Interactively Learning Preference Constraints in Linear Bandits

David Lindner [1] **Sebastian Tschiatschek** [2] **Katja Hofmann** [3] **Andreas Krause** [1]

Abstract

We study sequential decision-making with known rewards and unknown constraints, motivated by situations where the constraints represent expensive-to-evaluate human preferences, such as safe and comfortable driving behavior. We formalize the challenge of interactively learning about these constraints as a novel linear bandit problem which we call *constrained linear best-arm identification*. To solve this problem, we propose the *Adaptive Constraint Learning* (ACOL) algorithm. We provide an instance-dependent lower bound for constrained linear best-arm identification and show that ACOL's sample complexity matches the lower bound in the worst-case. In the average case, ACOL's sample complexity bound is still significantly tighter than bounds of simpler approaches. In synthetic experiments, ACOL performs on par with an oracle solution and outperforms a range of baselines. As an application, we consider learning constraints to represent human preferences in a driving simulation. ACOL is significantly more sample efficient than alternatives for this application. Further, we find that learning preferences as constraints is more robust to changes in the driving scenario than encoding the preferences directly in the reward function.

1. Introduction

Often, (sequential) decision-making problems are formalized as maximizing an unknown reward function that captures an expensive-to-evaluate objective, for example, user preferences (see Chapter 1 of Lattimore & Szepesvári (2020) for examples). However, in many practical situations, it can be more natural to model problems with a known reward function and unknown, expensive-to-evaluate constraints.

[1]ETH Zurich, Switzerland [2]University of Vienna, Austria [3]Microsoft Research Cambridge, UK. Correspondence to: David Lindner <david.lindner@inf.ethz.ch>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

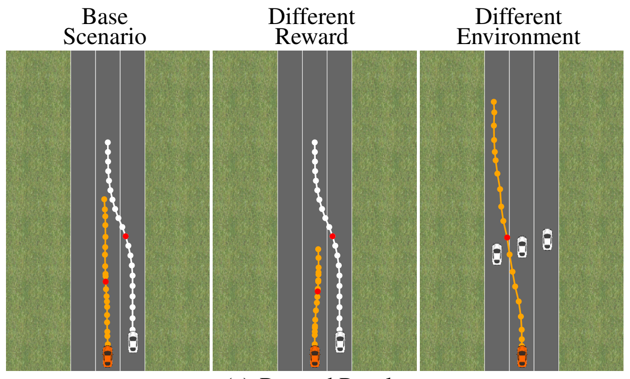
For example, a cookie manufacturer might want to create a low-calorie cookie.[4] The cookie should have the lowest amount of calories possible, but at least 95% of customers should like it. To evaluate this constraint, the manufacturer has to produce specific cookies and test them with customers. The reward, i.e., the amount of calories for a recipe, is easy to evaluate without producing a cookie. Because customer trials are expensive, the cookie manufacturer wants to find the best constrained solution with as few trials as possible.

As a second example, consider finding safe control parameters for an autonomous car. A car manufacturer might have a set of controllers to choose from that perform a specific task, such as reaching a target destination as quickly as possible. The ideal controller achieves this task well and drives safely and comfortably. Whereas the objective – travel time – is easy to specify as a reward function, the constraints – perceived safety and comfort – may require feedback from human drivers and passengers. Similarly as in the previous example, the manufacturer's goal is to find the best, safe controller with as few trials that involve human feedback as possible. We assume that the controllers are evaluated in a simulation, so it is acceptable to evaluate an unsafe controller during training; however, the constraints have to be satisfied during deployment.

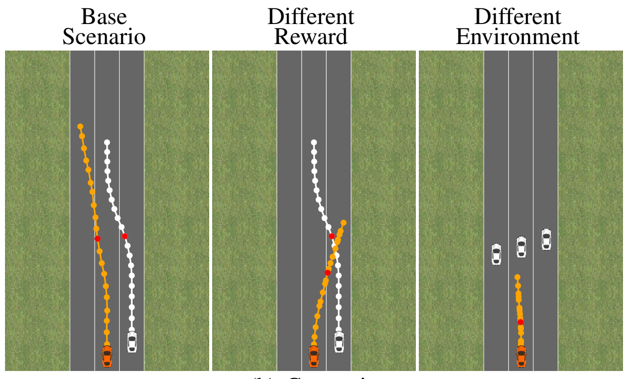
In both examples, the decision-making problem is naturally characterized by an easy-to-evaluate part (the reward) and an expensive-to-evaluate part (the constraint). Additionally, we observe that constraints are more robust to changes in the environment and can be transferred to selecting controllers for different goals, in contrast to encoding the constraints as a penalty in the reward function (see Figure 1). Hence, in this paper, we study *learning about unknown, expensive-to-evaluate constraints*.

Specifically, we propose a two-phase approach to solving problems with unknown constraints. In the first phase, we learn to estimate the expensive-to-evaluate constraint function well enough for solving the constraint optimization problem. In the second phase, we recommend a solution.

[4]Example adapted from Gelbart et al. (2014).



(a) Reward Penalty



(b) Constraint

Figure 1. We want to select a controller for driving the orange car. In the base scenario (left image), the car should drive at velocity v, which we encode as reward. We model other driving rules, like "usually drive in a lane" or "don't get too close to other cars", either as a penalty on the reward function (in (a)) or as a constraint (in (b)). In the "different reward" scenario (middle image), the car should pull over to the right of the street instead of keeping the velocity. If we reuse the same reward penalty for this task, the controller does not achieve the task because the penalty is too strong. So, we would have to tune the penalty for this new task, whereas the constrained controller still completes the new task safely without any tuning. In the "different environment" scenario (right image), the goal of driving at a target velocity remains the same, but the environment changes. In this changed environment, three vehicles block the road. Here, the penalized controller trades off the penalty with achieving high reward and tries to go through the cars to keep the velocity, which is too dangerous. On the other hand, the constraint formulation does not allow violating a constraint such as "don't get too close to other cars" which makes the orange car stop before the street is blocked. In Section 4.3, we study this example in more detail.

Constraint violations are allowed in the first phase, but the final recommendation has to satisfy the constraints.

Contributions. We formalize learning about unknown constraints to find the best constrained solution as a novel linear bandit problem (Section 3) which we call *constrained linear best-arm identification* (CBAI). We provide an instance-dependent sample complexity lower bound (Section 3.1) and propose *Adaptive Constraint Learning* (ACOL), an algorithm that almost matches this lower bound (Section 3.3). Our empirical evaluation shows that ACOL gets close to the performance of an oracle solution that has access to the true constraint function while outperforming a range of simpler baselines (Section 4.1). As a concrete application, we consider learning driving behavior in a simulation, where the constraints represent human preferences about driving behavior (Section 4.3). We demonstrate empirically that ACOL can learn these constraints and propose heuristic variants of the algorithm that empirically improve sample efficiency. Additionally, we quantify the observation that learning driving preferences as constraints instead of rewards increases the robustness and transferability of the learned preferences.

2. Related Work

Learning constraints is similar to actively classifying arms as "feasible" or "infeasible"; but, in contrast to typical active learning (Settles, 2012), we do not need to classify all arms. Instead, we only want to find the best feasible arm, which can require fewer samples than classifying all arms. Our problem formalization as a linear *multi-armed bandit best-arm identification* problem (Audibert et al., 2010) is similar to Soare et al. (2014) in the unconstrained setting, but focused on learning constraints.

Much prior work on constraints in multi-armed bandits considers other notions of constraints than we do. For example, constraints holding in expectation rather than with high probability (Pacchiano et al., 2021), or constraints in the form of a lower bound (threshold) on the reward (Locatelli et al., 2016; Kazerouni et al., 2017; Kano et al., 2019; Khezeli & Bitar, 2020).

Amani et al. (2019) and Moradipari et al. (2021) consider a linear bandit setting with a separate (linear) constraint function. Both differ from our work in three important ways: (1) they assume an unknown reward function whereas we assume the reward to be known, (2) they focus on cumulative regret minimization whereas we focus on best-arm identification, and (3) they require the constraints to be satisfied during exploration whereas we only require them to be satisfied for the final recommendation. These works adapt bandit algorithms based on upper confidence bounds (Amani et al., 2019) or Thompson sampling (Moradipari et al., 2021) to minimize regret in the constraint setting. To enable safe exploration, they need to assume a convex and compact set of arms; we do not require this assumption.

Wang et al. (2022) also study best-arm identification with linear constraints. In contrast to our work, they assume unknown rewards and focus on safety constraints that must be satisfied during exploration. To make this possible, they need to make more assumptions about the structure of the set of arms. In particular, they assume that the decision maker can only query in each dimension independently. Because of this, their algorithm cannot be applied to our setting without significant changes.

Our algorithm is conceptually similar to other bandit algorithms based on the principle of eliminating sub-optimal arms step-by-step. Our theoretical analysis employs similar tools as that used for best-arm-identification in unconstrained linear bandits (Soare et al., 2014; Fiez et al., 2019).

Some works in Bayesian optimization (BO) also study the problem of exploring to find the best constrained solution to a problem with an expensive-to-evaluate constraint function. Common approaches heuristically extend BO methods to incorporate an unknown constraint function (Gardner et al., 2014; Hernández-Lobato et al., 2016; Perrone et al., 2019). In contrast to this line of work, we obtain sample complexity guarantees by focusing on linear constraint functions. Similar to the bandit literature, most work on BO with constraints focuses on the setting where safety constraints must hold during exploration (e.g., Sui et al., 2015).

We apply our algorithm to the problem of learning from human preferences, which is essential for building systems with hard-to-specify goals, e.g., in robotics (Daniel et al., 2014). We use an environment by Sadigh et al. (2017) who model human preferences as rewards rather than constraints.

3. The Linear Constrained Best-Arm Identification Problem

We seek to find the best constrained solution from a discrete set of options represented by feature vectors $x \in \mathcal{X} \subset \mathbb{R}^d$. We assume that both the known reward function and the unknown constraint function are linear in x.

Definition 1. *A constrained linear best-arm identification (CBAI) problem $\nu = (\mathcal{X}, \theta, \phi, \tau)$ consists of a finite action set $\mathcal{X} \subset \mathbb{R}^d$, a reward parameter $\theta \in \mathbb{R}^d$, a constraint parameter $\phi \in \mathbb{R}^d$, and threshold $\tau \in \mathbb{R}$. The decision-maker knows $\mathcal{X}$ and θ, but not ϕ and τ. In each iteration, the decision-maker selects an arm $x \in \mathcal{X}$ and observes $\phi^T x + \eta_x$, where η_x is sub-Gaussian noise. Their goal is to identify a constrained optimal arm*

$$x^* \in \underset{x \in \mathcal{X}, \phi^T x \leq \tau}{\operatorname{argmax}} \; \theta^T x$$

within as few iterations as possible.

In our initial example, $\mathcal{X}$ contains all potential cookie recipes. $\theta^T x$ encodes the amount of calories for recipe x, which the decision-maker knows and wants to minimize. $\phi^T x$ encodes the unknown customer preferences, which the decision-maker must infer from as few experiments as possible. In the following, we assume w.l.o.g. $\tau = 0$ but generalization to $\tau \neq 0$ is straightforward. If τ is unknown, we can simply model it as a constant shift in the const To simplify notation, we omit τ and talk about a CBAI problem $\nu = (\mathcal{X}, \theta, \phi)$.

3.1. Lower Bounds

We first provide a lower bound on the sample complexity of solving a given CBAI problem. The following theorem states how many samples are necessary to distinguish a given CBAI instance from the closest instance with a different solution, which is necessary to solve an instance.

Theorem 1 (CBAI lower bound)**.** *Assume $\eta_x \sim \mathcal{N}(0, 1)$ for all $x \in \mathcal{X}$. For any CBAI problem $\nu = (\mathcal{X}, \theta, \phi)$, there exists another CBAI problem $\nu' = (\mathcal{X}, \theta, \phi')$ with the same set of actions $\mathcal{X}$ and reward parameter θ but a different constraint parameter and optimal arm, such that the expected number of iterations τ needed by any allocation strategy that can distinguish between ν and ν' with probability at least $1 - \delta$ is lower bounded as*

$$\mathbb{E}[\tau] \geq 2 \log\left(\frac{1}{2.4\delta}\right) \max_{x \in \mathcal{X}_\theta^{\geq}(x_\nu^*)} \frac{\|x\|^2_{A_\lambda^{-1}}}{(\phi^T x)^2},$$

where λ is a probability distribution over arms which the allocation strategy follows, i.e., $\lambda(x)$ is the probability that it pulls arm x, $A_\lambda = \sum_x \lambda(x) x x^T$ is the design matrix, $\mathcal{X}_\theta^{\geq}(x_\nu^*) = \{x' \in \mathcal{X} | \theta^T x' \geq \theta^T x_\nu^*\}$ is the set of all arms with reward no less than x_ν^*, the optimal arm for problem ν.

