Master Thesis

Exploiting Side Information in Partial Monitoring Games
An Empirical Study of the CBP-SIDE Algorithm with Applications to Procurement

Author(s):
Lienert, Ian

Publication Date:
2013

Permanent Link:
https://doi.org/10.3929/ethz-a-009980100

Rights / License:
In Copyright - Non-Commercial Use Permitted
Exploiting Side Information in Partial Monitoring Games

An Empirical Study of the CBP-SIDE Algorithm
with Applications to Procurement

Master Thesis
I. Lienert
September 25, 2013

Advisors: Prof. Dr. A. Krause, Dr. G. Bartók, A. Singla
Department of Computer Science, ETH Zürich
Abstract

Partial monitoring games are played sequentially between a learner and an environment. In each round, the learner performs an action and the environment selects an outcome. The result is manifested as a loss incurred upon the learner. In general, the learner may not be aware of his exact losses, but receives some form of limited feedback. In many cases, the learner may receive side information at the beginning of each round which he may use to further inform his actions. It turns out that certain settings lend themselves naturally to this model. Procurement, dynamic pricing and product testing are a few examples.

We implement an instantiation of the CBP-SIDE algorithm that uses linear least squares estimators to determine the best possible action given the side information. We compare the performance CBP-SIDE on a variety of problem instances, using a leading contextual bandit algorithm as our baseline. Our settings include: contextual bandits with linear payoffs, several synthetic problem instances of varying levels of difficulty and an online procurement setting. In the latter, we introduce a novel consideration of worker utilities and evaluate the performance of CBP-SIDE across several loss models.
Contents

1 Introduction 1
   1.1 Our Contributions 4

2 Related Work 5
   2.1 Online Learning with Side Information 5
   2.2 Online Learning in Procurement Markets 6

3 Preliminaries 9
   3.1 Partial Monitoring with Linear Side Information 9
   3.2 Regret 10
   3.3 Cells and Observability 11

4 The CBP-SIDE Algorithm 13
   4.1 Fitting a Linear Model 13
   4.2 Reducing the Action Set 15
   4.3 Theoretical Guarantees 16

5 Extensions 17
   5.1 Runtime Optimizations 17
   5.2 State Caching 18
   5.3 Regression 19
   5.4 Performance Improvements 19

6 Contextual Bandits as Partial Monitoring Games 21
   6.1 Contextual Bandits with Linear Payoffs 21
   6.2 Reconciling the Models 22
   6.3 Discussion 23

7 Experimental Results on Synthetic Games 25
Chapter 1

Introduction

In online learning, a learner (he) must make decisions in a dynamic but structured environment (she/it) over the course of a sequence of rounds. The learner performs an action, suffers some loss and then alters his behaviour in future rounds based on this loss. What makes this field of study different from traditional machine learning is that the learner is presented with the dual problem of both minimizing his cumulative loss while at the same time learning the true structure of the environment.

At the end of each round, the learner may be presented with full information about the environment. That is, upon choosing an action, the learner is aware of his loss as well as the loss associated with all of the other actions he could have taken. Sports betting is an excellent example of online learning with full information. A gambler may select the action of betting $X$ dollars on team $Y$ winning the match. Upon observing the match and seeing that team $Y$ lost, the gambler is not only aware of his loss of $X$ dollars, but he knows exactly what his losses or gains would have been had he chosen his bet differently.

