Active Information Acquisition for Linear Optimization

Shuran Zheng
School of Engineering
and Applied Sciences
Harvard University
Cambridge, MA

Bo Waggoner
The Warren Center for
Network and Data Sciences
Univerisity of Pennsylvania
Philadelphia, PA

Yang Liu
School of Engineering
and Applied Sciences
Harvard University
Cambridge, MA

Yiling Chen
School of Engineering
and Applied Sciences
Harvard University
Cambridge, MA

Abstract

We consider partially-specified optimization problems where the goal is to actively, but efficiently, acquire missing information about the problem in order to solve it. An algorithm designer wishes to solve a linear program (LP), $\max \mathbf{c}^T\mathbf{x}$ s.t. $\mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$, but does not initially know some of the parameters. The algorithm can iteratively choose an unknown parameter and gather information in the form of a noisy sample centered at the parameter's (unknown) value. The goal is to find an approximately feasible and optimal solution to the underlying LP with high probability while drawing a small number of samples. We focus on two cases. (1) When the parameters $\mathbf{b}$ of the constraints are initially unknown, we propose an efficient algorithm combining techniques from the ellipsoid method for LP and confidence-bound approaches from bandit algorithms. The algorithm adaptively gathers information about constraints only as needed in order to make progress. We give sample complexity bounds for the algorithm and demonstrate its improvement over a naive approach via simulation. (2) When the parameters $\mathbf{c}$ of the objective are initially unknown, we take an information-theoretic approach and give roughly matching upper and lower sample complexity bounds, with an (inefficient) successive-elimination algorithm.

1 INTRODUCTION

Many real-world settings are modeled as optimization problems. For example, a delivery company plans driver routes to minimize the driver's total travel time; an airline assigns vehicles to different origin-destination pairs to maximize profit. However, in practice, some parameters of the optimization problems may be initially unknown. The delivery company may not know the average congestion or travel time of various links of the network, but has ways to poll Waze[1] drivers to get samples of travel times on links of the network. The delivery company may not know the demand and the revenues for each origin-destination pair, but can get estimates of them by selling tickets on chosen origin-destination pairs.

To capture such settings, this paper proposes a model of optimization wherein the algorithm can iteratively choose a parameter and draw a "sample" that gives information about that parameter; specifically, the sample is an independent draw from a subgaussian random variable centered at the true value of the parameter. This models, for instance, observing the congestion on a particular segment of road on a particular day. Drawing each sample is presumed to be costly, so the goal of the algorithm is to draw the fewest samples necessary in order to find a solution that is approximately feasible and approximately optimal.

Thus, the challenge falls under an umbrella we term *active information acquisition for optimization (AIAO)*. The key feature of the AIAO setting is the structure of the optimization problem itself, i.e. the objective and constraints. The challenge is to understand how the difficulty of information acquisition relates to this underlying structure. For example, are there information-theoretic quantities relating the structure to the sample complexity? Meanwhile, the opportunity of AIAO is to exploit algorithms for the underlying optimization problem. For example, can one interface with the algorithm to reduce sample complexity by only acquiring the information needed, as it is needed?

These are the questions investigated in this paper, which focuses on active information acquisition for linear opti-

[1]https://www.waze.com

mization problems. Specifically, we consider linear programs in the form

$$\max_{\mathbf{x}} \mathbf{c}^T\mathbf{x}, \text{ s.t. } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \tag{1}$$

with $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{c} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^m$. We will consider either the case that the $\mathbf{b}$ in the constraints is unknown or else the case that the $\mathbf{c}$ in the objective is unknown, with all other parameters initially known to the algorithm. The algorithm can iteratively choose an unknown parameter, e.g. b_i, and draw a "sample" from it, e.g. observing $b_i + \eta$ for an independent, zero-mean, subgaussian η. The algorithm must eventually output a solution $\mathbf{x}$ such that, with probability $1 - \delta$, $\mathbf{A}\mathbf{x} \leq \mathbf{b} + \varepsilon_1 \mathbf{1}$ and $\mathbf{c}^T\mathbf{x} \geq \mathbf{c}^T\mathbf{x}^* - \varepsilon_2$, where $\mathbf{x}^*$ is the optimal solution. The goal is for the algorithm to achieve this while using as few total samples as possible.

There is a natural "naive" or "static" approach: draw samples for all unknown parameters until they are known to high accuracy with high probability, then solve the "empirical" linear program. However, we can hope to improve by leveraging known algorithms and properties of linear programs. For example, in the case that $\mathbf{b}$ is unknown, if a linear program has an optimal solution, it has an optimal solution that is an extreme point (a corner point of the feasible region); and at this extremal optimal solution, several constraints are binding. These suggest that it is more important to focus on binding constraints and to gather information on the differing objective values of extreme points. Algorithms developed in this paper leverage these properties of linear programs to decide how much information to acquire for each unknown parameter.

1.1 APPROACHES AND RESULTS

Two settings and our approaches. The paper investigates two settings: unknown $\mathbf{b}$ but known $\mathbf{c}$, and unknown $\mathbf{c}$ but known $\mathbf{b}$. We always suppose $\mathbf{A}$ is known and assume that the linear program has an optimal solution.

It might initially appear that these cases are "equivalent" via duality theory, but we argue that the two cases are quite different when we do not know the parameters of LP exactly. Given a *primal* linear program of the form (1), the *dual* program is given by $\min_{\mathbf{y}} \mathbf{b}^T\mathbf{y}$ s.t. $\mathbf{A}^T\mathbf{y} \geq \mathbf{c}, \mathbf{y} \geq \mathbf{0}$, which is easily transformed into the maximization format of (1). In particular, the parameters $\mathbf{c}$ in the objective function of a primal LP becomes the parameters in the constraints of the dual LP. By duality theory, the (exact) optimal solutions to the primal and dual are connected by *complementary slackness* conditions. However, this approach breaks down in the approximate setting for two reasons. First, approximately optimal solutions do not satisfy complementary slackness; and second, even knowing which constraints bind does not suffice to determine the optimal solution $\mathbf{x}^*$ when some of the constraint or objective parameters are unknown.[2] We hence take two different approaches toward our two settings.