The proof in Appendix A.1 uses a proof strategy similar to that for lower bounds for standard linear bandits (Soare et al., 2014; Soare, 2015; Fiez et al., 2019). We consider the log-likelihood ratio of making a series of observations in instance ν compared to ν' and consider how we can choose ν' to have a different solution but a small log-likelihood ratio, i.e., the decision-maker makes similar observations as if they were in ν. In contrast to the standard linear bandit case, we need to carefully reason about the constraints when ensuring that ν' has a different solution than ν. We distinguish the case that the solution of ν is infeasible in ν' and the case that an arm with larger reward is feasible in ν' but not ν. Reasoning about these two cases yields the result.

Our lower bound has a similar form as those for best-arm identification in linear bandits (Soare et al., 2014). In particular, we have the same uncertainty term in the numerator. Instead of a suboptimality gap in the denominator, we get the distance to the constraint boundary: the problem is harder if arms are closer to the constraint boundary. However, our maximization is over individual arms instead of directions, i.e., pairs of arms, and the set $\mathcal{X}_\theta^{\geq}(x_\nu^*)$ does not appear in the linear bandit case.

We want to characterize the sample complexity of different algorithms for solving the CBAI problem. To this end, let us define the sample complexity of a given problem instance using the lower bound we just derived.

Definition 2 (CBAI sample complexity)**.** *We define the* sample complexity *of a CBAI problem ν as*

$$H_{\mathrm{CLB}}(\nu) \coloneqq \min_{\lambda} \max_{x \in \mathcal{X}_\theta^{\geq}(x_\nu^*)} \frac{\|x\|^2_{A_\lambda^{-1}}}{(\phi^T x)^2}$$

This describes the best sample complexity that any algorithm can achieve on CBAI problem ν. It will also be helpful to have a worst-case upper bound on $H_{\mathrm{CLB}}(\nu)$ as a point of comparison, which the next proposition provides.

Proposition 1. *For any CBAI problem ν, we have $H_{\mathrm{CLB}}(\nu) \leq d/(C^+_{\min})^2$, where $C^+_{\min} = \min_{x\in\mathcal{X}} |\phi^T x|$. This bound is tight, i.e, there is an instance ν, such that we have $H_{\mathrm{CLB}}(\nu) = d/(C^+_{\min})^2$.*

This results indicates that a CBAI problem is harder if it has a larger dimension d, or if the distance of the arm that is closest to the constraint boundary ($C^+_{\min}$) is smaller. This worst case bound corresponds to situations where all arms are linearly independent and pulling one arm does not provide any information about any other arm.

Oracle solution. We can make the definition of sample complexity more concrete by considering an oracle solution that has access to the true constraint value to select which arms to query. The oracle selects arms by explicitly minimizing H_{CLB}:

$$\lambda^\star \in \operatorname*{argmin}_{\lambda} \max_{x \in \mathcal{X}_\theta^{\geq}(x_\nu^*)} \frac{\|x\|^2_{A_\lambda^{-1}}}{(\phi^T x)^2}.$$

The oracle prefers arms with high uncertainty (high $\|x\|^2_{A_\lambda^{-1}}$) and arms close to the constraint boundary (low $(\phi^T x)^2$). Moreover, it focuses on reducing the uncertainty about arms that have higher reward than the true optimal arm (arms in $\mathcal{X}_\theta^{\geq}(x_\nu^*)$). In Appendix B.1, we show that this oracle solution has sample complexity on order $H_{\mathrm{CLB}}(\nu)$, i.e., it is indeed optimal.

3.2. Confidence Intervals for Linear Regression

Our algorithms rely on high probability confidence intervals on the linear constraints constructed from observations. Hence, let us briefly review how to construct such confidence intervals from observations with sub-Gaussian noise.

Suppose, an algorithm queried a sequence of arms $\mathbf{x}_t = (x_1, \dots, x_t)$. For a given x_i, it observed $\tilde{y}_i = \phi^T x_i + \eta_{x_i}$, where ϕ is the true constraint parameter, and η_{x_i} is sub-Gaussian noise. We now aim to find confidence intervals such that $\phi^T x \in [l^t_\phi(x), u^t_\phi(x)]$ with probability at least $1-\delta$, where $l^t_\phi(x) = \hat{\phi}^T x - \sqrt{\beta_t}\|x\|_{A^{-1}_{\mathbf{x}_t}}$ and $u^t_\phi(x) = \hat{\phi}^T x + \sqrt{\beta_t}\|x\|_{A^{-1}_{\mathbf{x}_t}}$. Based on these confidence intervals, we can decide whether a given arm is likely feasible or not.

If the queries follow a distribution that does not depend on the observations, it is straightforward to derive confidence intervals (e.g., Chapter 20 in Lattimore & Szepesvári (2020)).

Proposition 2. *Let $\mathbf{x}_t = (x_1, \dots, x_t)$ be a sequence of arms from a fixed allocation for which we have observed $\phi^T x_i + \eta_{x_i}$ where η_{x_i} is independent sub-Gaussian noise. If we estimate $\hat{\phi}$ from the observations using least-squares regression and choose $\beta_t = \sqrt{2\log(|\mathcal{X}|/\delta)}$ then we have $P(\exists x \in \mathcal{X} : \phi^T x \notin [l^t_\phi(x), u^t_\phi(x)]) \leq \delta$.*

However, in sequential decision-making we usually want to adapt our strategy after making observations. In this case, we need to be more careful in constructing confidence intervals, as observed by Abbasi-Yadkori et al. (2011). Unfortunately, the resulting confidence intervals are weaker than those for static allocations by a factor of $\sqrt{d}$.

Proposition 3 (Theorem 2 by Abbasi-Yadkori et al. (2011))**.** *Let $\mathbf{x}_t = (x_1, \dots, x_t)$ be a sequence of points selected with a possibly adaptive strategy for which we have observed $\phi^T x_i + \eta_{x_i}$ where η_{x_i} is independent sub-Gaussian noise. Assume, that $\|\phi\|_2 \leq S$ and $\|x\|_2 \leq L$ for all $x \in \mathcal{X}$. If we estimate $\hat{\phi}$ from the observations using least-squares regression, then for every $x \in \mathbb{R}^d$ and for all $t \geq 0$: $P(\exists x \in \mathcal{X} : \phi^T x \notin [l^t_\phi(x), u^t_\phi(x)]) \leq \delta$ with $\beta_t = \sqrt{d \log\left((1 + tL^2/\lambda)/\delta\right)} + \sqrt{\lambda}S$.*

3.3. Algorithms Using Static Confidence Intervals

To design an algorithm for solving CBAI problems, we need to decide (1) which arms to pull during exploration and (2) when we can stop the algorithm and return the correct arm with high probability. First, let us address the second question and then get back to the first one.

Stopping condition. Using the past observations, we can define confidence intervals for the constraint value of each arm. Let $l^t_\phi(x)$ and $u^t_\phi(x)$ be such that we know with high probability (w.h.p.) $\phi^T x \in [l^t_\phi(x), u^t_\phi(x)]$. Now we can also determine w.h.p. that all arms with $l^t_\phi(x) > 0$ are infeasible, and all arms with $u^t_\phi(x) \leq 0$ are feasible. Moreover, we can identify suboptimal arms by considering $\bar{r} = \max_{u^t_\phi(x)\leq 0} \theta^T x$. The solution to this optimization problem are the highest-reward arms that are feasible w.h.p. Therefore, all arms with reward less than $\bar{r}$ are clearly suboptimal. Combining these observations, we can define a set of arms that we are uncertain about, i.e., that could still be optimal:

$$\mathcal{U}_t = \{x \in \mathcal{X} | l_\phi^t(x) \leq 0 \text{ and } u_\phi^t(x) > 0 \text{ and } \theta^T x > \bar{r}\}$$

Note, that if $\mathcal{U}_t$ is empty, we can stop and return an arm in $\text{argmax}_{u_\phi^t(x)\leq 0}\, \theta^T x$. This arm will be optimal w.h.p.

Arm selection criterion. In each iteration, we have to decide which arm to pull. We could, e.g., combine the above stopping condition with querying uniformly random arms. This algorithm would return the correct optimal arm with high probability. However, random querying will usually not be the most sample efficient approach. Another natural approach is to select the arms that we are most uncertain about, which is sometimes called *uncertainty sampling*. We could, e.g., choose a fixed allocation

$$\lambda^{\text{G}} \in \operatorname*{argmin}_\lambda \max_{x\in\mathcal{X}} \|x\|_{A_\lambda^{-1}}.$$

This approach is also called *G-Allocation* in the experimental design literature. We show in Appendix B.2 that G-Allocation matches the worst-case lower bound in Proposition 1. However, we can do better by *focusing on arms that we cannot yet exclude as being certainly feasible, infeasible, or suboptimal*. Concretely, we modify G-Allocation to reduce uncertainty only about arms in $\mathcal{U}_t$:

$$\lambda^{\text{ACOL}} \in \operatorname*{argmin}_\lambda \max_{x\in\mathcal{U}_t} \|x\|_{A_\lambda^{-1}}$$

Rounding. All algorithms implementing a static allocation require a rounding procedure to translate an allocation λ into a finite sequence of arms $x_1, \dots, x_n$. The experimental design literature provides various efficient rounding procedures that are ε-approximate. We use a standard procedure described in Chapter 12 of Friedrich (2006).

Adaptive Constraint Learning (ACOL). Algorithm 1 shows the full algorithm we call *Adaptive Constraint Learning* (ACOL). The algorithm proceeds in rounds. In each round t it pulls arms to reduce the uncertainty about arms in $\mathcal{U}_t$, then updates $\mathcal{U}_t$, and decides if it can stop and return a recommendation. The round length N_t is chosen carefully to allow us to provide a tight sample complexity result.

The following theorem – the main theoretical result of our paper – establishes that ACOL returns the correct optimal solution to any CBAI problem and provides an upper bound on the number of samples necessary.

Theorem 2 (ACOL sample complexity)**.** *Assume Algorithm 1 is implemented with an ε-approximate rounding strategy. Then, after N iterations the algorithm returns an optimal arm with probability at least $1-\delta$, and we have:*

$$N \leq 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\sum_{t=1}^{\bar{t}} \min_\lambda \max_{x\in\mathcal{U}_t} \frac{\|x\|^2_{A_\lambda^{-1}}}{(\phi^T x)^2} + \bar{t}$$
$$\leq 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\bar{t}\bar{H}_{\text{CLB}}(\nu) + \bar{t}$$

where $\bar{H}_{\text{CLB}}(\nu) = \min_\lambda \max_{x\in\mathcal{X}} \|x\|^2_{A_\lambda^{-1}}/(\phi^T x)^2$, and $\bar{t} = \lceil -\log_2 C^+_{\min}\rceil$. Moreover, $\bar{H}_{\text{CLB}}(\nu) \leq d/(C^+_{\min})^2$

Algorithm 1 Adaptive Constraint Learning (ACOL).

```
1:  Input: significance $\delta$
2:  $\mathcal{U}_1 \leftarrow \mathcal{X}$                                    (uncertain arms)
3:  $\mathcal{F}_1 \leftarrow \varnothing$                                    (feasible arms)
4:  $t \leftarrow 1$                                                 (round)
5:  while $\mathcal{U}_t \neq \varnothing$ do
6:      $\delta_t \leftarrow \delta^2/t^2$
7:      $\lambda_t^* \leftarrow \text{argmin}_\lambda \max_{x\in\mathcal{U}_t} \|x\|^2_{A_\lambda^{-1}}$
8:      $\rho_t^* \leftarrow \min_\lambda \max_{x\in\mathcal{U}_t} \|x\|^2_{A_\lambda^{-1}}$
9:      $N_t \leftarrow \max\left\{\left\lceil 2^{2t+3}\log\left(\frac{|\mathcal{X}|}{\delta_t}\right)(1+\varepsilon)\rho_t^*\right\rceil, r(\varepsilon)\right\}$
10:     $\mathbf{x}_{N_t} \leftarrow \texttt{Round}(\lambda_t^*, N_t)$
11:     Pull arms $x_1, \dots, x_{N_t}$ and observe constraint values
12:     $t \leftarrow t+1$
13:     Update $\hat{\phi}_t$ and $A$ based on new data
14:     $l_\phi^t(x) \leftarrow \hat{\phi}_t^T x - \sqrt{\beta_t}\|x\|_{A^{-1}}$ for all arms $x$
15:     $u_\phi^t(x) \leftarrow \hat{\phi}_t^T x + \sqrt{\beta_t}\|x\|_{A^{-1}}$ for all arms $x$
16:     $\mathcal{F}_t \leftarrow \mathcal{F}_{t-1} \cup \{x | u_\phi^t(x) \leq 0\}$
17:     $\bar{r} \leftarrow \max_{x\in\mathcal{F}_t} \theta^T x$
18:     $\mathcal{U}_t \leftarrow \mathcal{U}_{t-1} \setminus \{x | l_\phi^t(x) > 0\} \setminus \{x | u_\phi^t(x) \leq 0\}$
            $\setminus \{x | \theta^T x < \bar{r}\}$
19: end while
20: return $x^* \in \text{argmax}_{x\in\mathcal{F}_t} \theta^T x$
```

We prove the theorem in Appendix A. The key step uses Proposition 2 to show that the confidence intervals shrink exponentially. This implies that in a logarithmic number of rounds, the largest confidence interval will be less than $C^+_{\min}$; and once this is the case, $\mathcal{U}_t$ is empty and the algorithm returns the correct solution. Combining this with the round lengths of N_t allows us to prove the result.

