICML*.

Azimi, J., Fern, X., Fern, A., Burrows, E., Chaplen, F., Fan, Y., Liu, H., Jaio, J., & Schaller, R. (2010). Myopic policies for budgeted optimization with constrained experiments. In *AAAI*.

Bond, D. R., & Lovley, D. R. (2003). Electricity production by geobacter sulfurreducens attached to electrodes. *Applications of Environmental Microbiology*, *69*, 1548–1555.

Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Brunato, M., Battiti, R., & Pasupuleti, S. (2006). A memory-based rash optimizer. In *AAAI-06 Workshop on Heuristic Search, Memory Based Heuristics and Their applications*.

Burrows, E. H., Wong, W.-K., Fern, X., Chaplen, F. W., & Ely, R. L. (2009). Optimization of ph and nitrogen for enhanced hydrogen production by synechocystis sp. pcc 6803 via statistical and machine learning methods. *Biotechnology Progress*, *25*, 1009–1017.

Chapelle, O., & Li, L. (2011). An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pp. 2249–2257.

Cox, D. D., & John, S. (1992). A statistical method for global optimization. In *IEEE Conference on Systems, Man and Cybernetics*, pp. 1241–1246.

Cox, D. D., & John, S. (1997). Sdo: A statistical method for global optimization. In *in Multidisciplinary Design Optimization: State-of-the-Art*, pp. 315–329.

Desautels, T., Krause, A., & Burdick, J. W. (2014). Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research*, *15*(1), 3873–3923.

Deshpande, A., Guestrin, C., Madden, S. R., Hellerstein, J. M., & Hong, W. (2004). Model-driven data acquisition in sensor networks. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pp. 588–599. VLDB Endowment.

Elder, J.F., I. (1992). Global rd optimization when probes are expensive: the grope algorithm. In *IEEE International Conference on Systems, Man and Cybernetics*, pp. 577–582.

Fan, Y., Hu, H., & Liu, H. (2007). Enhanced coulombic efficiency and power density of air-cathode microbial fuel cells with an improved cell configuration. *Journal of Power Sources*, In press.

Ginsbourger, D., Riche, R. L., & Carrarog, L. (2010). Kriging is well-suited to parallelize optimization..

Goel, A., Guha, S., & Munagala, K. (2006). Asking the right questions: model-driven optimization using probes. In *PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 203–212.

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

Khuller, S., Moss, A., & Naor, J. (1999). The budgeted maximum coverage problem. *Inf. Process. Lett.*, *70*(1), 39–45.

Krause, A., & Guestrin, C. (2005). A note on the budgeted maximization of submodular functions. *Technical Report*, *CMU-CALD-05-103*.

Krause, A., Singh, A., & Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: Theory, Efficient Algorithms and Empirical Studies. *The Journal of Machine Learning Research*, *9*, 235–284.

Lizotte, D., Madani, O., & Greiner, R. (2003). Budgeted learning of naive-bayes classifiers. In *UAI*.

Locatelli, M. (1997). Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, *10*(1), 57–76.

Madani, O., Lizotte, D., & Greiner, R. (2004). Active model selection. In *UAI*.

Moore, A., & Schneider, J. (1995). Memory-based stochastic optimization. In *NIPS*.

Moore, A., Schneider, J., Boyan, J., & Lee, M. S. (1998). Q2: Memory-based active learning for optimizing noisy continuous functions. In *ICML*, pp. 386–394.

Myers, R. H., Montgomery, D. C., & Anderson-Cook, C. M. (1995). *Response surface methodology: process and product optimization using designed experiments*. Wiley.

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

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT.

Reguera, G., McCarthy, K. D., Mehta, T., Nicoll, J. S., Tuominen, M. T., & Lovley, D. R. (2005). Extracellular electron transfer via microbial nanowires. *Nature*, 1098–1101.

Ross, A. M. (2008). Computing Bounds on the Expected Maximum of Correlated Normal Variables . *Methodology and Computing in Applied Probability*.

Schneider, J., & Moore, A. (2002). Active learning in discrete input spaces. In *Interface Symposium*.

Schonlau, M. (1997). *Computer Experiments and Global Optimization*. Ph.D. thesis, University of Waterloo.

Silberstein, A., Braynardand, R., Ellis, C., Munagala, K., & Yang, J. (2006). A sampling-based approach to optimizing top-k queries in sensor networks. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, p. 68, Washington, DC, USA. IEEE Computer Society.

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

Srinivas, N., Krause, A., Kakade, S., & Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. International Conference on Machine Learning (ICML)*.

Stuckman, B. E. (1988). A global search method for optimizing nonlinear systems. In *IEEE transactions on systems, man, and cybernetic*, Vol. 18, pp. 965–977.

Tatbul, N., Cetintemel, U., Zdonik, S. B., Cherniack, M., & Stonebraker, M. (2003). Load shedding in a data stream manager. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pp. 309–320. VLDB Endowment.