Active Information Acquisition for Linear Optimization

Shuran Zheng∗1, Bo Waggoner†2, Yang Liu†1, and Yiling Chen§1

1Harvard University
2University of Pennsylvania

September 29, 2017

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), \( \max \, c^T x \) s.t. \( Ax \leq b, x \geq 0 \), 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 \( c \) 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. (2) When the parameters \( b \) 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.

1 Introduction

In many real-world settings, the goal is to solve an optimization problem, but some parameters of the problem are initially unknown. For example, consider a delivery company that wishes to plan driver routes, but does not initially know average congestion or travel time of various links of the network.

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

∗sherry.zhengshuran@gmail.com
†bwag@seas.upenn.edu
‡yangli.seas.harvard.edu
§yiling@seas.harvard.edu
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 optimization problems. Specifically, we consider linear programs in the form

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

(1)

with $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{c} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^m$. For most of the paper, 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 $\eta$. The algorithm must eventually output a solution $\mathbf{x}$ such that, with probability $1 - \delta$, $\mathbf{A} \mathbf{x} \leq \mathbf{b} + \epsilon_1 \mathbf{1}$ and $\mathbf{c}^T \mathbf{x} \geq \mathbf{c}^T \mathbf{x}^* - \epsilon_2$, where $\mathbf{x}^*$ is the optimal solution. The goal is for the algorithm to achieve this while using as few total samples as possible. (In Appendix A, we give some motivating examples of practical problems that fit this model.)

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. 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 gather information on the differing objective values of extreme points and to focus on binding constraints. Algorithms developed in this paper leverage these properties of linear programs to decide how much information to acquire for each unknown parameter.

The setting considered in this paper, active information acquisition for optimization, is related at a high level to a large number of lines of work (in theoretical computer science (Balkanski et al., 2016), machine learning (Balcan et al., 2006, 2007; Castro and Nowak, 2008), and artificial intelligence (Brazinuas, 2006; Blum et al., 2004)) that deal with optimization and uncertainty. However, most of them solve a different conceptual problem, failing to model either the optimization aspect or the active aspect. At a technical level, some works (Chen et al., 2014; Gabillon et al., 2016; Chen et al., 2016a,b, 2017) are relatively close to our model, but do not provide useful solutions to the model of linear AIAO studied here. We discuss the related work in details in Appendix B.

1.1 Approaches and Results

Two settings and our approaches. The paper investigates two settings: unknown $\mathbf{c}$ but known $\mathbf{b}$, and unknown $\mathbf{b}$ but known $\mathbf{c}$. We always suppose $\mathbf{A}$ is known (see the discussion in Section 6 on how our algorithms may extend to unknown-$\mathbf{A}$) and assume that the linear program has an optimal solution.

It might initially appear that these cases are “equivalent” via duality theory, so we briefly digress to explain why this is not the case (indeed, we argue that the two cases are quite different). Given a primal linear program of the form (1), the dual program is given by $\min_{\mathbf{y}} \mathbf{b}^T \mathbf{y} \text{ s.t. } \mathbf{A}^T \mathbf{y} \geq \mathbf{c}, \mathbf{y} \geq 0$, 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, which allow one to use the optimal dual solution $\mathbf{y}^*$ to determine which of the primal constraints are binding at the optimal primal solution $\mathbf{x}^*$. 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 $x^*$ when some of the constraint or objective parameters are unknown.\textsuperscript{1} We hence take two different approaches toward our two settings.

**Unknown-c case.** In the unknown-objective setting, we take an information-theoretic perspective. Because the feasible region is known exactly, our (inefficient) algorithm focuses only on the set of extreme points of the feasible region as one of them must be an optimal solution to the LP. For each of the extreme point, there are a set of possible values for $c$ such that if $c$ 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 $c$. In fact, the algorithm therefore produces the exact optimal solution $x^*$ to the LP with high probability.

For this setting, we define an information-theoretic measure, $\text{Low}(I)$ for an instance $I$. 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.

**Theorem 1.1** (Informal). Given an instance $I$, let $K(I)$ be the number of extreme points of the LP’s feasibility region. For the unknown-c problem with failure probability $\delta$:

(i) On any instance $I$, at least $\Omega(\text{Low}(I) \ln \delta^{-1})$ samples are required in expectation.

(ii) On any instance $I$, Algorithm 1 draws at most $O\left(\text{Low}(I) \ln \frac{K(I)}{\delta}\right)$ samples with high probability.

(iii) There are families of instances $\{I_t\}_{t=1}^\infty$ on which any algorithm must draw $\Omega\left(\text{Low}(I_t) \ln \frac{K(I_t)}{\delta}\right)$ samples on instance $I_t$.

**Unknown-b case.** In contrast, in the unknown-b setting, the uncertainty is over the feasible region. An algorithm cannot initially identify extreme points but, with known $c$, it does know the direction that improves the objective function value. We hence take an “interior point” approach. Our algorithm, which we call UCB-Ellipsoid, starts with a large relaxed feasible region and an interior point in that region and draws samples to confidently “cut” the region in iterations. This leverages the key idea of the ellipsoid algorithm for solving linear programs: as we advance in the direction of the optimal solution, we can ignore irrelevant constraints and focus on either those that help cut the region or else (approximately) bind at an (approximately) optimal solution. Meanwhile, it also leverages the idea of “confidence bounds” from multi-armed bandits algorithms such as UCB (Upper Confidence Bound). For each unknown parameter, we maintain upper and lower confidence bounds which shrink as we gather more samples from the parameter. Each iteration of the ellipsoid algorithm, we will adaptively sample from various constraints until we are confident that some constraint is violated or that no constraint is. To do so, we sample the constraint that is most violated in an “optimistic” fashion, using the extreme value of the confidence bound.

We show the following bound on the sample complexity of UCB-Ellipsoid, along with a natural lower bound:

**Theorem 1.2** (Informal). For the unknown-b problem on $n$ variables and $m$ constraints (for simplicity take $m \geq n$), with approximation tolerance $\epsilon$ and failure probability $\delta$:

(i) The sample complexity of any algorithm in the worst case is $\Omega\left(n \ln \frac{1}{\delta} \cdot \epsilon^{-2}\right)$.

(ii) Algorithm 2 draws at most $O\left(\frac{m}{\Delta(I)} \log \frac{m}{\delta} \cdot \epsilon^{-2}\right)$ samples with high probability, where $\Delta(I) \geq 1$ is a problem-specific measure of difficulty.

\textsuperscript{1}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.
To get an idea of the sample complexity of this method in practice, we implemented it on a variety of synthetic linear programming problems. In our simulations, 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.

2 Model and Preliminaries

2.1 The AIALO problem

We now formally define an instance \( I \) 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 AIALO, for more general optimization problems, but we leave this for future work.

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

\[
\max_{x} c^T x \quad \text{s.t.} \quad Ax \leq b, \; x \geq 0.
\]  

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 \( \sigma^2 \)-subgaussian distribution with mean equal to the value of the parameter.

Given \( I \), we define the following sets of approximately-feasible, approximately-optimal solutions.

Definition 2.1. Given an instance \( I \), let \( x^* \) be an optimal solution to the LP. Define \( \text{OPT}(I; \varepsilon_1, \varepsilon_2) \) to be the set of solutions \( x \) satisfying \( c^T x \geq c^T x^* - \varepsilon_1 \) and \( Ax \leq b + \varepsilon_2 1 \). We use \( \text{OPT}(I) \) as shorthand for \( \text{OPT}(I; 0, 0) \).

2.2 Algorithm specification

An algorithm for the AIALO problem, run on an instance \( I \), functions as follows. The algorithm is given as input \( n \) (number of variables), \( m \) (number of constraints), and \( \sigma^2 \) (subgaussian parameter). It is also given the second component of the instance \( I \), 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 \( \varepsilon_1 \), a feasibility parameter \( \varepsilon_2 \), and a failure probability parameter \( \delta \).

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 \( I \)). At some point, the algorithm stops and outputs a solution \( x \in \mathbb{R}^n \).