Unknown-$\mathbf{b}$ case. In the unknown $\mathbf{b}$ setting, the uncertainty is over the constraints. Our algorithm combines two main ideas: the ellipsoid method for solving linear programs, and multi-armed bandit techniques for gathering data. The ellipsoid algorithm only requires the information of one violated constraint at each iteration, if there exists a violated one. We then leverage the multi-armed bandits method to find the most violated constraint (if there exists one) using as few samples as possible.

We theoretically bound the number of samples drawn by our algorithm as a function of the parameters of the problem. In simulations on generated linear programs, UCB-Ellipsoid far outperforms the naive approach of sampling all parameters to high accuracy, and approaches the performance of an oracle that knows the binding constraints in advance and needs only to sample these. In other words, the algorithm spends very few resources on unimportant parameters.

Unknown-$\mathbf{c}$ case. In the unknown-objective setting, we know the feasible region exactly. Our algorithm focuses only on the set of extreme points of the feasible region. For each of the extreme point, there are a set of possible values for $\mathbf{c}$ such that if $\mathbf{c}$ takes any value in the set, this extreme point is the optimal solution to the LP. The algorithm hence draws just enough samples to determine with high probability which is actually the case for the true $\mathbf{c}$.

We define an information-theoretic measure, $Low(\mathcal{I})$ for an instance $\mathcal{I}$. We show that this quantity essentially characterizes the sample complexity of a problem instance and we give an algorithm, not necessarily efficient, for achieving it up to low-order factors.

2 RELATED WORK

The setting considered in this paper, active information acquisition for optimization, is related at a conceptual

[2]Nor does knowing which constraints bind even necessarily help, as approximately satisfying them may still lead to large violations of other constraints. Thus, while we do not rule out some future approach that connects approximate solutions of the primal and dual, the evidence suggests to us that the two settings are quite different and we approach each differently in this paper.

level to a large number of lines of work that deal with optimization and uncertainty. But it differs from them mostly in either the active aspect or the optimization aspect. We'll discuss some of these related lines of work and the differences below.

Optimization under uncertainty Our problem considers optimization with uncertain model parameters, which is also a theme in stochastic programming [Heyman and Sobel, 2003, Neely, 2010], chance-constrained programming [Ben-Tal et al., 2009], and robust optimization [Ben-Tal et al., 2009, Bertsimas et al., 2004]. In both stochastic optimization and chance-constrained programming, there are no underlying true, fixed values of the parameters; instead, a probabilistic, distributional model of the parameters is used to capture the uncertainty and such model of uncertainty becomes part of the optimization formulation. Hence, optimality in expectation or approximate optimality is sought after under the probabilistic model. But in our problem underlying fixed parameters exist, and the problem is only how much information to gather about them. Meanwhile, robust optimization models deterministic uncertainty (e.g. parameters come from a known set) and often seeks for a worst-case solution, *i.e.* a solution that is feasible in the worst case over a set of possible constraints. A key distinction of our model is that there are underlying true values of the parameters and we do not incorporate any probabilistic or deterministic model of the uncertainty into the optimization problem itself. Instead, we take an "active querying" approach to approximately solve the true optimization problem with high probability.

Artificial intelligence. Several existing lines of work in the artificial intelligence literature, deal with actively acquiring information about parameters of an optimization problem in order to solve it. Preference elicitation [Braziunas, 2006, Blum et al., 2004] typically focuses on acquiring information about parameters of the *objective* by querying a user about his preferences, this is similar to our goal for the unknown-$\mathbf{c}$ setting. Relevant to our unknown-$\mathbf{b}$ case, for more combinatorial problems, the constraint acquisition literature [OConnell et al., 2002, Bessiere et al., 2015] is closer to our problem in some respects, as it posits an optimization problem with unknown constraints that must be learned via interactive querying. We emphasize that a central feature of the model in this paper is *noisy observations*: the observations of the algorithm are only noisy samples of a true underlying parameter. The key challenge is to choose how many repeated samples of each parameter to draw. This aspect of the problem is not to our knowledge present in preference elicitation or model/constraint acquisition.

Machine learning and theoretical computer science. Much work in *active learning* considers acquiring data points iteratively with a goal of low sample complexity [Balcan et al., 2006, 2007, Castro and Nowak, 2008].The key difference to AIAO is between *data* and *parameters*. In learning, the goal is to minimize the average or expectation of some loss function over a distribution of data points. Other than its likelihood, each data point plays the same role in the problem. Here, the focus is on how much information about each of various parameters is necessary to solve a structured optimization problem to a desired level of accuracy. In other words, the key question here, which is how much an optimization algorithm needs to know about the parameters of the problem it is solving, does not apply in active learning.

A line of work on sample complexity of reinforcement learning (or approximate reinforcement learning) [Kakade et al., 2003, Lattimore and Hutter, 2012, Azar et al., 2012, Wang, 2017, Chen and Wang, 2016] also bears some resemblance to our problem. A typical setting considered is solving a model-free Markov Decision Processes (MDP), where the transition probabilities and the reward functions are initially unknown but the algorithm can query an oracle to get samples. This problem is a special case of our AIALO problem with unknown $\mathbf{A}$ and $\mathbf{b}$ because an MDP can be formulated as an linear program. The solutions provided focuses on the particular structure of MDP, while we consider general linear programs.

Broadly related is recent work on *optimization from samples* Balkanski et al. [2016], which considers the sample complexity of a two-stage process: (1) draw some number of i.i.d. data points; (2) optimize some loss function or submodular function on the data. In that setting, the algorithm sees a number of input-output pairs of the function, randomly distributed, and must eventually choose a particular input to optimize the function. Therefore, it is quite different from our setting in both important ways: (1) the information collected are data points (and evaluations), as in ML above, rather than parameters as in our problem; (2) (so far) it is not active, but acquires information in a batch.

A line of work that is closely related to our unknown-$\mathbf{c}$ problem is the study of *combinatorial pure exploration* (CPE) problem, where a learner collects samples of unknown parameters of an objective function to identify the optimal member in a solution set. The problem was first proposed in Chen et al. [2014], and subsequently studied by Gabillon et al. [2016], Chen et al. [2016a,b, 2017]. CPE only considers combinatorial optimization problems whose solution set contains only binary vectors of length n. A recent work by Chen et al. [2017]

extended CPE to a *General-Sampling* problem by allowing general solution sets. Our unknown-$\mathbf{c}$ problem can be fitted into the setting of General-Sampling. Our algorithm for unknown-$\mathbf{c}$ was inspired by the work of Chen et al. [2017], but leverages the structure of LP and hence has better sample complexity performance than directly treating it as a General-Sampling problem. The General-Sampling problem does not encompass all AIAO settings, e.g., our unknown-$\mathbf{b}$ case.