The sample complexity of ACOL is of order $\bar{H}_{\text{CLB}}(\nu)$, except for logarithmic factors. Also, we show that $\bar{H}_{\text{CLB}} \leq d/(C^+_{\min})^2$, so the bound matches the lower bound of Proposition 1 for worst-case instances, but it is much tighter for benign instances. In particular, the bound in Theorem 2 contains the same min-max problem as the instance dependent sample complexity $H_{\text{CLB}}(\nu)$, only with the maximization being over different sets, namely $\mathcal{U}_t$ instead of $\mathcal{X}_\theta^{\geq}(x_\nu^*)$. Note, that we cannot expect a practical algorithm to only explore arms in $\mathcal{X}_\theta^{\geq}(x_\nu^*)$ because we do not know x_ν^* a priori. Instead, ACOL explores in $\mathcal{U}_t$, a conservative estimate of $\mathcal{X}_\theta^{\geq}(x_\nu^*)$ that shrinks over time given the knowledge so far. Theorem 2 does not exactly match the instance dependent lower bound, but the difference only depends on how well $\mathcal{U}_t$ approximates the set of relevant arms.

Algorithm 2 Greedy Adaptive Constraint Learning (G-ACOL).

```
1: Input: $\beta_t$, $\lambda$
2: initialize $\hat{S}_1(\hat{\phi}), \mathcal{U}_1 \leftarrow \mathcal{X}, \mathcal{F}_1 \leftarrow \varnothing, A \leftarrow \lambda I, t \leftarrow 1$
3: while $\mathcal{U}_t \neq \varnothing$ do
4:     $x^* \leftarrow \max_{x \in \mathcal{U}_t} \|x\|^2_{A^{-1}}$
5:     Pull arm $x^*$ and observe constraint value
6:     $t \leftarrow t+1$, $A \leftarrow A + xx^T$
7:     $l^t_\phi(x) \leftarrow \hat{\phi}_t^T x - \sqrt{\beta_t}\|x\|_{A^{-1}}$ for all arms $x$
8:     $u^t_\phi(x) \leftarrow \hat{\phi}_t^T x + \sqrt{\beta_t}\|x\|_{A^{-1}}$ for all arms $x$
9:     $\mathcal{F}_t \leftarrow \mathcal{F}_{t-1} \cup \{x | u^t_\phi(x) \leq 0\}$
10:    $\bar{r} \leftarrow \max_{x \in \mathcal{F}_t} \theta^T x$
11:    $\mathcal{U}_t \leftarrow \mathcal{U}_{t-1} \setminus \{x | l^t_\phi(x) > 0\} \setminus \{x | u^t_\phi(x) \leq 0\}$
                $\setminus \{x | \theta^T x < \bar{r}\}$
12: end while
13: return $x^* \in \operatorname{argmax}_{x \in \mathcal{F}_t} \theta^T x$
```

3.4. Algorithms Using Adaptive Confidence Intervals

Whereas the algorithm we just introduced comes with a strong sample complexity guarantee, it is impractical in various ways, primarily because of the round-based structure. In particular, the algorithm requires a rounding procedure to determine a sequence of actions; it then follows this sequence for a predefined round length and can not stop before finishing a round. Also, in between rounds, the algorithm discards all previously made observations, which is necessary to apply Proposition 2.

Next, we present an alternative version of this algorithm that uses the adaptive confidence intervals of Proposition 3. This allows us to remove the round-based structure in favor of a greedy algorithm that does not have the same limitation. This algorithm, which we call *Greedy Adaptive Constraint Learning* (G-ACOL), is shown in Algorithm 2. Unfortunately, for G-ACOL, we can only provide significantly weaker sample complexity guarantees; but we find it performs well empirically.

Since the adaptive confidence intervals hold for all $t > 0$ simultaneously, we can now check the stopping condition after each sample. Instead of determining a static allocation that reduces uncertainty about the uncertain arms, we now greedily select the arm to pull that reduces uncertainty within $\mathcal{U}_t$ the most. Thanks to Proposition 3, this algorithm still stops and returns the correct solution. However, it achieves worse sample complexity due to the additional factor of $\sqrt{d}$ in Proposition 3.

Heuristic modifications. There is a variety of heuristic modifications that we can make to G-ACOL to improve its practical performance at the cost of losing some theoretical guarantees. First, we could use a different query rule within the set of uncertain arms, such as uniformly random querying, which reduces computational cost. Second, the β_t resulting from Proposition 3 tends to be very large. In practice, we can try to tune β_t to get good confidence intervals that are much smaller than the ones suggested by the theory. Third, we can turn the algorithm into an "anytime" algorithm by defining a recommendation rule, such as recommending the best arm that is certainly feasible. Then, we can stop the algorithm after an a priori unknown budget of queries and receive a best guess for the optimal arm.

4. Experiments

We perform three experiments. First, in Section 4.1, we consider synthetic CBAI instances to evaluate ACOL and compare it to natural baselines. Additionally, we investigate the effect of various heuristic modifications to the algorithm. Second, in Section 4.2, we compare ACOL to algorithms that safely minimize regret. And, third, in Section 4.3, we consider learning constraints that represent human preferences in a simulated driving scenario. This experiment illustrates how to model preference learning problems as CBAI problems. In the driving simulation, we also demonstrate the benefits of learning constraints in terms of robustness and transferability.

We provide more details on the experiments in Appendix C and we provide the full source code to reproduce our experiments.[1] For all experiments we use a significance of $\delta = 0.05$ and, if not stated differently, observations have Gaussian noise with $\sigma = 0.05$.

4.1. Synthetic Experiments

We consider two synthetic CBAI instances and a range of baselines and multiple variants of ACOL/ G-ACOL.

Instance 1 – Irrelevant dimensions. First, we consider CBAI instances which contain a number of dimensions that are irrelevant for learning the correct constraint boundary. The problems have dimension d, and $d+1$ arms: $x_1, \dots, x_{d+1}$. For each $i = 1, \dots, d-1$, we have $x_i = \mathbf{e}_i$, whereas $x_d = (1-\varepsilon)\mathbf{e}_d$, and $x_{d+1} = (1+\varepsilon)\mathbf{e}_d$, for some $\varepsilon > 0$. $\mathbf{e}_i$ denotes the i-th unit vector. The reward and constraint parameter are both $\theta = \phi = \mathbf{e}_d$. We define a threshold $\tau = 1$; hence, $x_1, \dots, x_{d-1}$ are feasible but suboptimal, x_d is optimal and x_{d+1} is infeasible. Importantly, the arms $x_1, \dots, x_{d-1}$ are "irrelevant" to finding the correct constraint boundary between x_d and x_{d+1}. An ideal algorithm would focus its queries primarily on x_d and x_{d+1}. We can vary the problem difficulty by changing ε (more difficult for small values), and d (more difficult for large values).

Instance 2 – Unit sphere. To create CBAI instances with a range of different reward and constraint functions, we sample arms $x_1, \dots, x_n$ uniformly from a d-dimensional

[1] https://github.com/lasgroup/adaptive-constraint-learning

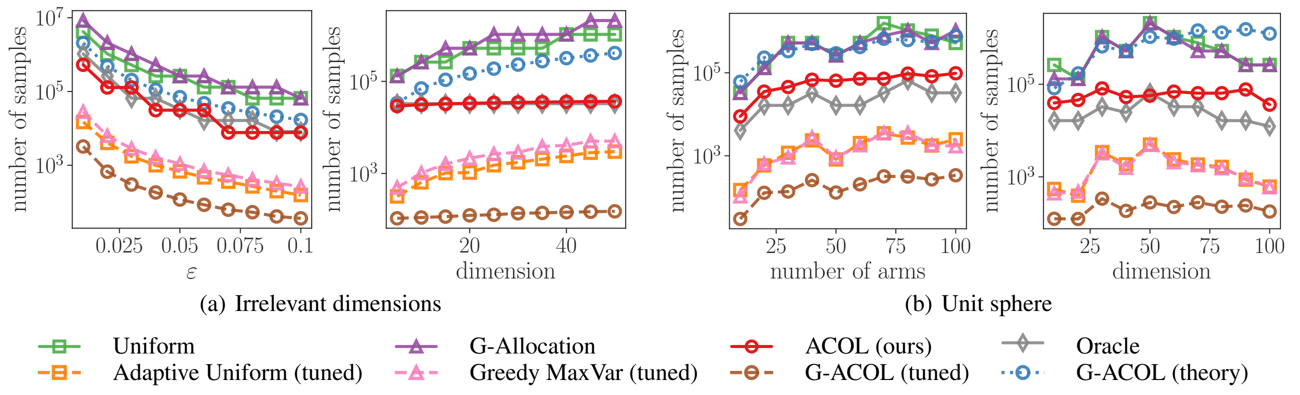


Figure 2. For both synthetic instances, we plot the median number of iterations for finding the constrained optimal solution as a function of different parameters of the problem instance. All methods return the correct constrained optimal solution. For "irrelevant dimensions", we vary ε for fixed $d = 10$, and d for fixed $\varepsilon = 0.05$. For "unit sphere", we vary n for fixed $d = 30$, and d for fixed $n = 30$. Note that for "unit sphere", the instances are randomly sampled for each random seed, whereas the "irrelevant dimensions" instance stays the same. For legibility, we only show the median computed over 30 random seeds, and omit a few of the baselines we evaluated. For plots with all baselines that include confidence intervals, see Appendix D. Overall, ACOL is the most sample efficient approach of all algorithms that provide theoretical guarantees. In rare cases, it even needs fewer samples than the oracle. However, this is mostly an artifact of both algorithms using slightly different round lengths (cf. Appendix C). By "tuning" β_t, we can gain several orders of magnitude in sample efficiency at the cost of theoretical guarantees. G-ACOL remains the most sample efficient among these tuned approaches.

unit sphere. We also sample the reward parameter θ from the unit sphere. As constraint parameter, we choose $\phi = x_i - x_j$ where x_i and x_j are the two closest arms in ℓ_2-distance. We can increase the problem difficulty by increasing the dimension d and the number of arms n.

Baselines. We compare ACOL and G-ACOL to various baselines. The *Oracle* solution uses knowledge of the true constraint parameter to choose the best possible static allocation (cf. Appendix B.1). In practice, we cannot implement the oracle because we do not know the constraint parameter; but, it yields a performance upper bound to which we can compare other algorithms. *G-Allocation* uses a static allocation that uniformly reduces uncertainty (cf. Appendix B.2), whereas *Uniform* pulls all arms with equal probability. We also consider variants of these algorithms that use the adaptive confidence interval in Proposition 3. We call the adaptive version of G-Allocation *Greedy MaxVar* because it greedily selects arms with the highest uncertainty esimate from $\mathcal{U}_t$. We call uniform sampling with the adaptive confidence intervals *Adaptive Uniform* respectively. For all algorithms that use adaptive confidence intervals, in addition to the version using Proposition 3, we test a "tuned" version that considers β_t as a numeric hyperparameter instead (indicated by the name of the algorithms followed by *(tuned)*). We chose $\beta_t = \frac{1}{4}$, for all experiments, which we determined from minimal tuning on the "irrelevant dimensions" instance for the Greedy MaxVar algorithm. For clarity, we omit a few of the baselines that perform poorly in our plots. Appendix D provides the full results.

Results. Figure 2 shows our results in the synthetic CBAI instances. All algorithms find the correct solution, but their sample efficiency varies widely. From all algorithms with theoretical guarantees, the (unrealistic) oracle solution needs the fewest number of iterations, as expected. But ACOL can get close to the oracle performance and outperforms G-Allocation and uniform sampling in all cases. For example, if we increase the number of irrelevant dimensions in the first experiment, G-Allocation and uniform sampling need more samples to determine which dimension is relevant. In contrast, both ACOL quickly focuses on the relevant dimension. Therefore, the number of iterations it needs does not increase when adding irrelevant dimensions to the problem, similar to the oracle solution.

Methods that use adaptive confidence intervals with β_t suggested by Proposition 3 turn out to be less sample efficient than their round-based counterparts using static confidence intervals, including G-ACOL performing worse than ACOL. The reason for this is that the confidence interval in Proposition 3 is quite loose. We can heuristically choose smaller confidence intervals and consider β_t as a tunable hyperparameter. We find that we can achieve orders of magnitude better sample complexity without much tuning and still always find the correct solution. Even though this approach loses the theoretical guarantees, it could be very valuable in practical applications.

4.2. Comparing ACOL to Regret Minimization

To highlight the difference of our constrained linear best-arm identification setting to regret minimization with constraints, we perform an experiment to compare G-ACOL to the approaches by Amani et al. (2019) and Moradipari et al.