Definition 2.2 ((\( \delta, \varepsilon_1, \varepsilon_2 \))-correct algorithm). An algorithm \( A \) is \((\delta, \varepsilon_1, \varepsilon_2)\)-correct if for any instance \( I \) and inputs \((\delta, \varepsilon_1, \varepsilon_2)\), with probability at least \( 1 - \delta \), \( A \) outputs a solution \( x \in \text{OPT}(I; \varepsilon_1, \varepsilon_2) \). In the case \( \varepsilon_1 = \varepsilon_2 = 0 \), we say \( A \) is \( \delta \)-correct.

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

\[ \text{Distribution } D \text{ with mean } \mu \text{ is } \sigma^2 \text{-subgaussian if, for } X \sim D, \text{ we have } \mathbb{E}[e^{t(X-\mu)}] \leq e^{\sigma^2 t^2/2} \text{ for all } t. \text{ The family of sub-Gaussian distributions with parameter } \sigma \text{ encompasses all distributions that are supported on } [0, \sigma] \text{ as well as many unbounded distributions such as Gaussian distributions with variance } \sigma^2. \]
For the purposes of this paper, the sample complexity of an algorithm on \( I \) is the maximum, over all possible random outcomes, of the number of samples the algorithm draws when run on instance \( I \). The sample complexity with respect to \( n, m, \text{ and } \sigma^2 \) refers to the maximum sample complexity of the algorithm over all instances with these parameters (recalling that we only consider instances for which an optimal solution exists and has norm at most \( R \)).

### 2.3 Cases in this paper

**Unknown objective.** We will first consider the unknown-c case. Here, every parameter of the objective \( c \) 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 \( Ax \leq b \) and \( x \geq 0 \), 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.

Information-theoretically, any algorithm for this case begins with the set of feasible points (a polytope), at least one of which is exactly optimal and hence in \( OPT(I; \varepsilon_1, 0) \). Information obtained over time implicitly eliminates far-from-optimal points until those that remain are \( \varepsilon_1 \)-optimal with high confidence. For a more combinatorial slant, note that the same argument is true just for the corners of the polytope, the extreme points, of which there is a finite (though exponential) number, and at least one of which is exactly optimal. In fact, if there is a unique optimal solution, then by iteratively eliminating extreme points it is possible to achieve \((\delta, 0, 0)\)-correctness for the unknown-c case. We will focus on this goal. (In the case where there is not a unique optimal solution, a tweak to our algorithm can achieve \((\delta, \varepsilon_1, 0)\)-correctness.)

**Unknown constraint.** Second, we will consider the unknown-b case where every parameter of the constraint vector \( b \) is initially unknown, and all other parameters are initially known. Geometrically, the algorithm is given an objective “direction” \( c \) and a set of constraint “orientations” \( A \), 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. This represents a qualitative difference from the unknown-c case.\(^4\)

**Other cases.** We briefly discuss in Section 6 the case of both unknown constraint and (partially) unknown \( A \). Variants on the observational model are also interesting in this case. For instance, perhaps instead of being able to sample a particular parameter \( A_{ij} \) or \( b_i \), the algorithm can only propose an \( x \) and observe a random sample centered on the quantity \( A_i x - b_i \). While we briefly discuss how Ellipsoid-UCB can extend to solve such cases, they go beyond the scope of this paper and we leave sample complexity bounds and/or empirical investigation to future work.

### 3 The Unknown Objective Function Case

In this section, consider instances of AIALO where \( A \) and \( b \) are initially known, but \( c \) is initially unknown. We focus on the case where there is a unique optimal solution \( x^* \), and consider the problem of finding an

\(^4\)In particular, it may be possible to hope for some sort of duality-based reduction from the unknown-b case to the unknown-c case, because the latter can be solved to exact feasibility and optimality; but the reverse seems less likely. This is an exciting direction for future work, especially in terms of duality relationships between approximate solutions.
exact optimal solution with confidence \( \delta \) (i.e., a \( \delta \)-correct algorithm). For this section, which focuses more on information-theoretic structure than practical algorithms, we also make the simplifying assumption that each parameter’s distribution is a Gaussian of variance 1 (in particular is 1-subgaussian). Our results are relatively straightforward to extend to the more general setting.

We first introduce a function \( \text{Low}(I) \) that characterizes the sample complexity required for an LP instance \( I \). The function \( \text{Low}(I) \) is defined by the solution of a convex program. We then give an instance-wise lower bound in terms of the \( \text{Low}(I) \) function and the failure probability parameter \( \delta \). 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 \( \Delta \) is the gap between the objective function value of the optimal extreme point \( (x^*) \) and the second-best.

### 3.1 Lower Bounds

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

**Definition 3.1** (\( \text{Low}(I) \)). For any instance \( I \) of AIALO (or more generally, for any linear program), we define \( \text{Low}(I) \in \mathbb{R} \) to be the optimal solution to the following convex program.

\[
\min \sum_{i=1}^{n} \tau_i \tag{3}
\]

s.t. \[
\sum_{i=1}^{n} \frac{(s^{(k)}_i - x^*_i)^2}{\tau_i} \leq \left( c^T (x^* - s^{(k)}) \right)^2, \forall k
\]

\[\tau_i \geq 0, \forall i\]

Here \( x^* \) is the optimal solution to the LP in \( I \) and \( s^{(1)}, \ldots, s^{(k)} \) are the extreme points of the feasible region \( \{ x : Ax \leq b, x \geq 0 \} \).

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

\[
\hat{c}^T (x^* - s^{(k)}) > 0 \iff c^T (x^* - s^{(k)}) > 0.
\]

This will hold by a standard concentration bound (Lemma C.4) if enough samples of each parameter are drawn; in particular, “enough” is given by the \( k \)-th constraint in (3).

**Theorem 3.1** (Instance lower bound). Let \( I \) be an instance of AIALO in the unknown-\( c \) case. For \( 0 < \delta < 0.1 \), any \( \delta \)-correct algorithm \( A \) must draw

\[
\Omega(\text{Low}(I) \ln \delta^{-1})
\]

samples in expectation on \( I \).

**Proof Sketch:** Given \( I \) with optimal solution \( x^* \), we construct an instance \( I' \) which differs only in the unknown parameters \( c \), modified just enough so that the extreme point \( s^{(k)} \) becomes the optimal solution to \( I \). We then appeal to Lemma C.1, which relies on a multi-armed-bandits-style result of Kaufmann et al. (2016) showing that, if two algorithms are given samples from distributions that are similar enough, then their outputs are also similar unless enough samples are drawn from the various distributions. This allows us
to conclude that, since the δ-correct algorithm must be able to distinguish \( I \) from \( I \), its vector of expected number of samples per parameter must satisfy the \( k \)-th constraint in the definition of \( \text{Low}(I) \). Repeating this argument for each \( s^{(k)} \) shows that the expected vector of samples must be feasible for the convex program in the definition of \( \text{Low}(I) \), so the expected number of samples must exceed \( \Omega(\text{Low}(I) \ln \delta^{-1}) \).

We believe that it is unlikely for an algorithm to match the instance-wise lower bound without knowing the value of \( c \) and \( x^* \) in the definition of \( \text{Low}(I) \). 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 \( Ax \leq b, x \geq 0 \) 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 3.2** (Worst-case lower bound for unknown \( c \)). 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, \( I_1, I_2, \ldots \), such that \( \mathcal{A} \) takes

\[
\Omega \left( \text{Low}(I_k)(\ln |S_k^{(1)}| + \ln \delta^{-1}) \right)
\]

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

We postpone the proof of the worst-case lower bound to Appendix C.3.

### 3.2 A Successive Elimination Algorithm

Before the description of the algorithm, we first define a function \( \text{LowAll}(S, \varepsilon, \delta) \) 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 \( \varepsilon \) with probability \( 1 - \delta \). Define \( \text{LowAll}(S, \varepsilon, \delta) \) to be the optimal solution of the following convex program,

\[
\begin{align*}
\min \quad & \sum_{i=1}^{n} \tau_i \\
s.t. \quad & \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\tau_i} \leq \frac{\varepsilon^2}{2 \ln(2/\delta)}, \forall x, y \in S \\
& \tau_i \geq 0, \forall i.
\end{align*}
\]

Our algorithm starts with a set \( S^{(1)} \) that contains all extreme points of the feasible region \( \{ x : Ax \leq b, x \geq 0 \} \), 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 1 A successive elimination algorithm

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

The algorithm has the following sample complexity bound.