If the information revealed to the learner at the end of each round is exactly his loss for the chosen action, we call it bandit information. The canonical example of a problem with bandit information is that of $N$-armed bandits. A gambler in a casino has access to a number of slot machines. Machine $i$ pays out $1,000 with probability $p_i$ and pays out nothing with probability $1 - p_i$. By pulling the arm of the $i$-th machine, the agent observes the reward associated with only that machine. If the learner plays a fixed arm over the course of the whole process, he should start to become reasonably confident in his expected reward associated with that arm, but may have suffered large opportunity costs for not having chosen a more rewarding one along the way. In effect, the learner must trade off “exploration” of the different arms available to him and “exploitation” of the single fixed arm that he is most confident as having the highest expected payoff.
Partial monitoring games are a class of games that provide a model for an online learner acting in an environment where the losses incurred are concealed by limited feedback. Concretely, in each round, the environment selects an outcome or some state of the world, while the learner selects an action. The action-outcome pair are deterministically mapped to both a loss and a feedback. The learner suffers the loss, but does not observe it. He instead observes the feedback. It turns out that partial monitoring is capable of categorizing a wide variety of online learning problems, ranging from label efficient prediction [16] to $N$-armed bandits. We consider an augmented version of partial monitoring, one in which, at the beginning of each round, the learner is presented with some side information that may be pertinent to the environment’s outcome. In this work, we assume that the outcome is a stochastic function of the side information.

Product testing is a rudimentary example of partial monitoring. A manufacturer has a series of goods that he may address one-by-one. At each round, the manufacturer can test a good thereby rendering it useless, or ship the good. Testing the good provides the manufacturer with some useful information, but will destroy potentially usable goods. Shipping the good results in profit for the manufacturer, but his reputation may be negatively affected if a faulty good was shipped. The side information in this instance may encode visible information about the good (for example, scuff marks and burns) or expert advice. Here, the manufacturer would wish to learn the relationship between side information and the good’s quality so that he may ship only the goods that he can be confident are usable, based on the information provided. We revisit this problem later under the guise of apple tasting [17].

Consider the more complicated scenario of dynamic pricing. The learner is a monopolist who wishes to sell a single product to a stream of customers. As customers enter the store, the vendor can translate side information about each customer into an asking price for the product. If the asking price is at most the customer’s valuation of the product, the transaction happens and the payment is issued, but the vendor receives no further information about the true valuation. If the price was set too high, the customer rejects and the round ends. In the case of a failed transaction, the vendor may suffer a fixed storage cost, but may have undercharged in the case of a successful one. The goal here is to learn the optimal price that most aligns with the customers’ valuations given the side information.

A dual problem, and one that we focus the most attention on in this work, is that of online procurement. The learner is a requester, say on a digital crowdsourcing platform such as Amazon Mechanical Turk (AMT), who wishes to hire a large number of workers, each to perform a single task. Again,
the requester has access to a set of side information associated with each worker (age, location, etc.). The requester offers a price and the worker may accept or reject without revealing her true cost. We may also be interested in procuring services from high quality workers who have some experience in the domain. After a successful transaction, the requester may also receive explicit feedback about the worker’s value.

In all of these settings, an intelligent learner would utilize the past history of the game in such a way as to induce how the environment would behave in future rounds. He should exploit this model to his own benefit and minimize his cumulative loss over time. A commonly accepted measure of performance of a learner is regret. Regret measures the difference in cumulative loss between a learner and an oracle who knows the mechanics of the environment. Good online learners manage to attain no-regret status on problem instances where it’s possible to do so. That is, if $T$ is the total number of rounds the game is played for, then $R_T/T \to 0$ as $T \to \infty$, where $R_T$ is the regret of the learner after $T$ rounds. In other words, the learner’s regret should be sublinear.

We implement an instantiation of CBP-SIDE [9], a meta-algorithm designed for stochastic partial monitoring games with side information. This instantiation uses linear least squares estimators to map side information to an estimate of the outcome distribution as served up by the environment. The algorithm leverages these estimates to determine expected loss differences between pairs of actions and successively eliminates “probably suboptimal” actions. We outline the structure and internals of the algorithm and exhibit optimizations that were made to boost performance.

As a baseline competitor, we use the LinUCB algorithm [20], which is designed for contextual bandit games, a subclass of partial monitoring games with side information. Here, each action is associated with a per-round side information vector, which we assume to be linearly related to the expected reward of that action. As in traditional bandit problems, we assume that the losses and feedback coincide.