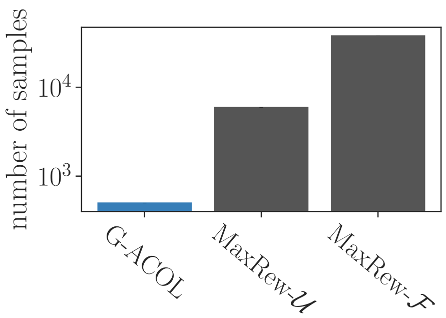


Figure 3. We compare G-ACOL to MaxRew-$\mathcal{F}$ and MaxRew-$\mathcal{U}$, that adapt regret minimization approaches to the CBAI setting. We focus on a simple 1-dimensional problem, where we ensure the set of feasible arms is connected. We find that MaxRew-$\mathcal{F}$ is particularly sample inefficient because it only selects arms that are certainly feasible. MaxRew-$\mathcal{U}$ is also less sample efficient than G-ACOL because it selects arms with high reward over other arms that would be more informative during exploration.

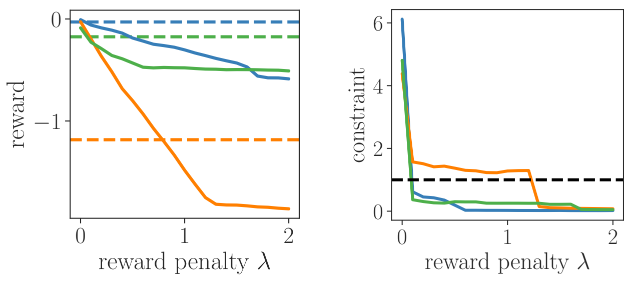


Figure 4. We quantify our finding that learning constraints is more robust to changes in the environment than learning a penalized reward function. We consider the three scenarios from Figure 1: the base scenario (—), a scenario with a different goal (—), and a scenario with a change in the environment (—). We find a policy that optimizes the reward function $\theta^T x - \lambda\phi^T x$ and plot the reward and the constraint of the solution for different values of λ. In particular, we need to choose a different value of λ for each environment to find the best solution with a constraint value below 1. The dashed horizontal lines in the reward plot show the reward a constrained solution obtains on the corresponding instance, which does not require any tuning. For each scenario, the smallest λ we find to yield a feasible solution still gives a worse solution in terms of reward than the constrained solution.

(2021). The algorithm by Amani et al. (2019) performs UCB and the algorithm by Moradipari et al. (2021) performs Thompson sampling, both within the set of certainly feasible arms.

We can translate both approaches to our setting with known rewards by greedily selecting arms from $\mathcal{F}_t$ w.r.t. their reward. Because we do not start with a known safe arm, we add an additional phase in which we select arms randomly until $\mathcal{F}_t$ is not empty. Let us call this approach *MaxRew-$\mathcal{F}$*. As a hybrid of this approach and ACOL, we can design an algorithm that greedily select arms from $\mathcal{U}_t$ w.r.t. their reward. Let us call this algorithm *MaxRew-$\mathcal{U}$*.

Unfortunately, MaxRew-$\mathcal{F}$ gets stuck in our synthetic instances because we do not make any assumptions on the safe set such as convexity and compactness. To evaluate these algorithms, we, therefore, consider a third synthetic instance in which the safe set is connected. We consider 10 arms in $d = 1$ that are equally spaced between 0 and 1. The reward and constraint vectors are $\theta = \phi = 1$, and the threshold is $\tau = 0.25$. Here the safe set is connected, but we can learn the constraint boundary more efficiently if we are allowed to violate the constraint during exploration.

We compare G-ACOL to MaxRew-$\mathcal{F}$ and MaxRew-$\mathcal{U}$ in Figure 3. We find that G-ACOL explores much more efficiently than both of the other approaches. MaxRew-$\mathcal{F}$ is particularly sample inefficient, because it ensures feasibility during exploration, which is not necessary in our case. In Appendix D, we provide results for MaxRew-$\mathcal{U}$ in all of our environments. We cannot provide these results for MaxRew-$\mathcal{F}$ because it gets stuck in all other environments.

4.3. Preference Learning Experiments

We now consider the application that initially motivated us to define the CBAI problem. As discussed in Section 1, we are interested in situations where the reward parameter θ describes an easy-to-specify goal or metric, and the constraint parameter ϕ describes expensive-to-evaluate human preferences.

As an example of this, we consider a driving simulator, which Sadigh et al. (2017) originally introduced to study learning reward functions to represent human preferences about driving behavior. Instead, we change the setting to have the reward θ represent an easy-to-specify goal such as "drive at velocity v", and the constraint ϕ represent other driving rules such as "usually drive in a lane" or "don't get too close to other cars", as shown in Figure 1. Appendix C provides more details on the environment.

The decision-maker has to select a controller to drive the car from a set of precomputed controllers $\mathcal{X}$, i.e., the set of "arms". The optimal controller x^* maximizes $\theta^T x^*$ and satisfies $\phi^T x^* \leq \tau$. The decision-maker can try out individual controllers to get feedback on whether they are feasible. In contrast to our previous experiments, the feedback is binary. However, we can still model it via a sub-Gaussian noise model by ensuring the constraint values are in $[0, 1]$ and interpreting them as probabilities. Therefore, this is a CBAI problem, and we can apply the same algorithms we applied to our synthetic problems.

Robustness of learning constraints. First, we want to quantify the observation of Figure 1 that constraints can be

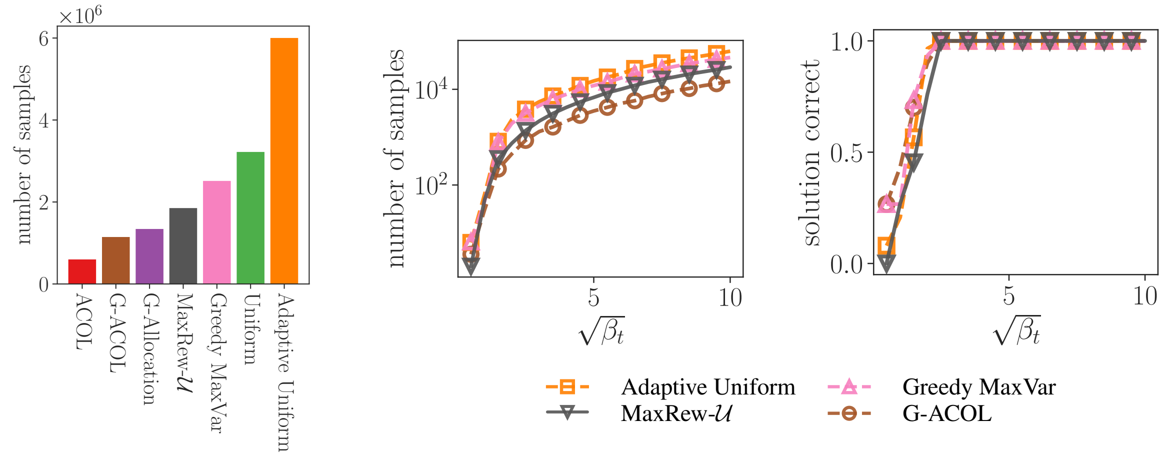


Figure 5. The left chart shows the number of iterations that all algorithms with theoretical guarantees need to find the correct solution in the driving scenario. ACOL is the fastest, but it still needs $\sim 10^5$ samples. Instead, we can use heuristic confidence intervals where we consider β_t as a hyperparameter instead of choosing the values suggested by theory. The two right plots shows the number of iterations and the percentage of the times the methods return a correct solution as a function of β_t. None of these algorithms is guaranteed to return the correct solution. But, empirically, we find that for $\sqrt{\beta_t}$ beyond the vertical line, the algorithms always return the correct solution. This again shows that tuning β_t can drastically improve the sample efficiency while still returning the correct solution empirically.

a particularly robust representation of human preferences. Specifically, using constraints to represent human preferences can increase robustness to changes in the environment and allow to transfer the constraints to different reward functions. Constraints are more robust than modeling the same preferences as a penalty on the reward function. Figure 4 quantifies this by directly comparing the two options in terms of the reward and constraint values they achieve. In particular, we find that the magnitude of the reward penalty often has to be updated if the environment changes, whereas the constraint formulation is robust to such changes.

Results of learning constraints. We consider the driving scenario as a CBAI problem and study learning the constraint function. Here, we only report results for the base scenario in Figure 1. Appendix D contains similar results for the other two scenarios which are qualitatively similar. In Figure 5, we compare the performance of ACOL and other algorithms with theoretical correctness guarantees to versions of these algorithms with heuristic confidence intervals. In both cases ACOL or G-ACOL is the most sample efficient algorithm. By choosing the heuristic confidence intervals, we can reduce the number of samples necessary by two orders of magnitude from $\sim 10^5$ to $\sim 10^3$, at the cost of theoretical guarantees. In all cases, using ACOL is preferable over alternatives because it finds the correct solution with fewer queries about the constraint function.

5. Conclusion

It is natural to formalize sequential decision-making problems in many practical situations as optimizing a known reward function subject to unknown, expensive-to-evaluate constraints. We studied constrained linear best-arm identification (CBAI), a linear bandit setting to learn about constraints efficiently, and proposed Adaptive Constraint Learning (ACOL) to efficiently solve this problem.

Limitations and future work. Our theoretical analysis is limited to a single constraint function, which might not be appropriate for applications where the constraints are non-additive. It should be possible to extend the same theoretical ideas to multiple linear constraints that all have to be satisfied, which would allow to apply ACOL to such situations. From the empirical perspective, we found that modelling human preferences as constraints rather than rewards can be more robust. Future work should study using constraints to model human preferences in more practical applications.

Broader impact. Sample efficient methods to learn about human preferences could help to avoid misspecified objectives in ML (Amodei et al., 2016). By focusing on learning constraints, it might be possible to make preference learning more robust and interpretable. Of course, such algorithms could be misused, but we are optimistic that robust methods to learn from humans will lead to safer ML methods overall.

Acknowledgements

This project has received funding from the Microsoft Swiss Joint Research Center (Swiss JRC), and from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement No 815943. We thank Ilija Bogunovic and Alexandru Țifrea for valuable feedback on early drafts of this paper.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, 2011.

Amani, S., Alizadeh, M., and Thrampoulidis, C. Linear stochastic bandits under safety constraints. In *Advances in Neural Information Processing Systems*, 2019.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in AI safety. *arXiv:1606.06565*, 2016.

Audibert, J.-Y., Bubeck, S., and Munos, R. Best arm identification in multi-armed bandits. In *Conference on Learning Theory (COLT)*, 2010.

Bıyık, E., Palan, M., Landolfi, N. C., Losey, D. P., and Sadigh, D. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning (CoRL)*, 2020.

Daniel, C., Viering, M., Metz, J., Kroemer, O., and Peters, J. Active reward learning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2014.

Fiez, T., Jain, L., Jamieson, K. G., and Ratliff, L. Sequential experimental design for transductive linear bandits. In *Advances in Neural Information Processing Systems*, 2019.

Friedrich, P. *Optimal design of experiments*. Siam. Society for industrial and applied mathematics, 2006.

Gardner, J. R., Kusner, M. J., Xu, Z. E., Weinberger, K. Q., and Cunningham, J. P. Bayesian optimization with inequality constraints. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014.

Gelbart, M. A., Snoek, J., and Adams, R. P. Bayesian optimization with unknown constraints. In *Uncertainty in Artificial Intelligence (UAI)*, 2014.

Hernández-Lobato, J. M., Gelbart, M. A., Adams, R. P., Hoffman, M. W., and Ghahramani, Z. A general framework for constrained Bayesian optimization using information-based search. *Journal of Machine Learning Research*, 17, 2016.

Kano, H., Honda, J., Sakamaki, K., Matsuura, K., Nakamura, A., and Sugiyama, M. Good arm identification via bandit feedback. *Machine Learning*, 108, 2019.

Kaufmann, E., Cappé, O., and Garivier, A. On the complexity of best-arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 17, 2016.

Kazerouni, A., Ghavamzadeh, M., Abbasi Yadkori, Y., and Van Roy, B. Conservative contextual linear bandits. In *Advances in Neural Information Processing Systems*, 2017.

Khezeli, K. and Bitar, E. Safe linear stochastic bandits. In *AAAI Conference on Artificial Intelligence*, 2020.

Kiefer, J. and Wolfowitz, J. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12, 1960.

Lattimore, T. and Szepesvári, C. *Bandit algorithms*. Cambridge University Press, 2020.

Locatelli, A., Gutzeit, M., and Carpentier, A. An optimal algorithm for the thresholding bandit problem. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016.

Moradipari, A., Amani, S., Alizadeh, M., and Thrampoulidis, C. Safe linear Thompson sampling with side information. *IEEE Transactions on Signal Processing*, 2021.

Pacchiano, A., Ghavamzadeh, M., Bartlett, P., and Jiang, H. Stochastic bandits with linear constraints. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

Perrone, V., Shcherbatyi, I., Jenatton, R., Archambeau, C., and Seeger, M. Constrained bayesian optimization with max-value entropy search. In *NeurIPS 2019 Workshop on Metalearning*, 2019.

Rubinstein, R. Y. and Kroese, D. P. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer, 2004.

Sadigh, D., Dragan, A. D., Sastry, S., and Seshia, S. A. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

Settles, B. *Active learning*. Morgan & Claypool Publishers, 2012.

Soare, M. *Sequential resource allocation in linear stochastic bandits*. PhD thesis, Université Lille 1-Sciences et Technologies, 2015.