**Theorem 3.3** (Sample complexity of Algorithm 1). For the AIALO with unknown-\(c\) problem, Algorithm 1 is \(\delta\)-correct and, on instance \(I\), with probability \(1 - \delta\) draws at most the following number of samples:

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

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

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

**Proof Sketch:** We define a good event \(\mathcal{E}\) to be the event that for each iteration \(r\) and each pair of extreme points in \(S^{(r)}\), the difference between them is estimated within error \(\varepsilon^{(r)}/\lambda\). We first show that \(\mathcal{E}\) holds with probability at least \(1 - \delta\), giving correctness. Conditioning on \(\mathcal{E}\), we can prove that the algorithm will not delete the optimal solution and will terminate before \([\log(\Delta^{-1})] + 1\) iterations. Then, we bound the number of samples used in each iteration separately. We show that in each iteration \(r\), the optimal solution of \(\text{Low}(I)\) times \(\alpha^{(r)} = 32\lambda^2 \ln(2/\delta^{(r)})\) is a feasible solution of the convex program that defines \(\text{LowAll}(S^{(r)}, \varepsilon^{(r)}/\lambda, \delta^{(r)})\). Therefore the number of samples used in iteration \(r\) is no more than \(\alpha^{(r)} \text{Low}(I)\). Together with the upper bound of iteration number, this completes the proof. \(\square\)

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 1 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.
4 The Unknown Constraint Case

In this section, we study the AIALO problem with unknown-\(b\). Since we can never know the exact feasible region in this case, we choose non-zero \(\varepsilon_1\) and \(\varepsilon_2\) and try to find a solution of \(\mathcal{I}\) in \(OPT(\mathcal{I}; \varepsilon_1, \varepsilon_2)\) using as few samples as possible.

The section begins with a lower bound in terms of the binding constraints. Then we introduce UCB-Ellipsoid algorithm and give an upper bound of its sample complexity. Finally we compare our algorithm with the lower bound, as well as a naive approach using simulations.

**Theorem 4.1** (Lower bound for unknown \(b\)). Suppose we have a \((\delta, \varepsilon_1, \varepsilon_2)\)-correct algorithm \(A\) 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-\(b\) with \(n\) variables with objective function \(|c|_\infty = 1\) such that \(A\) 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 D.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.** Before giving our algorithm, we describe the standard ellipsoid algorithm for linear programming. The high-level idea is to maintain a boundary, using an “ellipsoid” shape, around the space where the optimal solution must lie; then repeatedly cut this space using either the constraints or the objective. The algorithm begins with an ellipsoid \(\mathcal{E}^{(0)}\) known to contain the optimal solution. Formally, an ellipsoid is the set of points \(\{x : x^T P^{-1} x \leq 1\}\) specified by a positive definite matrix \(P\).

The algorithm sets \(x^{(0)}\) to be the center of this ellipsoid (which can be computed efficiently). Then, it checks two cases: (1) \(x^{(0)}\) 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 \(A_i x^{(0)} \leq b_i\). If (1), the algorithm considers the halfspace defined by the “constraint” \(c^T x \geq c^T x^{(0)}\), 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, \(b\) is initially unknown. When the ellipsoid algorithm needs to find a violated constraint (or certify that none is violated), how should we sample the constraints \(b_i\)?

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 \(\mu_1, \ldots, \mu_m\) and variances \(\sigma_1^2, \ldots, \sigma_m^2\), and we wish to find the largest \(\mu_i\). After drawing a few samples from each distribution, we obtain estimates \(\hat{\mu}_i\) 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.

In our case, at each stage of the ellipsoid algorithm, we wish to find the largest \(A_i x - b_i\) or ensure that all are nonpositive. 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 2. At each round \( k = 1, 2, \ldots \), we choose the center point \( x^{(k)} \) of the current ellipsoid \( E^{(k)} \) and call the subroutine Algorithm 3 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

\[
\hat{b}_{i,T_i(t)} = \frac{T_i(t)}{\sum_{s=1}^{T_i(t)} X_{i,s}}
\]

(6)

to be the empirical mean of \( b_i \) up to “time” \( t \).

**Algorithm 2** Modified ellipsoid algorithm

- Let \( E^{(0)} \) 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 \( i \).

\[\textbf{while } \text{ stopping criterion is not met}^5 \textbf{ do} \]

- Let \( x^{(k)} \) be the center of \( E^{(k)} \)
- Call UCB method to get constraint \( i \) or “feasible”
- if \( x^{(k)} \) is feasible then
  - Let \( y \leftarrow -c \)
  - \( y \leftarrow A^T_i \)
- else
- end if

- Let \( E^{(k+1)} \) be the minimal ellipsoid that contains \( E^{(k)} \cap \{ p : y^T p \leq y^T x^{(k)} \} \)
- Let \( k \leftarrow k + 1 \)

end while

Output \( x \) 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 \( V_i(k) = A_i x^{(k)} - b_i \) to be the amount by which the \( i \)-th constraint is violated by \( x^{(k)} \), and \( V^*(k) = \max_i V_i(k) \). Define \( gap_{i,\epsilon}(k) = \max \{|V_i(k)|, V^*(k) - V_i(k), \epsilon \} \) and \( \Delta_{i,\epsilon} = \min_k \text{gap}_{i,\epsilon}(k) \).

**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:

\[
O \left( \sum_{i=1}^{m} \frac{\sigma_i^2}{\Delta_i^2 \epsilon_2 / 2} \log \frac{m}{\delta} + \sum_{i=1}^{m} \frac{\sigma_i^2}{\Delta_i^2 \epsilon_2 / 2} \log \log \left( \frac{\sigma_i^2}{\Delta_i^2 \epsilon_2 / 2} \right) \right).
\]

Specifically, the number of samples used for \( b_i \) is at most \( \frac{\sigma_i^2}{\Delta_i^2 \epsilon_2 / 2} \left( \log(m / \delta) + \log \log(\sigma_i^2 / \Delta_i^2 \epsilon_2 / 2) \right) \).

\( ^5 \)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 \( \sqrt{c^T P^{-1} c} \leq \min \{ \epsilon_1, \epsilon_2 \} \), where \( P \) is the matrix corresponding to ellipsoid \( E^{(k)} \) as discussed above.
Algorithm 3 UCB-method

Input $x^{(k)}$
Output either index $j$ of a violated constraint, or “feasible”.
Set $\delta' = (\frac{\delta}{20m})^{2/3}$

loop
1. Let $j$ be the constraint with the largest index,
   
   $$j = \arg \max_{i} A_i x^{(k)} - \hat{b}_{j,T_j(t)} + U_i(T_i(t)),$$

   where $U_i(s) = 3\sqrt{\frac{2\sigma_i^2 \log(\log(3s/2)/\delta')}}{s}$ and $\hat{b}_{j,T_j(t)}$ as in Definition 4.1.

2. If $A_j x^{(k)} - \hat{b}_{j,T_j(t)} > 0$ return $j$.

3. If $A_j x^{(k)} - \hat{b}_{j,T_j(t)} < 0$ return “feasible”.

4. If $U_j(T_i(t)) < \varepsilon_2^2/2$ 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: Our analysis is inspired by the techniques used in Jamieson et al. (2014). Define event $\mathcal{A}$ to be the event that $|\hat{b}_{i,s} - b_i| \leq U_i(s)$ for all $s \geq 0$ and $i \in [m]$. According to Lemma 3 in Jamieson et al. (2014), $\mathcal{A}$ holds with probability at least $1 - \delta$. We prove the correctness and the sample complexity of the algorithm conditioning on that $\mathcal{A}$ holds.

Correctness: If UCB-method always gives a correct answer, the ellipsoid algorithm will be able to find an $\varepsilon_1$-suboptimal solution. So we only need to prove the correctness of the UCB-method. 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 $\varepsilon_2$ (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 $x^{(k)}$, 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 $\text{gap}_{i,\varepsilon_2/2}(k)$. Since $U_i$ is a decreasing function, this immediately gives us an upper bound of $T_i(t)$, the number of samples used on constraint $i$ before sample number $t$. Thus for each ellipsoid iteration $k$, we have 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:

- $|V_i(k)|$ is small, meaning that the ellipsoid center $x^{(k)}$ is very close to binding constraint $i$. 