3 MODEL AND PRELIMINARIES

3.1 THE AIALO PROBLEM

We now formally define an instance $\mathcal{I}$ of the *active information acquisition for linear optimization (AIALO)* problem. We then describe the format of algorithms for solving this problem. Note that one can easily extend this into a more general formal definition of AIAO, for more general optimization problems, but we leave this for future work.

An instance $\mathcal{I}$ consists of three components. The first component consists of the *parameters* of the underlying linear program on n variables and m constraints: a vector $\mathbf{c} \in \mathbb{R}^n$, a vector $\mathbf{b} \in \mathbb{R}^m$, and a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$. Naturally, these specify the program[3]

$$\max_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \text{ s.t. } \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0}. \tag{2}$$

We assume for simplicity in this paper that all linear programs are feasible and are known *a priori* to have a solution of norm at most R. The second component specifies which parameters are *initially known* and which are *initially unknown*. The third and final component specifies, for each unknown parameter (say c_i), of a σ^2-subgaussian distribution with mean equal to the value of the parameter.[4]

Given $\mathcal{I}$, we define the following sets of approximately-feasible, approximately-optimal solutions.

Definition 3.1. *Given an instance $\mathcal{I}$, let $\mathbf{x}^*$ be an optimal solution to the LP. Define $OPT(\mathcal{I}; \varepsilon_1, \varepsilon_2)$ to be the set of solutions $\mathbf{x}$ satisfying $\mathbf{c}^T\mathbf{x} \ge \mathbf{c}^T\mathbf{x}^* - \varepsilon_1$ and $\mathbf{A}\mathbf{x} \le \mathbf{b} + \varepsilon_2 \mathbf{1}$. We use $OPT(\mathcal{I})$ as shorthand for $OPT(\mathcal{I}; 0, 0)$.*

[3]Note that any linear program can be transformed into the given format with at most a factor 2 increase in n and m.

[4]Distribution $\mathcal{D}$ with mean μ is σ^2-subgaussian if, for $X \sim \mathcal{D}$, we have $\mathbb{E}[e^{t(X-\mu)}] \le e^{\sigma^2 t^2/2}$ for all t. The family of sub-Gaussian distributions with parameter σ encompasses all distributions that are supported on $[0, \sigma]$ as well as many unbounded distributions such as Gaussian distributions with variance σ^2.

3.2 ALGORITHM SPECIFICATION

An algorithm for the AIALO problem, run on an instance $\mathcal{I}$, functions as follows. The algorithm is given as input n (number of variables), m (number of constraints), and σ^2 (subgaussian parameter). It is also given the second component of the instance $\mathcal{I}$, i.e. a specification of which parameters are known and which are unknown. For each parameter that is specified as "known", the algorithm is given the value of that parameter, e.g. it is given "$A_{11} = 42$." Finally, the algorithm is given an optimality parameter ε_1, a feasibility parameter ε_2, and a failure probability parameter δ.

The algorithm may iteratively choose an unknown parameter and *sample* that parameter: observe an independent and identically-distributed draw from the distribution corresponding to that parameter (as specified in the third component of the instance $\mathcal{I}$). At some point, the algorithm stops and outputs a *solution* $\mathbf{x} \in \mathbb{R}^n$.

Definition 3.2 (($\delta, \varepsilon_1, \varepsilon_2$)-correct algorithm)**.** *An algorithm $\mathcal{A}$ is $(\delta, \varepsilon_1, \varepsilon_2)$-*correct *if for any instance $\mathcal{I}$ and inputs $(\delta, \varepsilon_1, \varepsilon_2)$, with probability at least $1 - \delta$, $\mathcal{A}$ outputs a solution $\mathbf{x} \in OPT(\mathcal{I}; \varepsilon_1, \varepsilon_2)$. In the case $\varepsilon_1 = \varepsilon_2 = 0$, we say $\mathcal{A}$ is δ-*correct.

Our goal is to find algorithms with low *sample complexity*, i.e., the number of samples drawn by the algorithm.

4 UNKNOWN CONSTRAINTS

We will first consider the *unknown-$\mathbf{b}$ case* where every parameter of the constraint vector $\mathbf{b}$ is initially unknown, and all other parameters are initially known. Geometrically, the algorithm is given an objective "direction" ($\mathbf{c}$) and a set of constraint "orientations" ($\mathbf{A}$), but does not initially know the "offset" or displacement b_i of each constraint i.

In this setting, we do not expect to attain either exact feasibility or exact optimality, as the exact constraints can never be known, and in general an arbitrarily small change in constraints of an LP leads to a nonzero change in the value of the optimal solution.

The section begins with a lower bound of the sample complexity (across all LP instances) of any correct algorithm.

Theorem 4.1 (Lower bound for unknown $\mathbf{b}$)**.** *Suppose we have a $(\delta, \varepsilon_1, \varepsilon_2)$-correct algorithm $\mathcal{A}$ where $\delta \in (0, 0.1), \varepsilon_1 > 0, \varepsilon_2 > 0$. Then for any $n > 0$, there exists infinitely many instances of the AIALO problem with unknown-$\mathbf{b}$ with n variables with objective function $\|\mathbf{c}\|_\infty = 1$ such that $\mathcal{A}$ must draw at least*

$$\Omega\left(n \ln(1/\delta) \cdot \max\{\varepsilon_1, \varepsilon_2\}^{-2}\right)$$

samples in expectation on each of them.

The idea of the lower bound (proof in Appendix C.1) is that in the worst case, an algorithm must accurately estimate at least all n binding constraints (in general with n variables, up to n constraints bind at the optimal solution). It remains open whether we can get a tighter lower bound which also captures the difficulty of ruling out non-binding constraints.