We generate synthetic outcomes across all of the settings mentioned above, plus an abstract contextual bandit setting using binary losses. We deploy and evaluate the performance of CBP-SIDE on these synthetic instances before moving on to our case study, wherein real-world data collected from AMT was utilized to simulate an online procurement market. Figures 1.1(a) and 1.1(b) depict how CBP-SIDE accrues losses over time in a real-world procurement setting. In 1.1(a), more loss information is encoded in the feedback, while in 1.1(b), this information is obscured. One can see that CBP-SIDE excels in such settings where determining the true loss is more difficult.
1. Introduction

(a) Performance on an “easy” game.

(b) Performance on a “hard” game.

Figure 1.1: CBP-SIDE versus various baselines on real-world procurement data.

1.1 Our Contributions

- We implement CBP-SIDE with linear least squares estimators and evaluate its performance on a variety of problem instances.
- We present optimized learning parameters that yield drastically reduced regret growth across several nontrivial settings.
- We provide a nontrivial representation of contextual bandits within partial monitoring. Contrarily, we provide nontrivial reductions from partial monitoring to contextual bandits (possibly involving some information loss) to yield a competitive baseline.
- We introduce a novel procurement setting, namely one which differentiates worker values, and model it as a partial monitoring game.
- We consider a number of interpretations of loss under procurement and prove that two “textbook” notions of loss are equivalent under partial monitoring.
We survey the appropriate literature in the field of online learning with side information with an emphasis on partial monitoring and contextual bandits. We also briefly review some work done in the area of online procurement as a learning problem since our case study (Chapter 8) deals exclusively with this topic.

2.1 Online Learning with Side Information

The theoretical component of this work deals almost exclusively with the problem of partial monitoring. The area of partial monitoring as a sequential prediction problem was first studied by Piccolboni and Schindelhauer [21], where it was determined that any partial monitoring game having discrete loss and feedback functions will have minimax regret of either $1 \tilde{O}(T^{3/4})$ or $\Omega(T)$. To prove the former, the authors presented the FEEDEXP3 algorithm and showed that when sufficient information about the losses is provided in the feedback, the given regret bound holds. To prove the latter, they contrarily showed that if there is insufficient loss information in the feedback, the regret must be linear with probability at least $1/2$. The upper bound in the sublinear regret case was later tightened to $O(T^{2/3})$ by Cesa-Bianchi et al. [13] who gave a randomized strategy that attains this regret.

The full classification of partial monitoring games with respect to their minimax regret growth was provided in Bartók et al. [10], who extended the above results to show that, quite surprisingly, each finite partial monitoring game falls into one of only four categories: trivial games having 0 minimax regret, easy games having $\Theta(\sqrt{T})$ minimax regret, hard games having $\tilde{O}(T^{2/3})$ minimax regret and hopeless games having $\Theta(T)$ minimax regret. They showed that locally observable games (to be defined in Chapter 3) are

\footnote{The tilde in $\tilde{O}()$ and $\tilde{\Theta}()$ abstracts away poly-logarithmic terms.}
2. Related Work

easy, while games that aren’t locally observable but are *globally observable* are hard. In proving the upper bound for easy games, Bartók et al. [10] designed the Balaton algorithm which enjoys the specified regret on locally observable games. In Bartók et al. [11], a powerful anytime algorithm was provided, capable of achieving regret in the order of all bounds above for the corresponding problem instances.

The papers listed so far assume a stochastic environment where each outcome is i.i.d. However, Foster and Rakhlin [15] showed that the bounds above were shown to hold even in the adversarial case. Recently, Bartók [7] tightened the regret bound for locally observable games by removing an explicit dependence on the number of actions.