There is no other constraint that is significantly violated, meaning that $i$ is very close to the most-violated constraint for $x^{(k)}$ if any.

Because this is unlikely to happen for most constraints, we expect $\Delta_{i,\varepsilon/2}$ 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 $\Delta_{i,\varepsilon/2}$ 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 $\Delta_{i,\varepsilon/2} \geq \varepsilon/2$. This implies that our bound is always better than $O\left(\frac{m \log(m/\delta)}{\varepsilon^2}\right)$, 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 $\frac{4m \log(m/\delta)}{\varepsilon^2}$ 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 Experiments

In this section, we investigate the empirical number of samples used by Ellipsoid-UCB algorithm for the unknown-$b$ case of AIALO. We fix $\delta = 0.1$ and focus on the impact of the other parameters\(^6\), 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 $4\sigma^2 \log(m/\delta)/\varepsilon^2$ 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 $4\sigma^2 \log(d/\delta)/\varepsilon^2$ 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. $c$ is sampled from $[-10, 10]^n$ uniformly at random. $b$ is uniformly drawn from $[0, 10]^n$. Each $A_i$ 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 $\sigma^2$.

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 $n$, $\sigma^2$ and $\varepsilon^{-2}$. However, it does not change much as $m$ increases.\(^7\) This will not be surprising if ellipsoid uses most of its

---

\(^6\)99.5 percent of the outputs turn out to satisfy relaxed feasibility and relaxed optimality.

\(^7\)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.
samples on binding constraints, just as the lower bound does. This is shown in Table 1, where it can be seen that Ellipsoid-UCB requires much fewer samples of non-binding constraints than binding constraints.

Figure 1: Number of samples as we vary $m$, $n$, $\sigma$ and $1/\epsilon$. Every data point is the mean of 50 randomly drawn problem instances. The baseline parameters are $m = 80$, $n = 6$, $\sigma = 1$, $\epsilon_1 = \epsilon_2 = 0.1$. In figure (d), $\epsilon_1 = \epsilon_2 = \epsilon$.

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.
### Table 1: Average number of samples used per binding constraint and per non-binding constraint. Numbers are average from 100 trials. Here, $m = 80$, $n = 4$, $\sigma = 1$, $\epsilon_1 = \epsilon_2 = 0.1$.

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

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.

## 6 Discussion and Future Work

One question is whether our approach can extend when the constraint matrix $A$ is unknown as well as $b$. The goal is again to solve the problem with few total observations. One model would be that the algorithm can request samples from any $A_{ij}$ or $b_i$ at any time. In this case, our algorithm can extend (although we have not yet proven theoretical bounds), by maintaining confidence bounds around each $A_{ij}$ and $b_i$. Once enough certainty is obtained that a given constraint is violated, many samples are drawn for $b_i$ and $A_{ij} \quad (\forall j)$, allowing an accurate update to the ellipsoid algorithm.

Another model would only allow samples from the violation of a constraint, i.e. the algorithm provides an $x$ and chooses a constraint $i$ and learns a noisy estimate of $b_i - A_i x$. Here we can still determine which constraint is violated (if any), but more creativity is required to determine the entries of $A_i$ so as to continue the ellipsoid algorithm. We have a proposal involving requestion $b_i - A_i x$ for a large set of different $x$, giving noisy linear equations which can be solved to learn $A_i$ accurately.

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.
References

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.

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. CoRR, abs/1706.01081, 2017. URL http://arxiv.org/abs/1706.01081.

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.

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.
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.

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.

John Duchi. Derivations for linear algebra and optimization.
A Motivating Examples

Our settings can be used to model many real-world optimization problems. In this section, we expand on some example real-world problems that fall into our framework:

(i) A company who wants to decide a production plan to maximize profit faces a linear program. But when entering a new market, the company may not initially know the average unit price $c_i$ for their different products in this market. Sampling $c_i$ corresponds to surveying a consumer on his willingness to buy product $i$.

(ii) A delivery company who wants to plan driver routes may not know the average traffic/congestion of road segments. Each segment $e$ (an edge in the graph) has a true average travel time $b_e$, but any given day it is $b_e + \text{noise}$. One can formulate shortest paths as an LP where $b$ is the vector of edge lengths. Sampling $b_e$ corresponds to sending an observer to the road segment $e$ to observe how long it takes to traverse on a given day.

(iii) A ride sharing company (e.g. Uber) wants to decide a set of prices for rides request but it may not know customers’ likelihood of accepting the prices $c_i$. Sampling $c_i$ in this setting corresponds to posting different prices to collect information.

(iv) For the purpose of recommending the best route in real time, a navigation App, e.g., Waze\(^8\), may want to collect traffic information or route information from distributed driver via their App.

B Related Work

Related work: Theoretical computer science and machine learning. 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.

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 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-$c$ problem can be fitted into the setting of General-Sampling. Nevertheless, our algorithm for unknown-$c$

\(^8\)www.waze.com
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-b case.

**Related work: 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 (Brazuinias, 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-e setting. But often the existing literatures do not analyze an active information acquisition setting; neither do they analyze the sample complexity. Relevant to our unknown-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 constraint. The key challenge is to choose how many repeated samples of each parameter to draw so as to gain confidence in using or discarding its constraint. This aspect of the problem is not to our knowledge present in preference elicitation or model/constraint acquisition. This also differs from the literatures on 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).

**C Proofs for the Unknown Objective Function Case**

**C.1 Change of Distribution Lemma**

A key element to derive the lower bounds is the Change of Distribution lemma, which was first formulated in Kaufmann et al. (2016). The lemma provides a general relation between the expected number of draws and Kullback-Leibler divergences of the arms distributions. The core elements of the model that the lemma can be applied to are almost the same as the classical bandit model. We will state it here and explain the applicability of our setting. In the bandit model, there are \( n \) arms, with each of them being characterized by an unknown distribution \( \nu_i, i = 1, 2, ..., n \). The bandit model consists of a sequential strategy that selects a subset of arms of size \( K \). Upon selection, each arm reveals a reward generated from its corresponding distribution. The rewards for each arm form an i.i.d. sequence. The selection strategy/algorithm invokes a stopping time \( T \) when the algorithm will terminate and output a solution.

If we consider a LP instance \( \mathcal{I} \) with unknown parameters \( d \) as a bandit model, an unknown parameter \( d_i \) will correspond to an arm \( a \) in the Lemma C.2, and thus \( \nu_a \) is just the Gaussian distribution with mean \( d_i \) and variance \( 1 \) (both being unknown). Each step, we select \( K = 1 \) coefficient to sample with. Let \( T \) be a sufficiently large number such that algorithm \( A \) terminates before time \( T \) almost surely. Then we will be able to safely apply the Change of Distribution lemma to our setting. The lemma can be stated as follows in our setting of the problem.

**Lemma C.1.** Let \( A \) be a \((\delta, \epsilon_1, \epsilon_2)\)-correct algorithm with \( \delta \in (0, 0.1) \). Let \( \mathcal{I}, \mathcal{I}' \) be two LP instances that are equal on all known parameters, and let \( d, d' \) be their respective vectors of unknown parameters. Suppose each instance has samples distributed Gaussian with variance 1. Suppose \( \text{OPT}(\mathcal{I}; \epsilon_1, \epsilon_2) \) and \( \text{OPT}(\mathcal{I}'; \epsilon_1, \epsilon_2) \) are disjoint. Then letting \( \tau_i \) be the number of samples \( A \) draws for parameter \( d_i \) on input \( \mathcal{I} \), we have

\[
\mathbb{E} \sum_i \tau_i(d_i - d'_i)^2 \geq 0.8 \ln \frac{1}{\delta}.
\]
**Proof.** We use a result on bandit algorithms by Kaufmann et al. (2016), which is restated as follows.

**Lemma C.2** (Kaufmann et al. (2016)). Let \( \nu \) and \( \nu' \) be two bandit models with \( n \) arms such that for all arm \( a \), the distribution \( \nu_a \) and \( \nu'_a \) are mutually absolutely continuous. For any almost-surely finite stopping time \( T \) with respect to \( (\mathcal{F}_t) \),

\[
\sum_{i=1}^{n} \mathbb{E}_\nu[N_a(T)]KL(\nu_a, \nu'_a) \geq \sup_{\mathcal{E} \in \mathcal{F}_T} d(\Pr(\mathcal{E}), \Pr'_{\mathcal{E}}(\mathcal{E})),
\]

where \( d(x, y) = x \ln(x/y) + (1 - x) \ln((1 - x)/(1 - y)) \) is the binary relative entropy function, \( N_a(T) \) is the number of samples drawn on arm \( a \) before time \( T \) and \( KL(\nu_a, \nu'_a) \) is the KL-divergence between distribution \( \nu_a \) and \( \nu'_a \).