4.1 ELLIPSOID-UCB ALGORITHM

Background. The standard ellipsoid algorithm for linear programming begins with an ellipsoid $\mathcal{E}^{(0)}$ known to contain the optimal solution, Then, it checks two cases: (1) The center of this ellipsoid $\mathbf{x}^{(0)}$ is feasible, or (2) it is not feasible. If (2), say it violates constraint i, then the algorithm considers the halfspace defined by the constraint $\mathbf{A}_i\mathbf{x}^{(0)} \leq b_i$. If (1), the algorithm considers the halfspace defined by the "constraint" $\mathbf{c}^T\mathbf{x} \geq \mathbf{c}^T\mathbf{x}^{(0)}$, as the optimal solution must satisfy this constraint. In either case, it updates to a new ellipsoid $\mathcal{E}^{(1)}$ defined as the minimal ellipsoid containing the intersection of $\mathcal{E}^{(0)}$ with the halfspace under consideration.

The obstacle is that, now, $\mathbf{b}$ is initially unknown. A first observation is that we only need to find a single violated constraint, so there may be no need to sample most parameters at a given round. A second observation is that it suffices to find the **most violated** constraint. This can be beneficial as it may require only a few samples to find the most violated constraint; and in the event that no constraint is violated, we still need to find an upper bound on "closest to violated" constraint in order to certify that no constraint is violated.

To do so, we draw inspiration from algorithms for *bandits* problems (whose details are not important to this paper). Suppose we have m distributions with means $\mu_1, \dots, \mu_m$ and variances $\sigma_1^2, \dots, \sigma_m^2$, and we wish to find the largest μ_i. After drawing a few samples from each distribution, we obtain estimates $\widehat{\mu}_i$ along with confidence intervals given by tail bounds. Roughly, an "upper confidence bound" (UCB) algorithm (see *e.g.* Jamieson and Nowak [2014]) for finding $\max_i \mu_i$ proceeds by always sampling the i whose upper confidence bound is the highest.

We therefore will propose a UCB-style approach to doing so, but with the advantage that we can re-use any samples from earlier stages of the ellipsoid algorithm.

Algorithm and results. Ellipsoid-UCB is given in Algorithm 1. At each round $k = 1, 2, \dots$, we choose the center point $\mathbf{x}^{(k)}$ of the current ellipsoid $\mathcal{E}^{(k)}$ and call the subroutine Algorithm 2 to draw samples and check for violated constraints. We use the result of the oracle to cut the current space exactly as in the standard ellipsoid method, and continue.

Some notation: t is used to track the total number of samples drawn (from all parameters) and $T_i(t)$ denotes the number of samples of b_i drawn up to "time" t. The average of these samples is:

Definition 4.1. *Let $X_{i,s}$ denote the s-th sample of b_i and let $T_i(t)$ denote the number of times b_i is sampled in the first t samples. Define $\widehat{b}_{i,T_i(t)} = \sum_{s=1}^{T_i(t)} X_{i,s}/T_i(t)$ to be the empirical mean of b_i up to "time" t.*

Algorithm 1 Modified ellipsoid algorithm

```
Let $\mathcal{E}^{(0)}$ be the initial ellipsoid containing the feasible region.
Draw one sample for each $b_i$, $i \in [m]$.
Let $k = 0$ and $t = m$.
Let $T_i(t) = 1$ for all $i$.
while stopping criterion is not met[5] do
    Let $\mathbf{x}^{(k)}$ be the center of $\mathcal{E}^{(k)}$
    Call UCB method to get constraint $i$ or "feasible"
    if $\mathbf{x}^{(k)}$ is feasible then
        $\mathbf{x} \leftarrow \mathbf{x}^{(k)}$ if $\mathbf{x}$ not initialized or $\mathbf{c}^T\mathbf{x}^{(k)} > \mathbf{c}^T\mathbf{x}$.
        $\mathbf{y} \leftarrow -\mathbf{c}$
    else
        $\mathbf{y} \leftarrow \mathbf{A}_i^T$
    end if
    Let $\mathcal{E}^{(k+1)}$ be the minimal ellipsoid that contains $\mathcal{E}^{(k)} \cap \{\mathbf{p} : \mathbf{y}^T\mathbf{p} \leq \mathbf{y}^T\mathbf{x}^{(k)}\}$
    Let $k \leftarrow k + 1$
end while
Output $\mathbf{x}$ or "failure" if it was never set.
```

The performance of the algorithm is measured by how many samples (observations) it needs to draw. To state our theoretical results, define $V_i(k) = \mathbf{A}_i\mathbf{x}^{(k)} - b_i$ to be the amount by which the i-th constraint is violated by $\mathbf{x}^{(k)}$, and $V^*(k) = \max_i V_i(k)$. Define $gap_{i,\varepsilon}(k) = \max\{|V_i(k)|, V^*(k) - V_i(k), \varepsilon\}$ and $\Delta_{i,\varepsilon} = \min_k gap_{i,\varepsilon}(k)$.

Theorem 4.2 (Ellipsoid-UCB algorithm)**.** *The Ellipsoid-UCB algorithm is $(\delta, \epsilon_1, \epsilon_2)$-correct and with probability $1 - \delta$, draws at most the following number of samples:*

$$O\left(\sum_{i=1}^{m} \frac{\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2} \log\frac{m}{\delta} + \sum_{i=1}^{m} \frac{\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2} \log\log\left(\frac{\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2}\right)\right).$$

Specifically, the number of samples used for b_i is at most $\frac{\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2}\left(\log(m/\delta) + \log\log(\sigma_i^2/\Delta_{i,\varepsilon_2/2}^2)\right)$.

[5]Our stopping criterion is exactly the same as in the standard ellipsoid algorithm, for which there are a variety of possible criteria that work. In particular, one is $\sqrt{\mathbf{c}^T\mathbf{P}^{-1}\mathbf{c}} \leq \min\{\varepsilon_1, \varepsilon_2\}$, where P is the matrix corresponding to ellipsoid $\mathcal{E}^{(k)}$ as discussed above.