Soare, M., Lazaric, A., and Munos, R. Best-arm identification in linear bandits. In *Advances in Neural Information Processing Systems*, 2014.

Sui, Y., Gotovos, A., Burdick, J., and Krause, A. Safe exploration for optimization with Gaussian processes. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015.

Wang, Z., Wagenmaker, A., and Jamieson, K. Best arm identification with safety constraints. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Wen, M. and Topcu, U. Constrained cross-entropy method for safe reinforcement learning. *IEEE Transactions on Automatic Control*, 2020.

A. Proofs

This section provides the full proofs of our key results of the paper: the sample complexity lower bound for CBAI problems (Appendix A.1) and the sample complexity of ACOL (Appendix A.2).

A.1. Lower Bounds

Theorem 1 (CBAI lower bound)**.** *Assume $\eta_x \sim \mathcal{N}(0,1)$ for all $x \in \mathcal{X}$. For any CBAI problem $\nu = (\mathcal{X}, \theta, \phi)$, there exists another CBAI problem $\nu' = (\mathcal{X}, \theta, \phi')$ with the same set of actions $\mathcal{X}$ and reward parameter θ but a different constraint parameter and optimal arm, such that the expected number of iterations τ needed by any allocation strategy that can distinguish between ν and ν' with probability at least $1-\delta$ is lower bounded as*

$$\mathbb{E}[\tau] \geq 2\log\left(\frac{1}{2.4\delta}\right) \max_{x \in \mathcal{X}_\theta^{\geq}(x_\nu^*)} \frac{\|x\|^2_{A_\lambda^{-1}}}{(\phi^T x)^2},$$

where λ is a probability distribution over arms which the allocation strategy follows, i.e., $\lambda(x)$ is the probability that it pulls arm x, $A_\lambda = \sum_x \lambda(x) x x^T$ is the design matrix, $\mathcal{X}_\theta^{\geq}(x_\nu^*) = \{x' \in \mathcal{X} | \theta^T x' \geq \theta^T x_\nu^*\}$ is the set of all arms with reward no less than x_ν^*, the optimal arm for problem ν.

Proof. Our proof has a similar structure to the proof of Theorem 3.1 by Soare (2015). Let us denote the optimal arm of problem ν with x_ν^* and the optimal arm of ν' with $x_{\nu'}^*$. Let $\mathcal{A}$ be a δ-PAC algorithm to solve constrained linear bandit problems, and let A be the event that $\mathcal{A}$ recommends x_ν^* as the optimal arm. If we denote by $P_\nu(A)$ the probability of A happening for instance ν, and by $P_{\nu'}(A)$ the probability for instance ν', we have $P_\nu(A) \geq 1-\delta$ and $P_{\nu'}(A) \leq \delta$.

Let $\tilde{\varepsilon} = \phi' - \phi$, and let τ be the stopping time of $\mathcal{A}$. Let $(x_1, \dots, x_\tau)$ be the sequence of arms $\mathcal{A}$ pulls and $(z_1, \dots, z_t)$ the corresponding observed noisy constraint values $z_i = x_i^T \phi + \eta_{x_i}$ with $\eta_{x_i} \sim \mathcal{N}(0,1)$ being independent Gaussian noise.

Now, consider the log-likelihood ratio of these observations under algorithm $\mathcal{A}$:

$$\begin{aligned}
L_\tau &= \log\left(\prod_{s=1}^{\tau} \frac{P_\nu(z_s|x_s)}{P_{\nu'}(z_s|x_s)}\right) = \sum_{s=1}^{\tau} \log\left(\frac{P_\nu(z_s|x_s)}{P_{\nu'}(z_s|x_s)}\right) = \sum_{s=1}^{\tau} \log\left(\frac{P_\nu(\eta_s)}{P_{\nu'}(\eta'_s)}\right) = \sum_{s=1}^{\tau} \log\left(\frac{\exp(-\eta_s^2/2)}{\exp(-\eta_s'^2/2)}\right) \\
&= \sum_{s=1}^{\tau} \frac{1}{2}((z_s - x_s^T\phi')^2 - (z_s - x_s^T\phi)^2) = \sum_{s=1}^{\tau} \frac{1}{2}(z_s^2 - 2z_s x_s^T\phi' + (x_s^T\phi')^2 - z_s^2 + 2z_s x_s^T\phi - (x_s^T\phi)^2) \\
&= \sum_{s=1}^{\tau} \frac{1}{2}(2z_s x_s^T(\phi - \phi') + (x_s^T\phi' - x_s^T\phi)(x_s^T\phi' + x_s^T\phi)) = \sum_{s=1}^{\tau} \frac{1}{2}(-2z_s x_s^T\tilde{\varepsilon} + x_s^T\tilde{\varepsilon}(x_s^T\phi + x_s^T\tilde{\varepsilon} + x_s^T\phi)) \\
&= \sum_{s=1}^{\tau} (x_s^T\tilde{\varepsilon}) \frac{-2z_s + 2x_s^T\phi + x_s^T\tilde{\varepsilon}}{2} = \sum_{s=1}^{\tau} (x_s^T\tilde{\varepsilon}) \left(\frac{x_s^T\tilde{\varepsilon}}{2} - \eta_s\right)
\end{aligned}$$

Taking the expectation of this log-likelihood ratio gives:

$$\begin{aligned}
\mathbb{E}_\nu[L_\tau] &= \mathbb{E}_\nu\left[\sum_{s=1}^{\tau} (x_s^T\tilde{\varepsilon}) \left(\frac{x_s^T\tilde{\varepsilon}}{2} - \eta_s\right)\right] = \frac{1}{2}\mathbb{E}_\nu\left[\sum_{s=1}^{\tau} (x_s^T\tilde{\varepsilon})^2\right] - \underbrace{\mathbb{E}_\nu[\eta_s]}_{=0} \\
&= \frac{1}{2}\mathbb{E}_\nu\left[\sum_{s=1}^{\tau} \tilde{\varepsilon}^T x_s x_s^T \tilde{\varepsilon}\right] = \frac{1}{2}\mathbb{E}_\nu\left[\sum_{x \in \mathcal{X}} \mathbb{E}_\nu[\tau]\lambda(x)\tilde{\varepsilon}^T x x^T \tilde{\varepsilon}\right] \\
&= \frac{1}{2}\mathbb{E}_\nu[\tau]\mathbb{E}_\nu\left[\sum_{x \in \mathcal{X}} \lambda(x)\tilde{\varepsilon}^T x x^T \tilde{\varepsilon}\right] = \frac{1}{2}\mathbb{E}_\nu[\tau]\tilde{\varepsilon}^T A_\lambda \tilde{\varepsilon}
\end{aligned}$$

Next, we can apply Lemma 19 from Kaufmann et al. (2016):

$$\mathbb{E}_\nu[L_\tau] = \frac{1}{2}\mathbb{E}_\nu[\tau]\tilde{\varepsilon}^T A_\lambda \tilde{\varepsilon} \geq \mathrm{KL}(P_\nu(A), P_{\nu'}(A)) \geq \log\frac{1}{2.4\delta}$$

$$\mathbb{E}_\nu[\tau] \geq 2\log\left(\frac{1}{2.4\delta}\right)\frac{1}{\tilde{\varepsilon}^T A_\lambda \tilde{\varepsilon}} \tag{1}$$

To obtain a lower bound, we now aim to find the smallest $\tilde{\varepsilon}$ such that ν and ν' have different constrained optimal arms.

Let $\mathcal{X}_\theta^{\geq}(x) = \{x' \in \mathcal{X} | \theta^T x' \geq \theta^T x\}$ be the set of arms with higher reward than x. There are two ways we can modify ν to change its optimal arm. We can change ϕ to ϕ' such that either, **Case (i)**, the previous optimum x_ν^* becomes infeasible in ν', or, **Case (ii)**, a solution $x_\nu^* \in \mathcal{X}_\theta^{\geq}(x_\nu^*)$ that was infeasible in ν is now feasible in ν'. We will consider both cases separately, and aim to find an $\tilde{\varepsilon}$ for each case that minimizes $\tilde{\varepsilon}^T A_\lambda \tilde{\varepsilon}$.

Case (i). We want to find $\tilde{\varepsilon}$ that minimizes $\frac{1}{2}\varepsilon^T A_\lambda \varepsilon$ such that $\phi'^T x_\nu^* > 0$, i.e., the previously optimal arm becomes infeasible. We can write this constraint equivalently as

$$\phi'^T x_\nu^* > 0 \Leftrightarrow \phi^T x_\nu^* - \phi'^T x_\nu^* < \phi^T x_\nu^* \Leftrightarrow \varepsilon^T x_\nu^* < \phi^T x_\nu^* \Leftrightarrow \varepsilon^T x_\nu^* - \phi^T x_\nu^* < 0$$

Which results in the following optimization problem:

$$\min_\varepsilon \frac{1}{2}\varepsilon^T A_\lambda \varepsilon \quad \text{s.t.} \quad \varepsilon^T x_\nu^* - \phi^T x_\nu^* + \alpha \leq 0,$$

where $\alpha > 0$. The Lagrangian is $L(\varepsilon, \gamma) = \frac{1}{2}\varepsilon^T A_\lambda \varepsilon - \gamma(\varepsilon^T x_\nu^* - \phi^T x_\nu^* + \alpha)$, and requiring $\frac{\partial L}{\partial \varepsilon} = \frac{\partial L}{\partial \gamma} = 0$ yields:

$$\begin{aligned}
\frac{\partial L}{\partial \varepsilon} &= A_\lambda \varepsilon - \gamma x_\nu^* = 0 \Leftrightarrow A_\lambda \varepsilon = \gamma x_\nu^* \Leftrightarrow A_\lambda^{\frac{1}{2}}\varepsilon = \gamma A_\lambda^{-\frac{1}{2}} x_\nu^* \\
\frac{\partial L}{\partial \gamma} &= \varepsilon^T x_\nu^* - \phi^T x_\nu^* + \alpha = 0 \Leftrightarrow \varepsilon^T x_\nu^* = \phi^T x_\nu^* - \alpha
\end{aligned}$$

From the first equation, it follows that

$$\begin{aligned}
{x_\nu^*}^T \varepsilon &= {x_\nu^*}^T A_\lambda^{-\frac{1}{2}} A_\lambda^{\frac{1}{2}} \varepsilon = \gamma {x_\nu^*}^T A_\lambda^{-1} x_\nu^* = \gamma \|x_\nu^*\|^2_{A_\lambda^{-1}} \\
{x_\nu^*}^T \varepsilon &= {x_\nu^*}^T A_\lambda^{-\frac{1}{2}} A_\lambda^{\frac{1}{2}} \varepsilon = \frac{1}{\gamma}\varepsilon^T A_\lambda \varepsilon = \frac{1}{\gamma}\|\varepsilon\|^2_{A_\lambda}
\end{aligned}$$

and therefore

$$\begin{aligned}
{x_\nu^*}^T \varepsilon &= \|x_\nu^*\|_{A_\lambda^{-1}} \|\varepsilon\|_{A_\lambda} = \phi^T x_\nu^* - \alpha \\
\|\varepsilon\|_{A_\lambda} &= \frac{\phi^T x_\nu^* - \alpha}{\|x_\nu^*\|_{A_\lambda^{-1}}} > \frac{\phi^T x_\nu^*}{\|x_\nu^*\|_{A_\lambda^{-1}}}
\end{aligned}$$

where the last inequality follows because $\alpha > 0$ and A_λ is positive definite.