Let \( I \) and \( I' \) be the two bandit models in Lemma C.2. Applying above lemma we have

\[
\sum_{i=1}^{n} \mathbb{E}_{A,I}[\tau_i]KL(\mathcal{N}(d_i, 1), \mathcal{N}(d'_i, 1)) \geq d(\Pr(\mathcal{E}), \Pr'_{A,I}(\mathcal{E})), \text{ for all } \mathcal{E} \in \mathcal{F}_T,
\]

where \( \mathcal{N}(\mu, \sigma) \) is the Gaussian distribution with mean \( \mu \) and variance \( \sigma \), \( \Pr_{A,I}[\mathcal{E}] \) is the probability of event \( \mathcal{E} \) when algorithm \( A \) is given input \( I \), and \( \mathbb{E}_{A,I}[X] \) is the expected value of random variable \( X \) when algorithm \( A \) is given input \( I \). According to the result in Duchi, the KL-divergence for two Gaussian distribution with mean \( \mu_1, \mu_2 \) and variance \( \sigma_1, \sigma_2 \) is equal to

\[
\log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2}.
\]

Thus we have \( KL(\mathcal{N}(d_i, 1), \mathcal{N}(d'_i, 1)) = \frac{1}{2}(d_i - d'_i)^2 \). We further define event \( \mathcal{E} \) to be the event that algorithm \( A \) finally outputs a solution in set \( OPT(I; \varepsilon_1, \varepsilon_2) \), then since \( A \) is \((\delta, \varepsilon_1, \varepsilon_2)\)-correct and \( OPT(I; \varepsilon_1, \varepsilon_2) \) is disjoint from \( OPT(I'; \varepsilon_1, \varepsilon_2) \), we have \( \Pr_{A,I}[\mathcal{E}] \geq 1 - \delta \) and \( \Pr_{A,I'}[\mathcal{E}] \leq \delta \). Therefore

\[
\sum_{i=1}^{n} \mathbb{E}_{A,I}[\tau_i] \frac{1}{2}(d_i - d'_i)^2 \geq d(1 - \delta, \delta) \geq 0.4 \ln \delta^{-1}.
\]

The last step uses the fact that for all \( 0 < \delta < 0.1 \),

\[
d(1 - \delta, \delta) = (1 - 2\delta) \ln \frac{1 - \delta}{\delta} \geq 0.8 \ln \frac{1}{\sqrt{\delta}} = 0.4 \ln \delta^{-1}.
\]

\[\square\]

**C.2 Proof for Theorem 3.1**

We restate the instance-wise lower bound for unknown objective function LP problems.

**Theorem 3.1** (Instance lower bound). Let \( I \) be an instance of AIALO in the unknown-\( c \) case. For \( 0 < \delta < 0.1 \), any \( \delta \)-correct algorithm \( A \) must draw

\[\Omega(Low(I) \ln \delta^{-1})\]

samples in expectation on \( I \).
Let $\mathcal{I}$ be a LP instance $\max_{\{x: Ax \leq b\}} c^T x$, and $A$ be a $\delta$-correct algorithm, where $0 < \delta < 0.1$. Define $t_i$ to be the expected number of samples that algorithm will draw for $c_i$ when the input is $\mathcal{I}$.

We only need to show that $5t_i \ln(1/\delta)$ is a feasible solution of the convex program (3) that computes $\text{Low}(\mathcal{I})$.

Consider a constraint in (3)

$$\sum_{i=1}^n \frac{(s_i^{(k)} - x_i^*)^2}{t_i} \leq \left( c^T (x^* - s^{(k)}) \right)^2,$$

where $x^*$ is the optimal solution of $\mathcal{I}$ and $s^{(k)}$ is a corner point of the feasible region of $\mathcal{I}$. To prove that $5t_i \ln(1/\delta)$ satisfies this constraint, we will construct a new LP instance $\mathcal{I}_{\Delta}$ by adding $\Delta$ to the objective function $c$, such that $s^{(k)}$ becomes a better solution than $x^*$. We construct vector $\Delta$ as follows,

$$\Delta_i = \frac{D(x_i^* - s_i^{(k)})}{t_i}, \quad \text{and} \quad D = \frac{-2c^T(x^* - s^{(k)})}{\sum_{i=1}^n (s_i^{(k)} - x_i^*)^2}.$$

It is not difficult to verify that $x^*$ is no longer the optimal solution of $\mathcal{I}_{\Delta}$:

$$\langle c + \Delta, x^* - s^{(k)} \rangle = \langle c, x^* - s^{(k)} \rangle + \langle \Delta, x^* - s^{(k)} \rangle$$

$$= \langle c, x^* - s^{(k)} \rangle - \sum_{i=1}^n \frac{2c^T(x^* - s^{(k)})}{\sum_{i=1}^n (s_i^{(k)} - x_i^*)^2} \cdot \frac{(x_i^* - s_i^{(k)})}{t_i} \cdot (x_i^* - s_i^{(k)})$$

$$= -\langle c, x^* - s^{(k)} \rangle < 0.$$

Then by Lemma C.1,

$$0.8 \ln(1/\delta) \leq \sum_{i=1}^n t_i \cdot \Delta_i^2$$

$$= \sum_{i=1}^n \left( \frac{D(x_i^* - s_i^{(k)})}{t_i} \right)^2$$

$$= \sum_{i=1}^n \left( \frac{(x_i^* - s_i^{(k)})^2}{t_i} \cdot D^2 \right)$$

$$= \sum_{i=1}^n \left( \frac{(x_i^* - s_i^{(k)})^2}{t_i} \cdot \left( \frac{-2c^T(x^* - s^{(k)})}{\sum_{i=1}^n (s_i^{(k)} - x_i^*)^2} \right)^2 \right)$$

$$= 4 \cdot \frac{(c^T(x^* - s^{(k)}))^2}{\sum_{i=1}^n (s_i^{(k)} - x_i^*)^2},$$

which is equivalent to

$$\sum_{i=1}^n \frac{(s_i^{(k)} - x_i^*)^2}{5t_i \ln(1/\delta)} \leq (c^T(x^* - s^{(k)}))^2.$$

Therefore $5t_i \ln(1/\delta)$ is a feasible solution of the convex program (3), which completes our proof.
C.3 Proof for Theorem 3.2

We prove the worst case lower bound for unknown $c$ case.

**Theorem 3.2** (Worst-case lower bound for unknown $c$). Let $n$ be a positive integer and $\delta \in (0, 0.1)$. For any $\delta$-correct algorithm $A$, there exists an infinite sequence of LP instances with $n$ variables, $I_1, I_2, \ldots$, such that $A$ takes
\[ \Omega \left( \text{Low}(I_k)(\ln |S_k^{(1)}| + \ln \delta^{-1}) \right) \]
samples in expectation on $I_k$, where $S_k^{(1)}$ is the set of all extreme points of the feasible region of $I_k$, and $\text{Low}(I_k)$ goes to infinity.

The following lemma will be used in the construction of desired LP instances.

**Lemma C.3.** Let $n$ be a positive integer. There exists a constant $c$, a positive integer $l = \Omega(n)$ and $z = 2^ cn$ sets $W_1, \ldots, W_z \subseteq [n]$ such that

- For all $i \in [z]$, we have $|W_i| = l = \Omega(n)$.
- For all $i \neq j$, $|W_i \cap W_j| \leq l/2$.

**Proof.** Define $l = n/10$. Let each $W_i$ be a uniformly random subset of $[n]$ with size $l$. Then it is satisfied that
\[ \Pr[|W_i \cap W_j| > l/2] \leq 2^{-\Omega(n)} \]
for all $1 \leq i, j \leq n, i \neq j$. So we can choose sufficiently small $c$ such that
\[ \Pr[\exists i \neq j, |W_i \cap W_j| > l/2] \leq z^2 2^{-\Omega(n)} < 1, \]
which implies the existence of a desired sequence of subsets. 