The problem of partial monitoring with *side information* was only recently proposed by Bartók and Szepesvári [9], who showed that locally observable games with the augmentation of side information still enjoy the same minimax regret as that of easy games without side information by presenting an algorithm scheme using plug-in estimators called CBP-SIDE. They showed that as long as the probability that the estimator falls outside a certain confidence width of the true parameter decays sufficiently quickly, the scheme suffers regret $\tilde{O}(\sqrt{T})$.

A huge body of work deals exclusively with a special case of partial monitoring, called *multi-armed bandits* [22], where the losses and feedback coincide. The subfield of contextual bandits with linear payoffs [1] extends the multi-armed bandit setting by introducing per-action feature vectors that provide further information about the expected loss of that action. Auer [4] introduced the LinRel algorithm, achieving optimal (modulo polylog terms) regret of $\tilde{O}(\sqrt{TD})$, where $D$ is the dimension of the side information. Chu et al. [14] then gave an algorithm, LinUCB, that is based closely on LinRel which attains similar asymptotic regret bounds but is simpler and more computationally efficient.\(^2\) Furthermore, LinUCB has been experimentally tested and has been shown to perform very well on real data [20]. It is with this in mind, that we decided on LinUCB as a baseline for the majority of our experiments.

2.2 Online Learning in Procurement Markets

A large portion of this thesis considers the problem of online procurement. In some settings, a bidding model is considered, where arriving agents have their own private cost for doing a particular task, but instead specify a bid to the requester which may be a distortion of their true cost ([24], [5], [25]). In these settings, there is an emphasis on designing not only optimal but\(^2\)LinRel requires computing the singular value decomposition of a matrix, while LinUCB need only invert it.
2.2. Online Learning in Procurement Markets

Truthful mechanisms, those mechanisms where the underlying agents have no incentive to lie about their valuations. Another setting considers posted price mechanisms where the requester specifies a fixed “take-it-or-leave-it” price to workers. Workers can either accept or reject the offer.

Several efficient algorithms exist for the budget feasible case where an overall payment threshold is imposed as a constraint ([6], [25], [5]). A “dual” framing of the problem is considered in frugal mechanisms where the budget is to be minimized subject to constraints ([18], [3]).

In this work, however, we restrict ourselves to a posted price model which is trivially truthful. Furthermore, we impose no budget constraint, i.e. we assume an unlimited budget, and instead include the budget explicitly in our loss function to be minimized. Online procurement problems of this flavour have been studied in Blum et al. [12] and Kleinberg and Leighton [19], but none of which address the addition of side information. We also consider the case where the requester has a utility function over each worker — a modification that, as far we know, has not yet been studied.
Adopting notation largely from Bartók and Szepesvári [9] as well as Bartók et al. [11], we introduce the necessary definitions and notation.

### 3.1 Partial Monitoring with Linear Side Information

A finite stochastic partial monitoring game with linear side information (henceforth, partial monitoring game) is a game $G = (L, H, K)$ played between a learner and an environment over the course of a finite number of rounds (time steps). Let $T$ be the number of rounds and denote the (finite) set of actions available to the learner by $\mathcal{N} = \{1, \cdots, N\}$ and the (finite) set of outcomes available to the environment by $\mathcal{M} = \{1, \cdots, M\}$. Let $L \in \mathbb{R}^{N \times M}$ be the loss matrix associating each action-outcome pair with a loss in $\mathbb{R}$ and the feedback matrix $H \in \Sigma^{N \times M}$ associating the respective pair with a feedback symbol from some alphabet $\Sigma$.