Case (ii). We want to find $\tilde{\varepsilon}$ that minimizes $\frac{1}{2}\varepsilon^T A_\lambda \varepsilon$ such that there exists an $x \in \mathcal{X}$ for which $\theta^T x > \theta^T x_\nu^*$ and $\phi'^T x \leq 0$, i.e., x has higher reward than x_ν^* and it is feasible in ν'. We can write these constraints as

$$\begin{aligned}
\theta^T x > \theta^T x_\nu^* &\Leftrightarrow \theta^T (x_\nu^* - x) + \alpha \leq 0 \\
\phi'^T x \leq 0 &\Leftrightarrow \varepsilon^T x + \phi^T x \leq 0
\end{aligned}$$

with $\alpha > 0$. This results in the following optimization problem:

$$\begin{aligned}
\min_\varepsilon \quad & \frac{1}{2}\varepsilon^T A_\lambda \varepsilon \\
\text{s.t.} \quad \exists x : \; & \theta^T(x_\nu^* - x) + \alpha && \leq 0 \\
& \varepsilon^T x + \phi^T x && \leq 0
\end{aligned}$$

The Lagrangian of this problem is

$$L(\varepsilon, \gamma, \delta) = \frac{1}{2}\varepsilon^T A_\lambda \varepsilon - \gamma(\theta^T(x_\nu^* - x) + \alpha) - \delta(\varepsilon^T x + \phi^T x)$$

Requiring $\frac{\partial L}{\partial \varepsilon} = \frac{\partial L}{\partial \delta} = 0$ results in

$$\frac{\partial L}{\partial \varepsilon} = A_\lambda \varepsilon - \delta x = 0 \Leftrightarrow A_\lambda \varepsilon = \delta x \Leftrightarrow A_\lambda^{\frac{1}{2}} \varepsilon = \delta A_\lambda^{-\frac{1}{2}} x$$
$$\frac{\partial L}{\partial \delta} = \varepsilon^T x + \phi^T x = 0 \Leftrightarrow \varepsilon^T x = -\phi^T x$$

It follows that

$$x^T \varepsilon = x^T A_\lambda^{-\frac{1}{2}} A_\lambda^{\frac{1}{2}} \varepsilon = \delta x^T A_\lambda^{-1} x = \delta \|x\|^2_{A_\lambda^{-1}}$$
$$x^T \varepsilon = x^T A_\lambda^{-\frac{1}{2}} A_\lambda^{\frac{1}{2}} \varepsilon = \frac{1}{\delta} \varepsilon^T A_\lambda \varepsilon = \frac{1}{\delta} \|\varepsilon\|^2_{A_\lambda}$$

and therefore

$$x^T \varepsilon = \|x\|_{A_\lambda^{-1}} \|\varepsilon\|_{A_\lambda} = \phi^T x \;\Rightarrow\; \|\varepsilon\|_{A_\lambda} = \frac{\phi^T x}{\|x\|_{A_\lambda^{-1}}}$$

Combining this result with the remaining constraint $\theta^T x > \theta^T x^*_\nu$ which implies $x \in \mathcal{X}^{\geq}_\theta(x^*_\nu)$, we can conclude

$$\|\varepsilon\|_{A_\lambda} \geq \min_{x \in \mathcal{X}^{\geq}_\theta(x^*_\nu)} \frac{\phi^T x}{\|x\|_{A_\lambda^{-1}}}$$

Combining cases (i) and (ii). We can conclude that the ε that minimizes $\|\varepsilon\|^2_{A_\lambda}$ while still ensuring that ν' has a different solution than ν, satisfies:

$$\|\varepsilon\|_{A_\lambda} \geq \min \Big[\underbrace{\frac{\phi^T x^*_\nu}{\|x^*_\nu\|_{A_\lambda^{-1}}}}_{\text{Case (i)}}, \underbrace{\min_{x \in \mathcal{X}^{\geq}_\theta(x^*_\nu)} \frac{\phi^T x}{\|x\|_{A_\lambda^{-1}}}}_{\text{Case (ii)}} \Big]$$

But because $x^*_\nu \in \mathcal{X}^{\geq}_\theta(x^*_\nu)$, it is simply

$$\|\varepsilon\|_{A_\lambda} \geq \min_{x \in \mathcal{X}^{\geq}_\theta(x^*_\nu)} \frac{\phi^T x}{\|x\|_{A_\lambda^{-1}}}$$

Combining this result with eq. (1), gives the final bound:

$$\mathbb{E}_\nu[\tau] \geq 2 \log\left(\frac{1}{2.4\delta}\right) \frac{1}{\left(\min_{x \in \mathcal{X}^{\geq}_\theta(x^*_\nu)} \frac{\phi^T x}{\|x\|_{A_\lambda^{-1}}}\right)^2} = 2 \log\left(\frac{1}{2.4\delta}\right) \max_{x \in \mathcal{X}^{\geq}_\theta(x^*_\nu)} \frac{\|x\|^2_{A_\lambda^{-1}}}{(\phi^T x)^2}$$

□

Next, we derive the worst case bound on the quantity making up the CBAI lower bound.

Proposition 1. *For any CBAI problem ν, we have $H_{\text{CLB}}(\nu) \leq d/(C^+_{\min})^2$, where $C^+_{\min} = \min_{x \in \mathcal{X}} |\phi^T x|$. This bound is tight, i.e, there is an instance ν, such that we have $H_{\text{CLB}}(\nu) = d/(C^+_{\min})^2$.*

Proof.

$$H_{\text{CLB}}(\nu) = \min_\lambda \max_{x \in \mathcal{X}^{\geq}_\theta(x^*_\nu)} \frac{\|x\|^2_{A_{\lambda^*}^{-1}}}{(\phi^T x)^2} \leq \frac{1}{{C^+_{\min}}^2} \min_\lambda \max_{x \in \mathcal{X}^{\geq}_\theta(x^*_\nu)} \|x\|^2_{A_{\lambda^*}^{-1}} \leq \frac{d}{{C^+_{\min}}^2}$$

where the last inequality uses the well-known result by Kiefer & Wolfowitz (1960). Equality holds, for example, if all $x \in \mathcal{X}$ are linearly independent and have the same constraint value $C^+_{\min}$. □

A.2. Adaptive Constraint Learning

In this section, we analyse the sample complexity of ACOL and prove our main result.

Theorem 2 (ACOL sample complexity)**.** *Assume Algorithm 1 is implemented with an ε-approximate rounding strategy. Then, after N iterations the algorithm returns an optimal arm with probability at least $1-\delta$, and we have:*

$$N \le 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\sum_{t=1}^{\bar{t}} \min_{\lambda}\max_{x\in\mathcal{U}_t} \frac{\|x\|^2_{A_\lambda^{-1}}}{(\phi^T x)^2} + \bar{t}$$
$$\le 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\bar{t}\bar{H}_{\mathrm{CLB}}(\nu) + \bar{t}$$

where $\bar{H}_{\mathrm{CLB}}(\nu) = \min_\lambda \max_{x\in\mathcal{X}} \|x\|^2_{A_\lambda^{-1}}/(\phi^T x)^2$, and $\bar{t} = \lceil -\log_2 C^+_{\min}\rceil$. Moreover, $\bar{H}_{\mathrm{CLB}}(\nu) \le d/(C^+_{\min})^2$

Proof. Let $\mathcal{E}_t \coloneqq \{\mathcal{U}_t \subseteq \mathcal{S}_t\}$ where $\mathcal{S}_t \coloneqq \{x\in\mathcal{X} | u^t_\phi(x) - l^t_\phi(x) \le 2^{-t}\}$. So, $\mathcal{E}_t$ is the event that all arms in $\mathcal{U}_t$ have confidence interval smaller than 2^{-t}. We will first show that $P(\mathcal{E}_1)\ge 1-\delta_1$ and $P(\mathcal{E}_t|\mathcal{E}_{t-1}) \ge 1-\delta_t$, which ensures that the set of arms we are uncertain about shrinks exponentially in the rounds t.

Let $x\in\mathcal{U}_t$. Then, using Proposition 2, and the ε-approximate rounding strategy, it holds with probability at least $1-\delta_t$ that:

$$u^t_\phi(x) - l^t_\phi(x) \le 2\sqrt{2\log\left(\frac{|\mathcal{X}|}{\delta_t}\right)\frac{1+\varepsilon}{N_t}}\|x\|_{A^{-1}_{\lambda_t^*}}$$

Using the length of a round $N_t = \left\lceil 2^{2t+3}\log\left(\frac{|\mathcal{X}|}{\delta_t}\right)(1+\varepsilon)\rho^*_t\right\rceil$, and that we select arms to reduce uncertainty in $\mathcal{U}_t$, we get

$$u^t_\phi(x) - l^t_\phi(x) \le 2^{-t}\sqrt{\left(\min_\lambda \max_{\tilde{x}\in\mathcal{U}_t}\|\tilde{x}\|^2_{A_\lambda^{-1}}\right)^{-1}}\|x\|_{A^{-1}_{\lambda_t^*}}$$
$$\le 2^{-t}\sqrt{\left(\min_\lambda \max_{\tilde{x}\in\mathcal{U}_t}\|\tilde{x}\|^2_{A_\lambda^{-1}}\right)^{-1}}\left(\min_\lambda \max_{\tilde{x}\in\mathcal{U}_t}\|\tilde{x}\|_{A_\lambda^{-1}}\right) \le 2^{-t}$$

Note, that x can only be in $\mathcal{U}_t$ if $u^t_\phi(x) > 0$ and $l^t_\phi(x)\le 0$. It follows that $P(\mathcal{E}_t|\mathcal{E}_{t-1}) \ge 1-\delta_t$.

Now consider round $\bar{t} \coloneqq \left\lceil \log_2 \frac{1}{C^+_{\min}}\right\rceil$. We show $P(\mathcal{U}_{\bar{t}} = \varnothing | \mathcal{E}_{\bar{t}}) = 1$. Assume $\mathcal{E}_{\bar{t}}$, i.e., $\mathcal{U}_t \subseteq \mathcal{S}_t$. Let $x\in\mathcal{U}_{\bar{t}}$, then:

$$|\phi^T x| \le 2^{-\bar{t}} \le 2^{-\log_2 1/C^+_{\min}} = C^+_{\min}$$

which is a contradiction because otherwise x would have a smaller constraint value than $C^+_{\min}$. Consequently, the set of uncertain arms $\mathcal{U}_{\bar{t}}$ is empty and the algorithm returns the correct solution given $\mathcal{E}_{\bar{t}}$. Lemma 1 shows that the unconditional probability of the algorithm returning the correct solution after round $\bar{t}$ is at least $1-\delta$.

Finally, we can compute the total number of samples the algorithm needs to return the correct solution:

$$N = \sum_{t=1}^{\bar{t}} \lceil 2^{2t+3}\log\left(\frac{|\mathcal{X}|}{\delta_t}\right)(1+\varepsilon)\rho^*_t\rceil \le \sum_{t=1}^{\bar{t}} 2^{2t+3}\log\left(\frac{|\mathcal{X}|}{\delta_t}\right)(1+\varepsilon)\rho^*_t + \bar{t}$$
$$\le 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\sum_{t=1}^{\bar{t}}(2^t)^2\rho^*_t + \bar{t} = 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\sum_{t=1}^{\bar{t}}(2^t)^2\min_\lambda\max_{\tilde{x}\in\mathcal{U}_t}\|\tilde{x}\|^2_{A_\lambda^{-1}} + \bar{t}$$
$$= 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\sum_{t=1}^{\bar{t}}\min_\lambda\max_{\tilde{x}\in\mathcal{U}_t}\frac{\|\tilde{x}\|^2_{A_\lambda^{-1}}}{(2^{-t})^2} + \bar{t} \overset{(a)}{\le} 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\sum_{t=1}^{\bar{t}}\min_\lambda\max_{\tilde{x}\in\mathcal{U}_t}\frac{\|\tilde{x}\|^2_{A_\lambda^{-1}}}{(\phi^T\tilde{x})^2} + \bar{t}$$
$$\overset{(b)}{\le} 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\sum_{t=1}^{\bar{t}}\min_\lambda\max_{\tilde{x}\in\mathcal{X}}\frac{\|\tilde{x}\|^2_{A_\lambda^{-1}}}{(\phi^T\tilde{x})^2} + \bar{t} \le 8\log\left(\frac{|\mathcal{X}|\bar{t}^2}{\delta^2}\right)(1+\varepsilon)\bar{t}\bar{H}_{\mathrm{CLB}}(\nu) + \bar{t}$$

Algorithm 3 Round based algorithm with a generic allocation λ^* with hyperparamater $v \in (1,2)$. For $\lambda^* \in \text{argmin}_\lambda \max_{x \in \mathcal{X}_\theta^{\geq}(x_\nu^*)} \|x\|_{A_\lambda^{-1}} / |\phi^T x|$ this algorithm becomes the oracle solution. For $\lambda^* \in \text{argmin}_\lambda \max_{x\in\mathcal{X}} \|x\|_{A_\lambda^{-1}}$ it becomes *G-Allocation*.

```
1: Input: static design $\lambda^*$, significance $\delta$
2: $\mathcal{U}_1 \leftarrow \mathcal{X}$                                   (uncertain arms)
3: $\mathcal{F}_1 \leftarrow \varnothing$                                   (feasible arms)
4: $t \leftarrow 1$                                                        (round)
5: while $\mathcal{U}_t \neq \varnothing$ do
6:     $\delta_t \leftarrow \delta^2/t^2$
7:     $N_t \leftarrow \lceil v^t \log(|\mathcal{X}|/\delta_t) \rceil$
8:     $\mathbf{x}_{N_t} \leftarrow \texttt{Round}(\lambda^*, N_t)$
9:     Pull arms $x_1, \dots, x_{N_t}$ and observe constraint values
10:    $t \leftarrow t+1$
11:    Update $\hat{\phi}_t$ and $A$ based on new data
12:    $l_\phi^t(x) \leftarrow \hat{\phi}_t^T x - \sqrt{\beta_t}\|x\|_{A^{-1}}$ for all arms $x \in \mathcal{X}$
13:    $u_\phi^t(x) \leftarrow \hat{\phi}_t^T x + \sqrt{\beta_t}\|x\|_{A^{-1}}$ for all arms $x \in \mathcal{X}$
14:    $\mathcal{F}_t \leftarrow \mathcal{F}_{t-1} \cup \{x | u_\phi^t(x) \leq 0\}$
15:    $\bar{r} \leftarrow \max_{x\in\mathcal{F}_t} \theta^T x$
16:    $\mathcal{U}_t \leftarrow \mathcal{U}_{t-1} \setminus \{x | l_\phi^t(x) > 0\} \setminus \{x | u_\phi^t(x) \leq 0\} \setminus \{x | \theta^T x < \bar{r}\}$
17: end while
18: return $x^* \in \text{argmax}_{x\in\mathcal{F}_t} \theta^T x$
```

where (a) follows because we showed that $|\phi^T x| \leq 2^{-t}$ w.h.p. for $x \in \mathcal{U}_t$, and (b) follows simply because $\mathcal{U}_t \subseteq \mathcal{X}$. In the last step, we defined $\bar{H}_{\text{CLB}}(\nu) = \min_\lambda \max_{\tilde{x}\in\mathcal{X}} \frac{\|\tilde{x}\|^2_{A_\lambda^{-1}}}{(\phi^T \tilde{x})^2}$

Moreover,

$$\bar{H}_{\text{CLB}}(\nu) = \min_\lambda \max_{x\in\mathcal{X}} \frac{\|x\|^2_{A_\lambda^{-1}}}{(\phi^T x)^2} \leq \frac{1}{C_{\min}^{+\;2}} \min_\lambda \max_{x\in\mathcal{X}} \|x\|^2_{A_\lambda^{-1}} \leq \frac{1}{C_{\min}^{+\;2}} \min_\lambda \max_{x\in\mathbb{R}^d} \|x\|^2_{A_\lambda^{-1}} \leq \frac{d}{C_{\min}^{+\;2}}$$

using the result by Kiefer & Wolfowitz (1960).