Now for any $\delta$-correct algorithm $A$, we prove the existence of LP instances $I_1, I_2, \ldots$, which all have $n$ variables.

For simplicity, all the linear program instances we construct in this proof share the same feasible region, which we define as follows. Let $W_1, \ldots, W_z \subseteq [n]$ be the sequence of subsets in Lemma C.3. For a subset $W \subseteq [n]$, we define a point $p^W$ as
\[ p^W_i = \begin{cases} 1, & \text{if } i \in W; \\ 0, & \text{otherwise}. \end{cases} \]

The feasible region we are going to use throughout this proof is the convex hull of $p^{W_1}, \ldots, p^{W_z}$.

To find a desired LP instance $I_k$, we first choose an arbitrary constant $\Delta_k$. We construct $z$ different LP instances $I_{\Delta_k,W_1}, \ldots, I_{\Delta_k,W_z}$ and show that at least one of them satisfies the condition in the theorem. Define the objective function $c^W_j$ of $I_{\Delta_k,W_j}$ to be
\[ c^W_j = \begin{cases} \Delta_k, & \text{if } i \in W_j; \\ -\Delta_k, & \text{otherwise}. \end{cases} \]

Then clearly the optimal solution of $I_{\Delta_k,W_j}$ is point $p^{W_j}$. We define $\Pr[A(I_{\Delta_k,W_i}) = p^{W_j}]$ to be the probability that algorithm $A$ outputs $p^{W_j}$ when the input is $I_{\Delta_k,W_i}$. Then we have
\[ \Pr[A(I_{\Delta_k,W_i}) = p^{W_j}] \geq 1 - \delta, \]
and
\[ \sum_{j: j \neq i} \Pr[A(I_{\Delta_i}, W_i) = p^{W_j}] \leq \delta. \]

Thus there must exist a \( W_k \) such that
\[ \Pr[A(I_{\Delta_i}, W_i) = p^{W_k}] \leq 2\delta/\varepsilon. \]

Let \( T \) be the number of samples used by algorithm \( A \) when the input is \( I_{\Delta_i}, W_i \). Since \( A \) is a \( \delta \)-correct algorithm, \( \Pr[A(I_{\Delta_i}, W_i) = p^{W_k}] \geq 1 - \delta > 0.9 \). So if we define event \( E \) to be the event that \( A \) outputs \( p^{W_k} \) and apply Lemma C.2,
\[ \mathbb{E}[T] \cdot (2\Delta^2) \geq d(\Pr[A(I_{\Delta_i}, W_i) = p^{W_k}], \Pr[A(I_{\Delta_i}, W_i) = p^{W_k}]) \]
\[ \geq \Omega(\ln(z/\delta)) \]
\[ = \Omega(\ln z + \ln(1/\delta)). \]

Here we use the following property of \( d(1 - \delta, \delta) \) function: for \( 0 < \delta < 0.1 \), \( d(1 - \delta, \delta) \geq 0.4 \ln(1/\delta) \). So we get a lower bound for \( \mathbb{E}[T] \),
\[ \mathbb{E}[T] \geq \Omega(\Delta^{-2}(\ln z + \ln(1/\delta))). \]

Meanwhile if we look at the Instance Lower Bound, \( \text{Low}(I_{\Delta_i}, W_k) \),
\[ \min_{\tau} \sum_{i=1}^{n} \tau_i \]
\[ \text{s.t.} \]
\[ \sum_{i=1}^{n} \frac{(p_i^{W_j} - p_i^{W_k})^2}{\tau_i} \leq \langle c^{W_k}, (p^{W_k} - p^{W_j}) \rangle^2, \forall j \]
\[ \tau_i \geq 0, \]
It is easy to verify that \( \tau_i = \frac{8}{\Delta^2} \) for all \( i \) is a feasible solution. So we have \( \text{Low}(I_{\Delta_i}, W_k) = \Theta\left(\frac{8n}{\Delta^2}\right) = \Theta(\Delta^{-2}) \). Therefore the number of samples that \( A \) will use on \( I_{\Delta_i}, W_k \) is \( \Omega\left(\text{Low}(I_{\Delta_i}, W_k)(\ln z + \ln(1/\delta))\right) \) in expectation.

By simply setting \( \Delta_k = \frac{1}{\varepsilon} \), we will get an infinite sequence of LP instances as stated in the theorem.

### C.4 Proof for Theorem 3.3

In this section, we prove the sample complexity of our successive elimination algorithm for unknown \( c \) case.

**Theorem 3.3** (Sample complexity of Algorithm 1). For the AIALO with unknown-c problem, Algorithm 1 is \( \delta \)-correct and, on instance \( I \), with probability \( 1 - \delta \) draws at most the following number of samples:
\[ O\left(\text{Low}(I) \ln \Delta^{-1}(\ln |S^{(1)}| + \ln \delta^{-1} + \ln \ln \Delta^{-1})\right), \]
where \( S^{(1)} \) is the set of all extreme points of the feasible region and \( \Delta \) is the gap in objective value between the optimal extreme point and the second-best,
\[ \Delta = \max_{x \in S^{(1)}} c^T x - \max_{x \in S^{(1)}} c^T x^*. \]

The following lemma will be used in our proof.
Lemma C.4. Given a set of Gaussian arms with unit variance and mean \(c_1, \ldots, c_n\). Suppose we take \(\tau_i\) samples for arm \(i\). Let \(X_i\) be the empirical mean. Then for an arbitrary vector \(p\),

\[
\Pr \left[ |p^T X - p^T c| \geq \varepsilon \right] \leq 2 \exp \left( -\frac{\varepsilon^2}{2 \sum_i \frac{p_i^2}{\tau_i}} \right)
\]

**Proof.** By definition, \(p^T X - p^T c\) follows Gaussian distribution with mean 0 and variance \(\sum_i p_i^2 / \tau_i\).

We define a good event \(E\) to be the event that \(|(x - y)^T (\hat{c}^{(r)} - c)| \leq \varepsilon^{(r)}/\lambda\) for all \(r\) and \(x, y \in S^{(r)}\). According to Lemma C.4,

\[
\Pr[E] \geq 1 - \sum_r \sum_{x \in S^{(r)}} \sum_{y \in S^{(r)}} 2 \exp - \frac{(\varepsilon/\lambda)^2}{2 \sum (x_i - y_i)^2 / \tau_i}.
\]

Since \(\tau\) satisfies the constraints in (4),

\[
\sum_r \sum_{x \in S^{(r)}} \sum_{y \in S^{(r)}} 2 \exp - \frac{(\varepsilon/\lambda)^2}{2 \sum (x_i - y_i)^2 / \tau_i} \leq \sum_r \sum_{x \in S^{(r)}} \sum_{y \in S^{(r)}} 2 \exp \left( -\ln(2/\delta^{(r)}) \right) = \sum_r \sum_{x \in S^{(r)}} \sum_{y \in S^{(r)}} \delta^{(r)} \leq \delta
\]

Therefore \(\Pr[E] \geq 1 - \delta\).

We first prove the correctness of the algorithm conditioning on \(E\).

**Lemma C.5.** When the good event \(E\) holds, the optimal LP solution \(x^* = \max_{Ax \leq b} c^T x\) will not be deleted.

**Proof.** Suppose to the contrary \(x^*\) is deleted in iteration \(r\), i.e., \(x^* \in S^{(r)}\) but \(x^* \notin S^{(r+1)}\). Then according to (5), when the objective function is \(\hat{c}^{(r)}\), \(x^*\) is at least \(\varepsilon^{(r)}/2 - 2\varepsilon^{(r)}/\lambda\) worse than \(x^{(r)}\),

\[
\langle x^{(r)} - x^*, \hat{c}^{(r)} \rangle > \varepsilon^{(r)}/2 + 2\varepsilon^{(r)}/\lambda.
\]

By the definition of the optimal solution \(x^*\),

\[
\langle c, x^* - x^{(r)} \rangle > 0.
\]

Combining the two inequalities will give

\[
\langle c - \hat{c}^{(r)}, x^* - x^{(r)} \rangle > \varepsilon^{(r)}/2 + 2\varepsilon^{(r)}/\lambda > \varepsilon^{(r)}/\lambda,
\]

contradictory to that event \(E\) holds.