Algorithm 2 UCB-method

```
Input $\mathbf{x}^{(k)}$
Output either index $j$ of a violated constraint, or "feasible".
Set $\delta' = \left(\frac{\delta}{20m}\right)^{2/3}$
loop
    1. Let $j$ be the constraint with the largest index,
       $$j = \arg\max_i \mathbf{A}_i\mathbf{x}^{(k)} - \widehat{b}_{i,T_i(t)} + U_i(T_i(t)),$$
       where $U_i(s) = 3\sqrt{2\sigma_i^2 \log\left(\log(3s/2)/\delta'\right)/s}$ and $\widehat{b}_{i,T_i(t)}$ as in Definition 4.1.
    2. If $\mathbf{A}_j\mathbf{x}^{(k)} - \widehat{b}_{j,T_j(t)} - U_j(T_j(t)) > 0$ return $j$.
    3. If $\mathbf{A}_j\mathbf{x}^{(k)} - \widehat{b}_{j,T_j(t)} + U_j(T_j(t)) < 0$ return "feasible".
    4. If $U_j(T_j(t)) < \varepsilon_2/2$ return "feasible".
    5. Let $t \leftarrow t + 1$
    6. Draw a sample of $b_j$.
    7. Let $T_j(t) = T_j(t-1) + 1$.
    8. Let $T_i(t) = T_i(t-1)$ for all $i \neq j$.
end loop
```

Proof Sketch: Define event $\mathcal{A}$ to be the event that $\left|\widehat{b}_{i,s} - b_i\right| \leq U_i(s)$ for all s and $i \in [m]$. According to Lemma 3 in Jamieson et al. [2014], $\mathcal{A}$ holds with probability at least $1 - \delta$.

Correctness: Conditioning on event $\mathcal{A}$ holds, our UCB method will only return a constraint that is violated (line 2) and when it returns "feasible", no constraint is violated more than ε_2 (line 3 and 4).

Number of samples: We bound the number of samples used on each constraint separately. Consider a fixed ellipsoid iteration k in which UCB method is given input $\mathbf{x}^{(k)}$, the key idea is to prove that if b_i is sampled in this iteration at "time" t, $U_i(T_i(t))$ should be larger than $gap_{i,\varepsilon_2/2}(k)$. This gives an upper bound of $T_i(t)$. Taking the maximum of them, we get the final result.

□

Discussion. To understand the bound, suppose for simplicity that each $\sigma_i = 1$. We observe that the first term will dominate in all reasonable parameter settings, so we can ignore the second summation in this discussion.

Next, note that each term in the sum reflects a bound on how many times constraint i must be sampled over the course of the algorithm. This depends inversely on $\Delta_{i,\varepsilon_2/2}$, which is a minimum over all stages k of the "precision" we need of constraint i at stage k. We only need a very precise estimate if both of the following conditions are satisfied: (1) $|V_i(k)|$ is small, meaning that the ellipsoid center $\mathbf{x}^{(k)}$ is very close to binding constraint i; (2) There is no other constraint that is significantly violated, meaning that i is very close to the most-violated constraint for $\mathbf{x}^{(k)}$ if any. Because this is unlikely to happen for most constraints, we expect $\Delta_{i,\varepsilon_2/2}$ to generally be large (leading to a good bound), although we do not have more precise theoretical bounds. The only constraints where we might expect $\Delta_{i,\varepsilon_2/2}$ to be small are the binding constraints, which we expect to come close to satisfying the above two conditions at some point. Indeed, this seems inevitable for any algorithm, as we explore in our experiments.

Comparison to static approach. Again suppose each $\sigma_i = 1$ for simplicity. Note that each $\Delta_{i,\varepsilon_2/2} \geq \varepsilon_2/2$. This implies that our bound is always better than $O\left(\frac{m\log(m/\delta)}{\varepsilon_2^2}\right)$, ignoring the dominated second term.

The static approach is to measure each b_i with enough samples to obtain a good precision so that relaxed feasibility can be satisfied with high probability, then solve the linear program using the estimated constraints. This uses $\frac{4m\log(m/\delta)}{\varepsilon_2^2}$ samples. (This number comes from using tail bounds to ensure good precision is achieved on every b_i.)

Therefore, the UCB-Ellipsoid algorithm dominates the static approach up to some constant factors and can show dramatic instance-wise improvements. Indeed, in some simple cases, such as the number of variables equal to the number of constraints, we do not expect any algorithm to be able to improve over the static approach. However, a nice direction for future work is to show that, if m is very large compared to n, then the UCB-Ellipsoid algorithm (or some other algorithm) is guaranteed to asymptotically improve on the static approach.

5 UNKNOWN OBJECTIVE FUNCTION

In this section, we consider the *unknown-$\mathbf{c}$ case*. Here, every parameter of the objective $\mathbf{c}$ is initially unknown, and all other parameters are initially known. Geometrically, the algorithm is initially given an exact description of the feasible polytope, in the form of $\mathbf{Ax} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$, but no information about the "direction" of the objective.

Because the constraints are known exactly, we focus on exact feasibility in this setting, i.e. $\varepsilon_2 = 0$. We also focus on an information-theoretic understanding of the problem, and produce an essentially optimal but computationally inefficient algorithm. We assume that there is a unique optimal solution $\mathbf{x}^*$,[6] and consider the problem of

[6]If we only aim for a ε_1-suboptimal solution, we can terminate our algorithm when $\varepsilon^{(r)}$ (defined in Line 5 of Algorithm 3) becomes smaller than $\varepsilon_1/2$, such that the algorithm no longer requires the best point to be unique.

finding an exact optimal solution with confidence δ (i.e., a δ-correct algorithm). We also make the simplifying assumption that each parameter's distribution is a Gaussian of variance 1 (in particular is 1-subgaussian). Our results can be easily extend to the general case.

Our approaches are based on the techniques used in Chen et al. [2017], but address a different class of optimization problems. We thus use the same notations as in Chen et al. [2017]. We first introduce a function $Low(\mathcal{I})$ that characterizes the sample complexity required for an LP instance $\mathcal{I}$. The function $Low(\mathcal{I})$ is defined by the solution of a convex program. We then give an instance-wise lower bound in terms of the $Low(\mathcal{I})$ function and the failure probability parameter δ. We also formulate a worst-case lower bound of the problem, which is polynomially related to the instance-wise lower bound. Finally, we give an algorithm based on successive elimination that matches the worst-case lower bound within a factor of $\ln(1/\Delta)$, where Δ is the gap between the objective function value of the optimal extreme point ($\mathbf{x}^*$) and the second-best.

5.1 LOWER BOUNDS

The function $Low(\mathcal{I})$ is defined as follows.