□

Lemma 1. *Let $\mathcal{E}_1, \dots, \mathcal{E}_T$ be a Markovian sequence of events such that $P(\mathcal{E}_1) \geq 1 - \delta_1$ and $P(\mathcal{E}_t|\mathcal{E}_{t-1}) \geq 1 - \delta_t$ for all $t = 2, \dots, T$, where $\delta_t = \delta^2/t^2$ and $\delta \in (0,1)$. $\mathcal{E}_t$ is independent of other events conditioned on $\mathcal{E}_{t-1}$. Then $P(\mathcal{E}_T) \geq 1 - \delta$.*

Proof.

$$P(\mathcal{E}_T) = \left(\prod_{t=2}^{T} P(\mathcal{E}_t|\mathcal{E}_{t-1})\right) P(\mathcal{E}_1) \geq \left(\prod_{t=2}^{\bar{t}} (1-\delta_t)\right)(1-\delta_1) \geq \prod_{t=1}^{\infty}\left(1 - \frac{\delta^2}{t^2}\right) = \frac{\sin(\pi\delta)}{\pi\delta} \geq 1 - \delta$$

where the last inequality holds for $0 \leq \delta \leq 1$. □

B. Alternative Algorithms

Given any static design λ^*, we can consider different round-based algorithms using the static confidence intervals from Proposition 2. Algorithm 3 shows the general algorithm. It uses the same stopping condition as ACOL but uses a more straightforward round length of $v^t \log(|\mathcal{X}|/\delta_t)$ with v a hyperparameter, and a fixed static allocation. In this section, we analyze two versions of this generic algorithm that are of particular interest: the *oracle* solution (Appendix B.1) and *G-Allocation* (Appendix B.2).

B.1. Oracle Solution

The oracle solution allocates samples according to $\lambda^* \in \operatorname{argmin}_\lambda \max_{x \in \mathcal{X}_\theta^{\geq}(x_\nu^*)} \|x\|_{A_\lambda^{-1}} / |\phi^T x|$ in Algorithm 3. Note that this design exactly matches the term in our instance dependent lower-bound in Theorem 1. Therefore, this is the ideal allocation to achieve good sample complexity. However, this oracle solution requires knowledge of ϕ, which we do not know in practice.

As expected, this algorithm matches the sample complexity lower bound, i.e., it is instance-optimal apart from logarithmic factors. The following theorem formalizes this.

Theorem 3 (Oracle sample complexity)**.** *The oracle algorithm finds the optimal solution to a constrained linear best-arm identification problem $\nu = (\mathcal{X}, \theta, \phi)$ within $N \propto H_{\mathrm{CLB}}(\nu)$ with probability at least $1 - \delta$.*

Proof. Assuming a $(1+\varepsilon)$-approximate rounding procedure, in round t we have: $\|x\|^2_{A^{-1}_{\mathbf{x}^*_{N_t}}} \leq \frac{1+\varepsilon}{N_t} \|x\|^2_{A^{-1}_{\lambda^*}}$. It follows, similar to the proof of Theorem 2, that in round t, for each $x \in \mathcal{X}_\theta^{\geq}$ if $\phi^T x > \phi^T x_\nu^*$:

$$\phi^T x - l_\phi^t(x) \leq \sqrt{2\log(|\mathcal{X}|/\delta_t)} \|x\|_{A^{-1}_{\mathbf{x}^*_n}} \leq \sqrt{2(1+\varepsilon)\log(|\mathcal{X}|/\delta_t)/N_t} \|x\|_{A^{-1}_{\lambda^*}}$$

A similar argument gives for x_ν^*:

$$u_\phi^t(x_\nu^*) - \phi^T x_\nu^* \leq \sqrt{2\log(|\mathcal{X}|/\delta_t)} \|x_\nu^*\|_{A^{-1}_{\mathbf{x}^*_n}} \leq \sqrt{2(1+\varepsilon)\log(|\mathcal{X}|/\delta_t)/N_t} \|x_\nu^*\|_{A^{-1}_{\lambda^*}}$$

Let us call the event that these confidence bounds hold $\mathcal{E}_t$. We have $P(\mathcal{E}_t | \mathcal{E}_{t-1}) \geq 1 - \delta_t$. Now, consider round $\bar{t} = \lceil \log_v (2(1+\varepsilon) H_{\mathrm{CLB}}(\nu)) \rceil$ with length $N_{\bar{t}} = \lceil 2(1+\varepsilon)\log(|\mathcal{X}|/\delta_t) H_{\mathrm{CLB}}(\nu) \rceil$. For all $x \in \mathcal{X}_\theta^{\geq}$ if $\phi^T x > \phi^T x_\nu^*$:

$$\phi^T x - l_\phi^{\bar{t}}(x) \leq \sqrt{\frac{1}{H_{\mathrm{CLB}}(\nu)}} \|x\|_{A^{-1}_{\lambda^*}} \leq \sqrt{\frac{(\phi^T x)^2}{\|x\|^2_{A^{-1}_{\lambda^*}}}} \|x\|_{A^{-1}_{\lambda^*}} \leq |\phi^T x|$$

Note that x is infeasible and $\phi^T x$, which implies $l_\phi^{\bar{t}}(x) \geq 0$ and in turn $x \notin \mathcal{U}_{\bar{t}}$. Similarly, $u_\phi^{\bar{t}}(x_\nu^*) - \phi^T x_\nu^* \leq |\phi^T x_\nu^*|$. x_ν^* is feasible and $\phi^T x_\nu^* \leq 0$. Hence, $u_\phi^{\bar{t}}(x_\nu^*) \leq 0$ and $x_\nu^* \notin \mathcal{U}_{\bar{t}}$. This implies that $\mathcal{U}_{\bar{t}} = \varnothing$ and, conditioned on $\mathcal{E}_{\bar{t}}$, the oracle algorithm solves the problem in round $\bar{t}$ with probability 1. We can apply Lemma 1 to conclude that, unconditionally, the algorithm solves the problem in round $\bar{t}$ with a probability of at least $1 - \delta$.

Let us compute the total iterations necessary:

$$N = \sum_{t=1}^{\bar{t}} \lceil 2(1+\varepsilon)\log(|\mathcal{X}|/\delta_t) H_{\mathrm{CLB}}(\nu) \rceil \leq \bar{t}(1 + 2(1+\varepsilon)\log(|\mathcal{X}|\bar{t}^2/\delta^2) H_{\mathrm{CLB}}(\nu)) \propto H_{\mathrm{CLB}}(\nu)$$

So, N is on order $H_{\mathrm{CLB}}(\nu)$ except for logarithmic factors, concluding the proof. □

B.2. G-Allocation

We obtain G-Allocation by choosing $\lambda^* \in \operatorname{argmin}_\lambda \max_{x \in \mathcal{X}} \|x\|_{A_\lambda^{-1}}$ in Algorithm 3. G-Allocation uniformly reduce the uncertainty about the constraint function for all arms. This is not ideal because it does not focus on which arms are plausible optimizers according to the known reward function.

Still, the following theorem shows that G-Allocation achieves sample complexity on order $d/{C^+_{\min}}^2$, so it matches the worst-case lower bound in Proposition 1.

Theorem 4. *G-Allocation finds the optimal arm within $N \propto d/{C^+_{\min}}^2$ iterations with probability at least $1 - \delta$.*

Proof. As in the proof of Theorem 3, we have in round t, for each $x \in \mathcal{X}_\theta^{\geq}$ if $\phi^T x > \phi^T x_\nu^*$:

$$u_\phi^t(x) - l_\phi^t(x) \leq 2\sqrt{2\log(|\mathcal{X}|/\delta_t)} \|x\|_{A^{-1}_{\mathbf{x}^*_n}} \leq 2\sqrt{2(1+\varepsilon)\log(|\mathcal{X}|/\delta_t)/N_t} \|x\|_{A^{-1}_{\lambda^*}}$$

Again, we call the event that these confidence bounds hold $\mathcal{E}_t$, and have $P(\mathcal{E}_t|\mathcal{E}_{t-1}) \geq 1 - \delta_t$. Now, consider round

$$\bar{t} = \log_v \left(8(1+\varepsilon) \operatorname*{argmin}_{\lambda} \max_{x \in \mathcal{X}} \|x\|_{A_\lambda^{-1}} / {C_{\min}^{+}}^2 \right)$$
$$N_{\bar{t}} = \left\lceil 8(1+\varepsilon) \log(|\mathcal{X}|/\delta_t) \operatorname*{argmin}_{\lambda} \max_{x \in \mathcal{X}} \|x\|_{A_\lambda^{-1}} / {C_{\min}^{+}}^2 \right\rceil$$

For all $x \in \mathcal{U}_t$ it follows that:

$$u_\phi^{\bar{t}}(x) - l_\phi^{\bar{t}}(x) \leq \sqrt{\frac{{C_{\min}^{+}}^2}{\|x\|_{A_{\lambda^*}^{-1}}}} \|x\|_{A_{\lambda^*}^{-1}} \leq C_{\min}^{+}$$

This implies that G-Allocation solves the problem in round $\bar{t}$ with probability 1, similar to the proof of Theorem 2. We can apply Lemma 1 to conclude that, unconditionally, the algorithm solves the problem in round $\bar{t}$ with a probability of at least $1 - \delta$.

Let us compute the total iterations necessary:

$$\begin{aligned} N &= \sum_{t=1}^{\bar{t}} \left\lceil 8(1+\varepsilon) \log(|\mathcal{X}|/\delta_t) \operatorname*{argmin}_{\lambda} \max_{x \in \mathcal{X}} \|x\|_{A_\lambda^{-1}} / {C_{\min}^{+}}^2 \right\rceil \\ &\leq \bar{t} \left(1 + 8(1+\varepsilon) \log(|\mathcal{X}|/\delta_t) \operatorname*{argmin}_{\lambda} \max_{x \in \mathcal{X}} \|x\|_{A_\lambda^{-1}} / {C_{\min}^{+}}^2 \right) \\ &\leq \bar{t} \left(1 + 8(1+\varepsilon) \log(|\mathcal{X}|/\delta_t) d / {C_{\min}^{+}}^2 \right) \propto d / {C_{\min}^{+}}^2 \end{aligned}$$

where the last inequality uses the result by Kiefer & Wolfowitz (1960). □

C. Experimental Details About the Driving Environment

This section provides details on the driving environment we use in Section 4.3. We provide full source code for all of our experiments at: `https://github.com/lasgroup/adaptive-constraint-learning`

We extend the *Driver* proposed by Sadigh et al. (2017) and Bıyık et al. (2020), to incorporate different tasks. Here, we provide a brief description of the dynamics and features of the environment.

The *Driver* environment uses point-mass dynamics with a continuous state and action space. The state $s = (x, y, \varphi, v)$ consists of the agent's position (x, y), its heading φ, and its velocity v. The actions $a = (a_1, a_2)$ consist of a steering input and an acceleration. The environment dynamics are given by

$$s_{t+1} = (x_{t+1}, y_{t+1}, \varphi_{t+1}, v_{t+1}) = (x_t + \Delta x, y_t + \Delta y, \varphi_t + \Delta\varphi, \text{clip}(v_t + \Delta v, -1, 1))$$
$$(\Delta x, \Delta y, \Delta\varphi, \Delta v) = (v \cos\varphi, v \sin\varphi, v a_1, a_2 - \alpha v)$$

where $\alpha = 1$ is a friction parameter, and the velocity is clipped to $[-1, 1]$ at each timestep.

The environment represents a highway with three lanes. In addition to the agent, the environment contains a second car that moves on a predefined trajectory. The reward and the constraint functions are linear in a set of features

$$f(s) = (f_1(s), f_2(s), f_3(s), f_4(s), f_5(s), f_6(s), f_7(s), f_8(s), 1)$$

that are described in detail in Table C.1.