We then bound the number of samples conditioning on \(E\). We first prove the following lemma.

**Lemma C.6.** When event \(E\) holds, all points \(s\) in set \(S^{(r+1)}\) satisfies

\[
\langle c, x^* - s \rangle < \varepsilon^{(r)}.
\]

**after the \(r^{th}\) iteration.**
Proof. Suppose when entering the $r^{th}$ iteration, there exists $s \in S^{(r)}$ such that $\langle c, x^* - s \rangle > \varepsilon^{(r)}$. Then since $\mathcal{E}$ holds and $\lambda = 10$,
\[
\langle c, x^* - s \rangle > (\mathcal{E}^{(r)}, x^* - s) - \varepsilon^{(r)}/\lambda \\
> (1 - 1/\lambda)\varepsilon^{(r)} \\
> \varepsilon^{(r)}/2 + 2\varepsilon^{(r)}/\lambda.
\]
By Lemma C.5, we have $x^* \in S^{(r)}$. Therefore $s$ will be deleted in this iteration. \qed

Now consider a fixed iteration $r$. Let $\tau^*$ be the optimal solution of the convex program (3) that computes $\text{low}(\mathcal{I})$. Define $\alpha = 32\lambda^2 \ln(2/\delta^{(r)})$. We show that $t = \alpha \tau^*$ is a feasible solution in the convex program (4) that computes $\text{LowAll}(S^{(r)}, \varepsilon^{(r)}, \delta^{(r)})$. For any $x, y \in S^{(r)}$,
\[
\sum \frac{(x_i - y_i)^2}{t_i} = \frac{1}{\alpha} \sum \frac{(x_i - y_i)^2}{\tau_i^*} \\
= \frac{1}{\alpha} \sum \frac{(x_i - x_i^* + x_i^* - y_i)^2}{\tau_i^*} \\
\leq \frac{1}{\alpha} \sum \frac{2(x_i - x_i^*)^2 + 2(x_i^* - y_i)^2}{\tau_i^*}
\]
due to the fact that $(a + b)^2 \leq 2a^2 + 2b^2$ for all $a, b \in \mathbb{R}$.

Since $\tau^*$ satisfies the constraints in $\text{Low}(\mathcal{I})$ function (3),
\[
\frac{1}{\alpha} \sum \frac{2(x_i - x_i^*)^2 + 2(x_i^* - y_i)^2}{\tau_i^*} \leq \frac{2}{\alpha} (\langle e^T (x^* - x) \rangle + \langle e^T (x^* - y) \rangle^2)
\]
And because of Lemma C.6,
\[
\frac{2}{\alpha} (\langle e^T (x^* - x) \rangle + \langle e^T (x^* - y) \rangle^2) \leq \frac{4}{\alpha} (\varepsilon^{(r-1)})^2 = \frac{(\varepsilon^{(r)})^2}{2\lambda^2 \ln(2/\delta^{(r)})}.
\]

So we have proved that $t = \alpha \tau^*$ is a feasible solution of the convex program that computes $\text{LowAll}(S^{(r)}, \varepsilon^{(r)}, \delta^{(r)})$. Thus the number of samples used in iteration $r$, $\sum_{i=1}^{n} t_i^{(r)}$, is no more than
\[
\sum_{i=1}^{n} t_i^{(r)} \leq \sum_{i=1}^{n} t_i = \alpha \sum_{i=1}^{n} \tau_i^* = O(\text{Low}(\mathcal{I}) (\ln |S^{(r)}| + \ln \delta^{-1} + \ln r)
\]

Conditioning on $\mathcal{E}$, the algorithm will terminate before $\lfloor \log(\Delta^{-1}) \rfloor + 1$ iterations according to Lemma C.6. Therefore the total number of samples is
\[
O \left( \text{Low}(\mathcal{I}) \ln \Delta^{-1} (\ln |S^{(1)}| + \ln \delta^{-1} + \ln \ln \Delta^{-1}) \right).
\]

D The Unknown Constraints Case

D.1 Proof for Theorem 4.1

**Theorem 4.1** (Lower bound for unknown $b$). Suppose we have a $(\delta, \varepsilon_1, \varepsilon_2)$-correct algorithm $A$ 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-$b$ with $n$ variables with objective function $||c||_\infty = 1$ such that $A$ 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.
Let $A$ be a $(\delta, \varepsilon_1, \varepsilon_2)$-correct algorithm. For a positive integer $n$, consider the following LP instance $I$ with $n$ variables and $n$ constraints,

$$\begin{align*}
\text{max} & \quad x_1 \\
\text{s.t.} & \quad x_1 \leq C, \\
& \quad x_1 + x_i \leq C, \quad \forall 2 \leq i \leq n, \\
& \quad x \geq 0.
\end{align*}$$

Clearly the optimal solution is $x^*_1 = C$ and $x^*_i = 0$ for $i > 1$. Every constraint is a binding constraint. Now we prove that for any $k \in [n]$, algorithm $A$ should take at least $\Omega \left( \ln(1/\delta) \cdot \max\{\varepsilon_1, \varepsilon_2\}^{-2} \right)$ for the $k^{th}$ constraint. We construct a new LP $I'$ by subtracting the right-hand side of $k^{th}$ constraint by $2(\varepsilon_1 + \varepsilon_2)$. Then $OPT(I; \varepsilon_1, \varepsilon_2)$ and $OPT(I'; \varepsilon_1, \varepsilon_2)$ must be disjoint, since for any $x \in OPT(I'; \varepsilon_1, \varepsilon_2)$, $x$ will not violate the $k^{th}$ constraint of $I'$ by more than $\varepsilon_2$, $x_1 \leq C - 2(\varepsilon_1 + \varepsilon_2) + \varepsilon_2 < C - 2\varepsilon_1$,

which means that $x \notin OPT(I; \varepsilon_1, \varepsilon_2)$. According to Lemma C.1,

$$E[\tau_k] \cdot 4(\varepsilon_1 + \varepsilon_2)^2 \geq 0.8 \ln(1/\delta)$$

And since $2 \max\{\varepsilon_1, \varepsilon_2\} \geq \varepsilon_1 + \varepsilon_2$,

$$E[\tau_k] = \Omega(\max\{\varepsilon_1, \varepsilon_2\}^{-2} \cdot \ln(1/\delta)).$$

### D.2 Proof for Theorem 4.2

Recall that our algorithm and the sample complexity theorem works as follows:

**Algorithm 4 Modified ellipsoid algorithm**

- Let $E^{(0)}$ 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$^9$ **do**

- Let $x^{(k)}$ be the center of $E^{(k)}$
- Call UCB method to get constraint $i$ or “feasible”
  - if $x^{(k)}$ is feasible then
    - Let $x \leftarrow x^{(k)}$ if $x$ is not initialized or $c^T x^{(k)} > c^T x$
    - $y \leftarrow -c$
  - else
    - $y \leftarrow A_i^T$
- end if
- Let $E^{(k+1)}$ be the minimal ellipsoid that contains $E^{(k)} \cap \{t : y^T t \leq y^T x^{(k)}\}$
- Let $k \leftarrow k + 1$

**end while**

Output $x$ or “failure” if it was never set.