Definition 5.1 ($Low(\mathcal{I})$)**.** *For any instance $\mathcal{I}$ of AIALO (or more generally, for any linear program), we define $Low(\mathcal{I}) \in \mathbb{R}$ to be the optimal solution to the following convex program.*

$$\begin{aligned} \min_{\tau} \quad & \sum_{i=1}^{n} \tau_i && (3) \\ s.t. \quad & \sum_{i=1}^{n} \frac{(s_i^{(k)} - x_i^*)^2}{\tau_i} \le \left(\mathbf{c}^T(\mathbf{x}^* - \mathbf{s}^{(k)})\right)^2, \forall k \\ & \tau_i \ge 0, \forall i \end{aligned}$$

Here $\mathbf{x}^*$ is the optimal solution to the LP in $\mathcal{I}$ and $\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(k)}$ are the extreme points of the feasible region $\{\mathbf{x} : \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0}\}$.

For intuition about $Low(\mathcal{I})$, consider a thought experiment where we are given an extreme point $\mathbf{x}^*$, and we want to check whether or not $\mathbf{x}^*$ is the optimal solution using as few samples as possible. Given our empirical estimate $\widehat{\mathbf{c}}$ we would like to have enough samples so that with high probability, for each $\mathbf{s}^{(k)} \ne \mathbf{x}^*$, we have

$$\widehat{\mathbf{c}}^{\mathrm{T}}(\mathbf{x}^* - \mathbf{s}^{(k)}) > 0 \iff \mathbf{c}^{\mathrm{T}}(\mathbf{x}^* - \mathbf{s}^{(k)}) > 0.$$

This will hold by a standard concentration bound (Lemma D.2) *if enough samples of each parameter are drawn*; in particular, "enough" is given by the k-th constraint in (3).

Theorem 5.1 (Instance lower bound)**.** *Let $\mathcal{I}$ be an instance of AIALO in the unknown-$\mathbf{c}$ case. For $0 < \delta < 0.1$, any δ-correct algorithm $\mathcal{A}$ must draw*

$$\Omega(Low(\mathcal{I}) \ln \delta^{-1})$$

samples in expectation on $\mathcal{I}$.

We believe that it is unlikely for an algorithm to match the instance-wise lower bound without knowing the value of $\mathbf{c}$ and $\mathbf{x}^*$ in the definition of $Low(\mathcal{I})$. To formally prove this claim, for any δ-correct algorithm $\mathcal{A}$, we construct a group of LP instances that share the same feasible region $\mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0}$ but have different objective functions and different optimal solutions. We prove that $\mathcal{A}$ will have unmatched performance on at least one of these LP instances.

Our worst-case lower bound can be stated as follows.

Theorem 5.2 (Worst-case lower bound for unknown $\mathbf{c}$)**.** *Let n be a positive integer and $\delta \in (0, 0.1)$. For any δ-correct algorithm $\mathcal{A}$, there exists an infinite sequence of LP instances with n variables, $\mathcal{I}_1, \mathcal{I}_2, \dots$, such that $\mathcal{A}$ takes*

$$\Omega\left(Low(\mathcal{I}_k)(\ln |S_k^{(1)}| + \ln \delta^{-1})\right)$$

samples in expectation on $\mathcal{I}_k$, where $S_k^{(1)}$ is the set of all extreme points of the feasible region of $\mathcal{I}_k$, and $Low(\mathcal{I}_k)$ goes to infinity.

5.2 SUCCESSIVE ELIMINATION ALGORITHM

Before the description of the algorithm, we first define a function $LowAll(S, \varepsilon, \delta)$ that indicates the number of samples we should take for each c_i, such that the difference in objective value between any two points in S can be estimated to an accuracy ε with probability $1 - \delta$. Define $LowAll(S, \varepsilon, \delta)$ to be the optimal solution of the following convex program,

$$\begin{aligned} \min_{\tau} \quad & \sum_{i=1}^{n} \tau_i && (4) \\ s.t. \quad & \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\tau_i} \le \frac{\varepsilon^2}{2\ln(2/\delta)}, \forall \mathbf{x}, \mathbf{y} \in S \\ & \tau_i \ge 0, \forall i. \end{aligned}$$

Our algorithm starts with a set $S^{(1)}$ that contains all extreme points of the feasible region $\{\mathbf{x} : \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0}\}$, which is the set of all possible optimal solutions. We first draw samples so that the difference between each pairs in $S^{(1)}$ is estimated to accuracy $\varepsilon^{(1)}$. Then we delete all points that are not optimal with high probability. In the next iteration, we halve the accuracy

$\varepsilon^{(2)} = \varepsilon^{(1)}/2$ and repeat the process. The algorithm terminates when the set contains only one point.

Algorithm 3 A successive elimination algorithm

```
1: $S^{(1)} \leftarrow$ set of all extreme points of feasible region $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$
2: $r \leftarrow 1$
3: $\lambda \leftarrow 10$
4: while $|S^{(r)}| > 1$ do
5:     $\varepsilon^{(r)} \leftarrow 2^{-r}$, $\delta^{(r)} \leftarrow \delta/(10r^2|S^{(1)}|^2)$
6:     $(t_1^{(r)}, \ldots, t_n^{(r)}) \leftarrow LowAll(S^{(r)}, \varepsilon^{(r)}/\lambda, \delta^{(r)})$
7:     Sample $c_i$ for $t_i^{(r)}$ times. Let $\widehat{c}_i^{(r)}$ be its empirical mean
8:     Let $\mathbf{x}^{(r)}$ be the optimal solution in $S^{(r)}$ with respect to $\widehat{\mathbf{c}}^{(r)}$
9:     Eliminate the points in $S^{(r)}$ that are $\varepsilon^{(r)}/2 + 2\varepsilon^{(r)}/\lambda$ worse than $\mathbf{x}^{(r)}$ when the objective function is $\widehat{\mathbf{c}}^{(r)}$,
           $$S^{(r+1)} \leftarrow \{\mathbf{x} \in S^{(r)} : \langle \mathbf{x}, \widehat{\mathbf{c}}^{(r)} \rangle \geq \langle \mathbf{x}^{(r)}, \widehat{\mathbf{c}}^{(r)} \rangle - \varepsilon^{(r)}/2 - 2\varepsilon^{(r)}/\lambda\} \quad (5)$$
10:    $r \leftarrow r + 1$
11: end while
12: Output $\mathbf{x} \in S^{(r)}$
```

The algorithm has the following sample complexity bound.