The (known) rewards for the three scenarios are:

Base scenario: $\theta_1 = (1, 0, 0, 0, 0, 0, 0, 0, 0)$
Different reward: $\theta_2 = (0, 1, 0, 0, 0, 0, 0, 0, 0)$
Different environment: $\theta_3 = (1, 0, 0, 0, 0, 0, 0, 0, 0)$

Algorithm 4 Cross-entropy method for (constrained) reinforcement learning. For more details on the cross-entropy method, see Rubinstein & Kroese (2004), and for the application to constrained RL, see Wen & Topcu (2020).

```
1: **Input**: $n_{\text{iter}}$, $n_{\text{samp}}$, $n_{\text{elite}}$
2: Initialize policy parameters $\mu \in \mathbb{R}^d$, $\sigma \in \mathbb{R}^d$.
3: **for** iteration = $1, 2, \dots, n_{\text{iter}}$ **do**
4:     Sample $n_{\text{samp}}$ samples of $\omega_i \sim \mathcal{N}(\mu, \text{diag}(\sigma))$
5:     Evaluate policies $\omega_1, \dots, \omega_{n_{\text{samp}}}$ in the environment
6:     **if** constrained problem **then**
7:         Sort $\omega_i$ in ascending order of constraint value $J(\omega_i)$
8:         Let $E$ be the first $n_{\text{elite}}$ policies
9:         **if** $J(\omega_{n_{\text{elite}}}) \leq 0$ **then**
10:            Sort $\{\omega_i | J(\omega_i) \leq 0\}$ in descending order of return $G(\omega_i)$
11:            Let $E$ be the first $n_{\text{elite}}$ policies
12:        **end if**
13:    **else**
14:        Sort $\omega_i$ in descending order of return $G(\omega_i)$
15:        Let $E$ be the first $n_{\text{elite}}$ policies
16:    **end if**
17:    Fit Gaussian distribution with mean $\mu$ and diagonal covariance $\sigma$ to $E$
18: **end for**
19: **return** $\mu$
```

The (unknown) constraint is:

$$\phi = (0, 0, 0.3, 0.05, 0.02, 0.5, 0.3, 0.8), \quad \tau = 1$$

Our *Driver* environment uses a fixed time horizon $T = 20$, and policies are represented simply as sequences of 20 actions because the environment is deterministic.

C.1. Cross-Entropy Method for Constrained RL

We find policies in the Driver environment with a given reward function using the cross-entropy method (Rubinstein & Kroese, 2004). For the constrained reinforcement learning problem, we use a modified cross-entropy method, proposed by Wen & Topcu (2020), that takes the feasibility of solutions into account. Algorithm 4 contains pseudocode of this method.

C.2. Binary Feedback

So far, we considered numerical observations of the constraint value $\phi^T x + \eta$ where η is subgaussian noise. In the driving environment, we (more realistic) binary observations in $\{-1, 1\}$.

If we assume that all true constraint values are in $[-1, 1]$, we can define the observation model $P(y = 1|\phi, x) = (\phi^T x + 1)/2$. We can consider this as bounded, sub-gaussian noise on the constraint value, and so all our analysis still applies.

C.3. Setup

To translate learning the unknown constraint function in the Driver environment into a constrained linear best arm identification problem, we consider a set of pre-computed policies Π. This set of policies corresponds to the arms of a linear bandit problem, and both the return $G(\pi)$ of a policy and the constraint function $J(\pi)$ are linear in the expected feature counts of the policy: $G(\pi) = \mathbf{f}(\pi) \cdot \mathbf{r}$ and $J(\pi) = \mathbf{f}(\pi) \cdot \mathbf{c}$.

For binary observations, we normalize the features of all policies such that all constraint values are between -1 and 1.

D. Additional Experimental Results

Here, we provide the additional results for the experiments discussed in the main paper. Full results are shown in Figure 6 for the bandit results and Figure 7 for the driving scenario. Table D.2 contains an overview of all algorithms and baselines

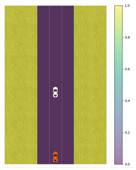
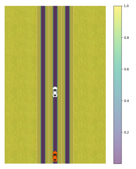
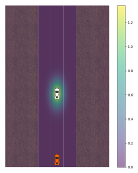
Feature	Description	Type	Definition		θ_1	θ_2	θ_3	ϕ
$f_1(s)$	Target velocity	Numerical	$-(v-0.4)^2$		1	0	1	0
$f_2(s)$	Target location	Numerical	$-(x-x_r)^2$ x_r center of right lane		0	1	0	0
$f_3(s)$	Stay on street	Binary	1 iff off street		0	0	0	0.3
$f_4(s)$	Stay in lane	Numeric	$\frac{1}{1+\exp(-bd+a)}$, d distance to closest lane center, $b = 10000$, $a = 10$		0	0	0	0.05
$f_5(s)$	Stay aligned with street	Numeric	$\lvert\cos(\theta)\rvert$		0	0	0	0.02
$f_6(s)$	Don't drive backwards	Binary	1 iff $v < 0$		0	0	0	0.5
$f_7(s)$	Stay within speed limit	Binary	1 iff $v > 0.6$		0	0	0	0.3
$f_8(s)$	Don't get too close to other cars	Numeric	$\exp(-b(c_1 d_x^2 + c_2 d_y^2) + ba)$, $a = 0.01$, $b = 30$, $c_1 = 4$, $c_2 = 1$		0	0	0	0.8

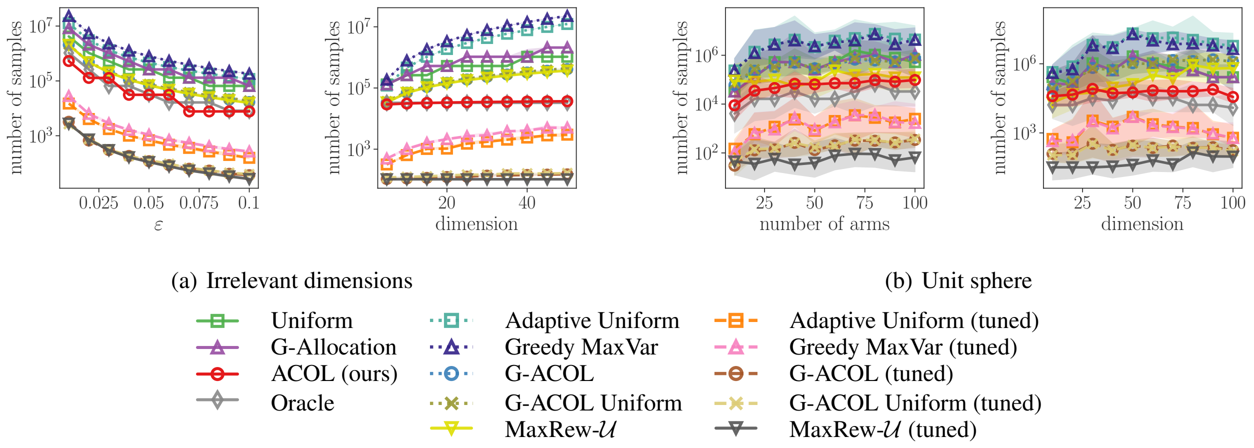
Table C.1. Features for representing the reward and constraint function in the Driver environment. The last four columns contain the reward weights for the three scenarios and the constraint weight that is shared.

Name	Confidence Intervals	Selection Criterion	Select From Arms	Plot Color
Oracle	static	Oracle	All	—◇—
G-Allocation	static	MaxVar	All	—△—
Uniform	static	Uniform	All	—□—
ACOL	static	MaxVar	Uncertain	—○—
Greedy MaxVar	adaptive	MaxVar	All	··△··
Adaptive Uniform	adaptive	Uniform	All	··□··
MaxRew-$\mathcal{U}$	adaptive	Max Rew	Uncertain	—▽—
MaxRew-$\mathcal{F}$	adaptive	Max Rew	Feasible	-
G-ACOL	adaptive	MaxVar	Uncertain	··○··
G-ACOL Uniform	adaptive	Uniform	Uncertain	··×··
Greedy MaxVar (tuned)	adaptive tuned	MaxVar	All	–△–
Adaptive Uniform (tuned)	adaptive tuned	Uniform	All	–□–
G-ACOL (tuned)	adaptive tuned	MaxVar	Uncertain	–○–
G-ACOL Uniform (tuned)	adaptive tuned	Uniform	Uncertain	–×–
MaxRew-$\mathcal{U}$ (tuned)	adaptive tuned	Max Rew	Uncertain	—▽—
MaxRew-$\mathcal{F}$ (tuned)	adaptive tuned	Max Rew	Feasible	-

Table D.2. Overview of all algorithms we evaluate.

that we evaluated.

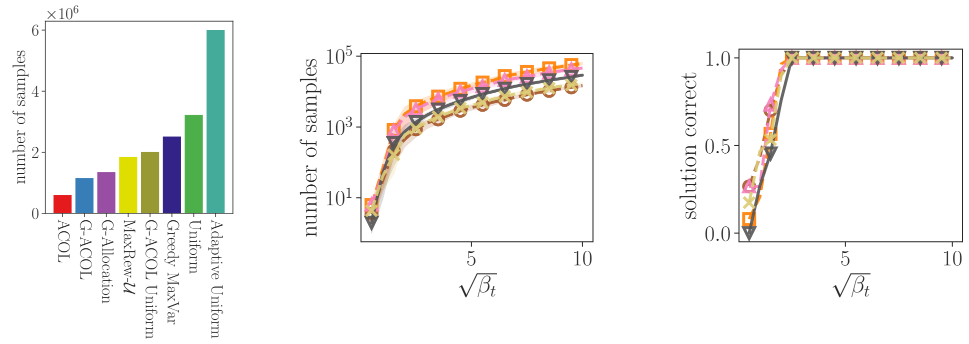
We find that methods that select arms from $\mathcal{U}$ randomly (G-ACOL Uniform) or by maximizing the reward (MaxRew-$\mathcal{U}$) can perform quite well in some cases with tuned confidence intervals. Indeed, MaxRew-$\mathcal{U}$ outperforms G-ACOL in the unit sphere experiment. This is not consistent across environments, and G-ACOL performs comparable or better in all other environments. Still, in some cases, when theoretical guarantees are not required, these heuristic approaches might be valuable alternatives.



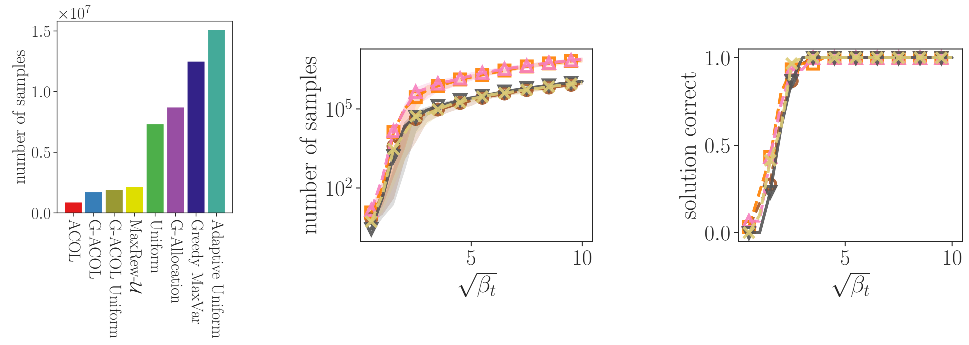
(a) Irrelevant dimensions

(b) Unit sphere

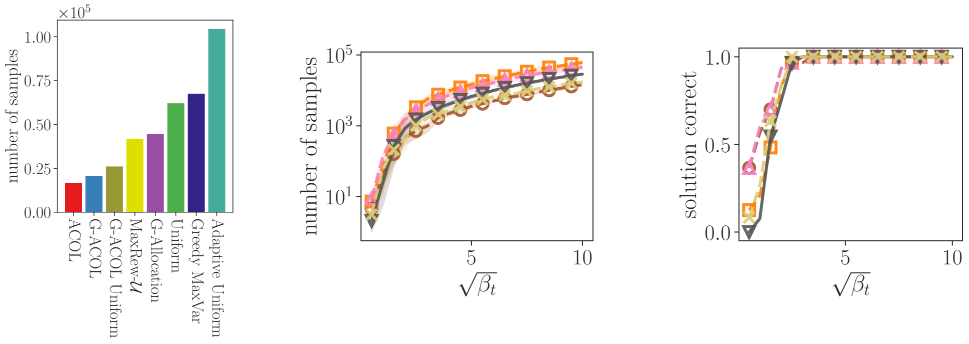
Figure 6. Similar plots to Figure 2, including some additional algorithms: the "non-tuned" versions of algorithms use the confidence interval from Proposition 3, and *G-ACOL Uniform* is G-ACOL with uniform sampling instead of the maximum variance objective. Table D.2 provides an overview of all baselines. Moreover, the plots here show the 25th and 75th percentiles over 30 random seeds. For "irrelevant dimensions", these are close to the median, but for "unit sphere", there is much more randomness because the instances are randomly generated.



(a) "Base scenario"



(b) "Different reward"



(c) "Different environment"

Adaptive Uniform
Greedy MaxVar
MaxRew-𝒰
G-ACOL
G-ACOL Uniform

Figure 7. Similar plots as in Figure 5 for all three driving scenarios from Figure 1, showing G-ACOL Uniform as aan additional baseline.