---

$^9$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 $\sqrt{c^T P^{-1} c} \leq \min\{\varepsilon_1, \varepsilon_2\}$, where $P$ is the matrix corresponding to ellipsoid $E^{(k)}$ as discussed above.
Algorithm 5 UCB-method

Input $x^{(k)}$
Set $\delta' = \left(\frac{\delta}{20m}\right)^{2/3}$

\[ \text{loop} \]
1. Let $j$ be the constraint with the largest index,
   \[ j = \arg \max_i A_i x^{(k)} - \hat{b}_{j,T_i(t)} + U_i(T_i(t)), \]
   where $U_i(s) = 3\sqrt{2\sigma_i^2 \log(\log(3n/2)/\delta')} / \delta$ and $\hat{b}_{j,T_i(t)}$ as in Definition 4.1.
2. If $A_j x^{(k)} - \hat{b}_{j,T_i(t)} - U_j(T_j(t)) > 0$ return $j$.
3. If $A_j x^{(k)} - \hat{b}_{j,T_i(t)} + U_j(T_j(t)) < 0$ return “feasible”.
4. If $U_j(T_i(t)) < \varepsilon_2/2$ 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 $i \neq j$.

\[ \text{end loop} \]

Theorem 4.2 (Ellipsoid-UCB algorithm). The Ellipsoid-UCB algorithm is $(\delta, \varepsilon_1, \varepsilon_2)$-correct and with probability $1 - \delta$, draws at most the following number of samples:

\[ O \left( \sum_{i=1}^{m} \frac{\sigma_i^2}{\Delta_i^{2/3} \varepsilon_2} \log \frac{m}{\delta} + \sum_{i=1}^{m} \frac{\sigma_i^2}{\Delta_i^{2/3} \varepsilon_2} \log \log \left( \frac{\sigma_i^2}{\Delta_i^{2/3} \varepsilon_2} \right) \right). \]

Specifically, the number of samples used for $b_i$ is at most $\frac{\sigma_i^2}{\Delta_i^{2/3} \varepsilon_2} \left( \log(m/\delta) + \log \log(\sigma_i^2/\Delta_i^{2/3} \varepsilon_2) \right)$.

Our analysis is inspired by the techniques used in Jamieson et al. (2014). The following lemma is the same as Lemma 3 in Jamieson et al. (2014), and is simplified by setting $\varepsilon = 1/2$. We choose $1/2$ only for simplicity. It will not change our result asymptotically. The constant in this lemma can be optimized by selecting parameters carefully.

Lemma D.1. Let $X_1, X_2, \ldots$ be i.i.d. sub-Gaussian random variables with scale parameter $\sigma$ and mean $\mu$. We have probability at least $1 - 20 \cdot \delta^{5/2}$ for all $t \geq 1$, $\left\lfloor \frac{1}{2} \sum_{s=1}^{t} X_s - \mu \right\rfloor \leq L(t, \delta)$, where $L(t, \delta) = 3\sqrt{2\sigma^2 \log(\log(3t^{2}/\delta))/t}$.

Define event $A$ to be the event that $|\hat{b}_{i,t} - b_i| \leq U_i(t)$ for all $t \geq 0$ and $i \in [m]$. Since our definition of $U_i(t)$ is the same as $L(t, (\delta/20m)^{2/3})$ in Lemma D.1 with scale parameter $\sigma_i$, the probability that event $A$ holds is at least $1 - \delta$ according to union bound.

We prove the correctness and the sample number of the algorithm conditioning on that $A$ holds.

Correctness: We first prove that the output of our algorithm satisfies relaxed feasibility and relaxed optimality when $A$ holds. If our UCB-method always gives correct answer, the ellipsoid algorithm will be able to find an $\varepsilon_1$-suboptimal solution. So we only need to prove the correctness of the UCB-method.
• When UCB method returns a violated constraint \( j \) in line 2, it is indeed a violated one: since \( |\hat{b}_{j,T_j(t)} - b_j| \leq U_j(T_j(t)) \),
\[
A_jx_k - b_j \\
\geq A_jx_k - \hat{b}_{j,T_j(t)} - U_j(T_j(t)) \\
> 0.
\]

• When it returns “feasible” in line 3, no constraint is violated:
\[
A_ix_k - b_i \\
\leq A_ix_k - \hat{b}_{i,T_i(t)} + U_i(T_i(t)) \\
\leq A_jx_k - \hat{b}_{j,T_j(t)} + U_j(T_j(t)) \\
< 0, \ \forall i \in [m].
\]

• When it returns ”feasible” in line 4, no constraint is violated more than \( \varepsilon_2 \):
\[
A_jx_k - b_i \\
\leq A_ix_k - \hat{b}_{i,T_i(t)} + U_i(T_i(t)) \\
\leq A_jx_k - \hat{b}_{j,T_j(t)} + U_j(T_j(t)) \\
\leq A_jx_k - \hat{b}_{j,T_j(t)} - U_j(T_j(t)) + 2U_j(T_j(t)) \\
\leq 0 + \varepsilon_2, \ \forall i \in [m].
\]

Therefore the relaxed feasibility should be satisfied and the relaxed optimality is guaranteed by ellipsoid algorithm.

**Number of samples:** We bound the number of samples used on each constraint separately. The number of samples used on constraint \( i \) can be stated as the maximum \( T_i(t) \) where \( t \) is a mini-stage in which a sample of \( b_i \) is drawn. We bound \( T_i(t) \) by showing that \( U_i(T_i(t)) \) should be larger than a certain value if \( b_i \) is sampled at mini-stage \( t \). This immediately give us an upper bound of \( T_i(t) \) since \( U_i(t) \) is a decreasing function of \( t \). Suppose \( b_i \) is sampled at mini-stage \( t \) in ellipsoid iteration \( k \). Let \( i^* \) be the constraint with largest violation. Conditioning on \( A \) holds, the fact that constraint \( i \) have a larger index than \( i^* \) gives
\[
V_i(k) + 2U_i(T_i(t)) \\
\geq A_ix_k - \hat{b}_{i,T_i(t)} + U_i(T_i(t)) \\
\geq A_{i^*}x_k - \hat{b}_{i^*,T_{i^*}(t)} + U_{i^*}(T_{i^*}(t)) \\
\geq V_{i^*}(k).
\] (7)

which implies \( 2U_i(T_i(t)) \geq V^*(k) - V_i(k) \). Now look at line 2 in UCB-method. If a sample of \( b_i \) is drawn, we should not quit in this step. So if \( V_i(k) > 0 \), we must have
\[
V_i(k) - 2U_i(T_i(t)) \\
\leq A_ix_k - \hat{b}_{i,T_i(t)} - U_i(T_i(t)) \\
\leq 0.
\] (8)
Similarly, because of line 3 in UCB-method, if $V_i(k) \leq 0$, it should be satisfied that
\[
V_i(k) + 2U_i(T_i(t)) \geq A_i x_k - \hat{b}_{i,T_i(t)} + U_i(T_i(t)) \geq 0.
\] (9)

Putting inequality (7), (8) and (9) and $U_i(T_i(t)) \geq \varepsilon_2/2$ together, we get the conclusion that $2U_i(T_i(t)) \geq \max\{V^*(k) - V_i(k), |V_i(k)|, \varepsilon_2/2\} = \text{gap}_{i,\varepsilon_2/2}(k)$ should be satisfied if we draw a sample of $b_i$ at mini\-stage $t$ in ellipsoid iteration $k$.

Then we do some calculation,
\[
2U_i(T_i(t)) \geq \text{gap}_{i,\varepsilon_2/2}(k)
\]
\[
\Rightarrow 6\sqrt{\frac{2\sigma_i^2 \log(\log(3T_i(t)/2))/\delta'}{T_i(t)}} \geq \frac{\text{gap}_{i,\varepsilon_2/2}^2(k)}{72\sigma_i^4}
\]
\[
\Rightarrow T_i(t) \leq \frac{108\sigma_i^2}{\text{gap}_{i,\varepsilon_2/2}^2(k)} \log \left( \frac{20m}{\delta} \right)
\]
\[
+ \frac{72\sigma_i^2}{\text{gap}_{i,\varepsilon_2/2}^2(k)} \log \log \left( \frac{108\sigma_i^2}{\text{gap}_{i,\varepsilon_2/2}^2(k)\delta'} \right). \tag{10}
\]

In the last step, we use the fact that for $0 < \delta \leq 1$, $c > 0$,
\[
\frac{1}{t} \log \left( \frac{\log(3t/2)}{\delta} \right) \geq c
\]
\[
\Rightarrow t \leq \frac{1}{c} \log \left( \frac{2\log(3/(2c\delta))}{\delta} \right).
\]

Take maximum of (10) over all $k$ and according to the definition of $\Delta_{i,\varepsilon_2/2}$,
\[
T_i(t) \leq \frac{108\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2} \log \left( \frac{20m}{\delta} \right) + \frac{72\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2} \log \log \left( \frac{108\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2\delta'} \right)
\]

Therefore the overall number of samples is at most
\[
O \left( \sum_i \frac{\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2} \log \frac{m}{\delta} + \sum_i \frac{\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2} \log \log \left( \frac{\sigma_i^2}{\Delta_{i,\varepsilon_2/2}^2} \right) \right).
\]