Theorem 5.3 (Sample complexity of Algorithm 3). *For the AIALO with unknown-$\mathbf{c}$ problem, Algorithm 3 is δ-correct and, on instance $\mathcal{I}$, with probability $1 - \delta$ draws at most the following number of samples:*

$$O\left(Low(\mathcal{I}) \ln \Delta^{-1} (\ln |S^{(1)}| + \ln \delta^{-1} + \ln \ln \Delta^{-1})\right),$$

where $S^{(1)}$ is the set of all extreme points of the feasible region and Δ is the gap in objective value between the optimal extreme point and the second-best,

$$\Delta = \max_{\mathbf{x} \in S^{(1)}} \mathbf{c}^T \mathbf{x} - \max_{\mathbf{x} \in S^{(1)} \setminus \mathbf{x}^*} \mathbf{c}^T \mathbf{x}.$$

Proof Sketch: We prove that conditioning on a good event $\mathcal{E}$ that holds with probability at least $1 - \delta$, the algorithm will not delete the optimal solution and will terminate before $\lfloor \log(\Delta^{-1}) \rfloor + 1$ iterations. Then we bound the number of samples used in iteration r by showing that the optimal solution of $Low(\mathcal{I})$ times $\alpha^{(r)} = 32\lambda^2 \ln(2/\delta^{(r)})$ is a feasible solution of the convex program that defines $LowAll(S^{(r)}, \varepsilon^{(r)}/\lambda, \delta^{(r)})$. Therefore the number of samples used in iteration r is no more than $\alpha^{(r)} Low(\mathcal{I})$. $\square$

This matches the worst-case lower bound within a problem-dependent factor $\ln(1/\Delta)$. Notice, however, that the size of $|S^{(1)}|$ can be exponentially large, and so is the size of the convex program (4). So Algorithm 3 is computationally inefficient if implemented straightforwardly, and it remains open whether the algorithm can be implemented in polynomial time or an alternative algorithm with similar sample complexity and better performance can be found.

6 EXPERIMENTS

In this section, we investigate the empirical number of samples used by Ellipsoid-UCB algorithm for the unknown-$\mathbf{b}$ case of AIALO. We fix $\delta = 0.1$ and focus on the impact of the other parameters[7], which are more interesting.

We compare three algorithms on randomly generated LP problems. The first is Ellipsoid-UCB. The second is the naive "static approach", namely, draw $4\sigma^2 \log(m/\delta)/\varepsilon_2^2$ samples of each constraint, then solve the LP using estimated means of the parameters. (This is the same approach mentioned in the previous section, except that previously we discussed the case $\sigma = 1$ for simplicity.) The third is designed to intuitively match the lower bound of Theorem 4.1: Draw $4\sigma^2 \log(d/\delta)/\varepsilon_2^2$ samples of each of only the binding constraints, where there are d of them, then solve the LP using estimated means of the b_i. (For a more fair comparison, we use the same tail bound to derive the number of samples needed for high confidence, so that the constants match more appropriately.)

We generate instances as follows. $\mathbf{c}$ is sampled from $[-10, 10]^n$ uniformly at random. $\mathbf{b}$ is uniformly drawn from $[0, 10]^n$. Each $\mathbf{A}_i$ is sampled from unit ball uniformly at random. Notice that the choice of $b_i \geq 0$ guarantees feasibility because the origin is always a feasible solution. We also add additional constraints $x_i \leq 500$ to make sure that the LP generated is bounded. When the algorithm makes an observation, a sample is drawn from Gaussian distribution with variance σ^2.

In Figure 1, each algorithm's number of samples (average of 50 instances) is plotted as function of different parameters. The number of samples used by Ellipsoid-UCB is proportional to n, σ^2 and ε^{-2}. However, it does not change much as m increases.[8] This will not be surprising if ellipsoid uses most of its samples on binding constraints, just as the lower bound does. This is shown in Table 1, where it can be seen that Ellipsoid-UCB re-

[7] 99.5 percent of the outputs turn out to satisfy relaxed feasibility and relaxed optimality.

[8] Indeed, the standard ellipsoid algorithm for linear programming requires a number of iterations that is bounded in terms of the number of variables regardless of the number of constraints.

quires much fewer samples of non-binding constraints than binding constraints.

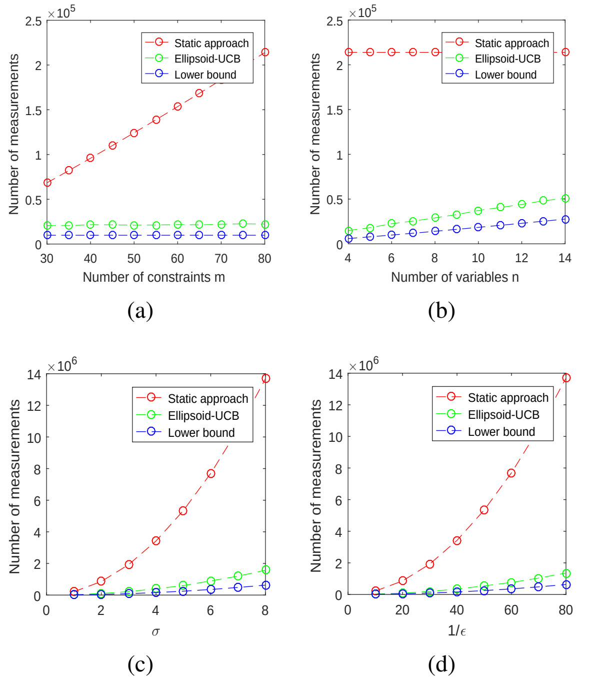

Figure 1: Number of samples as we vary m, n, σ and $1/\varepsilon$. Every data point is the mean of 50 randomly drawn problem instances. The baseline parameters are $m = 80$, $n = 6$, $\sigma = 1$, $\varepsilon_1 = \varepsilon_2 = 0.1$. In figure (d), $\varepsilon_1 = \varepsilon_2 = \varepsilon$.

	Binding	Non-binding
Static approach	2674	2674
Ellipsoid-UCB	3325	11.7
Lower bound	1476	0

Table 1: Average number of samples used per binding constraint and per non-binding constraint. Numbers are average from 100 trials. Here, $m = 80$, $n = 4$, $\sigma = 1$ $\varepsilon_1 = \varepsilon_2 = 0.1$.

Figure 2 addresses the variance in the number of samples drawn by Ellipsoid-UCB by plotting its empirical CDF over 500 random trials. The horizontal axis is the ratio of samples required by Ellipsoid-UCB to those of the lower bound. For comparison, we also mention R, the ratio between the performances of the static approach and the lower bound. These results suggest that the variance is quite moderate, particularly when the total number of samples needed grows.

7 DISCUSSION AND FUTURE WORK

One question is whether we can extend our results to situations when the constraint matrix $\mathbf{A}$ is unknown as well as $\mathbf{b}$. The goal is again to solve the problem with few

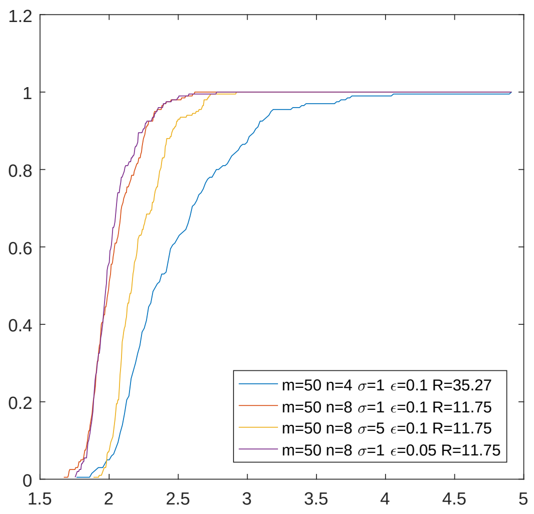

Figure 2: Empirical cumulative distribution function of Ellipsoid-UCB's number of samples, in units of the "lower bound", over 500 trials. Note that the lower bound varies when parameters change. $R = \frac{m \log(m)}{d \log(d)}$ is the ratio between the number of samples used by static approach and lower bound.

total observations. This extended model will allow us to study a wider range of problems. For example, the sample complexity problem in Reinforcement Learning studied by Kakade et al. [2003], Wang [2017], Chen and Wang [2016] is a special case of our AIALO problem with unknown $\mathbf{A}$ and $\mathbf{b}$.

A second extension to the model would allow algorithms access to varying qualities of samples for varying costs. For instance, perhaps some crowd workers can give very low-variance estimates for high costs, while some workers can give cheaper estimates, but have larger variance. In this case, some preliminary theoretical investigations suggest picking the worker that minimizes the product (price)(variance). A direction for future work is for the algorithm to select samples dynamically depending on the payment-variance tradeoffs currently available. A final interested direction is a more mechanism-design approach where the designer collects bids from the agents and selects a winner whose data is used to update the algorithm.

Acknowledgements

The authors are grateful to anonymous reviewers for their thorough reviews. This work is supported in part by National Science Foundation under grant CCF-1718549 and the Harvard Data Science Initiative.

References

Daniel P Heyman and Matthew J Sobel. *Stochastic models in operations research: stochastic optimization*, volume 2. Courier Corporation, 2003.

Michael J Neely. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3(1): 1–211, 2010.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

Dimitris Bertsimas, Dessislava Pachamanova, and Melvyn Sim. Robust linear optimization under general norms. *Operations Research Letters*, 32(6):510–516, 2004.

Darius Braziunas. Computational approaches to preference elicitation. Technical report, 2006.

Avrim Blum, Jeffrey Jackson, Tuomas Sandholm, and Martin Zinkevich. Preference elicitation and query learning. *Journal of Machine Learning Research*, 5 (Jun):649–667, 2004.

Sarah OConnell, Barry OSullivan, and Eugene C Freuder. Strategies for interactive constraint acquisition. In *Proceedings of the CP-2002 Workshop on User-Interaction in Constraint Satisfaction*, pages 62–76, 2002.

Christian Bessiere, Frédéric Koriche, Nadjib Lazaar, and Barry O'Sullivan. Constraint acquisition. *Artificial Intelligence*, 2015.

Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In *ICML*, 2006.

Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *COLT*, 2007.

Rui M Castro and Robert D Nowak. Minimax bounds for active learning. *Information Theory, IEEE Transactions on*, 54(5):2339–2353, 2008.

Sham Machandranath Kakade et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

Tor Lattimore and Marcus Hutter. Pac bounds for discounted mdps. In *International Conference on Algorithmic Learning Theory*, pages 320–334. Springer, 2012.

Mohammad Gheshlaghi Azar, Rémi Munos, and Bert Kappen. On the sample complexity of reinforcement learning with a generative model. *arXiv preprint arXiv:1206.6461*, 2012.

Mengdi Wang. Primal-dual π learning: Sample complexity and sublinear run time for ergodic markov decision problems. *CoRR*, abs/1710.06100, 2017. URL `http://arxiv.org/abs/1710.06100`.

Yichen Chen and Mengdi Wang. Stochastic primal-dual methods and sample complexity of reinforcement learning. *arXiv preprint arXiv:1612.02516*, 2016.

Eric Balkanski, Aviad Rubinstein, and Yaron Singer. The power of optimization from samples. In *Advances in Neural Information Processing Systems*, pages 4017–4025, 2016.

Shouyuan Chen, Tian Lin, Irwin King, Michael R Lyu, and Wei Chen. Combinatorial pure exploration of multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 379–387, 2014.

Victor Gabillon, Alessandro Lazaric, Mohammad Ghavamzadeh, Ronald Ortner, and Peter Bartlett. Improved learning complexity in combinatorial pure exploration bandits. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 1004–1012, 2016.

Wei Chen, Wei Hu, Fu Li, Jian Li, Yu Liu, and Pinyan Lu. Combinatorial multi-armed bandit with general reward functions. In *Advances in Neural Information Processing Systems*, pages 1659–1667, 2016a.

Lijie Chen, Anupam Gupta, and Jian Li. Pure exploration of multi-armed bandit under matroid constraints. In *Conference on Learning Theory*, pages 647–669, 2016b.

Lijie Chen, Anupam Gupta, Jian Li, Mingda Qiao, and Ruosong Wang. Nearly optimal sampling algorithms for combinatorial pure exploration. In *Proceedings of the 2017 Conference on Learning Theory*, pages 482–534, 2017.

Kevin Jamieson and Robert Nowak. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, pages 1–6. IEEE, 2014.

Kevin G Jamieson, Matthew Malloy, Robert D Nowak, and Sébastien Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *COLT*, volume 35, pages 423–439, 2014.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, 17(1):1–42, 2016.

John Duchi. Derivations for linear algebra and optimization.