At each time step $t$, the learner observes side information $x_t \in \mathbb{R}^D$ which completely determines the distribution over outcomes the environment plays according to. We assume $\|x_t\|_1 = 1$. The learner then selects action $i_t$ whilst, simultaneously, the environment selects action $j_t$ and the loss $L_{i_t,j_t}$ is suffered by the learner without his knowing. The symbol $H_{i_t,j_t}$ is instead revealed to the learner, after which the round ends. Note that, despite being ignorant of $L_{i_t,j_t}$ for each $t$, the learner is fully aware of $L$ and $H$. Unless specified otherwise, we assume that all entries of $H$ are natural numbers and if row $i$ of $H$ consists of $\sigma_i$ unique symbols, then the set of all symbols appearing in row $i$ is $\{1, \cdots, \sigma_i\}$. Since we are only interested in identifying different outcomes for a given action $i$, we can alter the symbols of row $i$ in any way we choose as long as we respect that different symbols remain different and similar symbols remain similar under the new transformation.

The sequence $\{x_t\}$ may be chosen arbitrarily and may also depend on the
3. Preliminaries

game’s history \( \mathcal{H}_{t-1} = (x_1, I_1, J_1, \ldots, x_{t-1}, I_{t-1}, J_{t-1}) \). The mapping from side information space to the \( M \)-dimensional probability simplex \( \Delta_M \subset \mathbb{R}^M \) is realized by the stochastic matrix \( K \in \mathbb{R}^{M \times D} \). That is, \( K \) maps side information vectors to probability distributions over outcomes. Denote the column vector associated with the \( i \)th row of \( L \) by \( \ell_i \). The expected loss of action \( i \) given side information \( x_t \) is thus given by \( \mathbb{E}[\mathcal{L}_{i, J_t} | x_t] = \ell_i^T K x_t \).

A bandit game is a special case of partial monitoring. In a bandit game, \( L = H \). That is, the learner is revealed his true loss at the end of each round.

3.2 Regret

Assume the learner uses policy \( \mathcal{A}(x_t, \mathcal{H}_{t-1}) \) to select each action. Then, the regret associated with policy \( \mathcal{A} \) measures the difference between the loss accrued by the learner using \( \mathcal{A} \) and that of the oracle who plays with full knowledge of \( K \). Concretely, the regret is given by

\[
R^A_T = \sum_{t=1}^{T} \mathcal{L}_{\mathcal{A}(x_t, \mathcal{H}_{t-1}), I_t} - \min_{i \in \mathcal{N}} \ell_i^T K x_t,
\]

Another useful metric is the expected regret of a policy. This is simply the expectation of the regret with respect to \( J_t \). Formally, the expected regret of policy \( \mathcal{A} \) is

\[
\mathbb{E}[R^A_T] = \sum_{t=1}^{T} \mathbb{E}[\mathcal{L}_{\mathcal{A}(x_t, \mathcal{H}_{t-1}), I_t} | x_t] - \min_{i \in \mathcal{N}} \ell_i^T K x_t.
\]

The learner’s goal is to minimize his cumulative loss \( \sum_{t=1}^{T} \mathcal{L}_{i_t, I_t} \) and hence his (expected) regret.

The worst-case regret of an algorithm \( \mathcal{A} \) on game \( G \) is defined as

\[
R_T^\mathcal{A}(G) = \sup_{J_1, \ldots, J_T} R_T^\mathcal{A}.
\]

The minimax expected regret (or minimax regret) of a game \( G \) is the worst-case regret of the best algorithm as given by

\[
R_T(G) = \inf_{\mathcal{A}} \mathbb{E}[R_T^\mathcal{A}(G)].
\]

Minimax regret is a property inherent to \( G \) and is used to measure the “hardness” of \( G \).
3.3 Cells and Observability

The cell of action $i$ is the set of distributions for which $i$ is optimal:

$$\mathcal{C}_i = \{p \in \Delta_M : \ell_i^T p \leq \ell_j^T p, \forall j \in \mathcal{N}\}.$$ 

The cell decomposition of a partial monitoring game is the set of all cells $\mathcal{C} = \{\mathcal{C}_i : i \in \mathcal{N}\}$. Note that for each $\mathcal{C}_i \in \mathcal{C}$, $\mathcal{C}_i$ is either empty or a closed convex polytope and that $\cup_{i \in \mathcal{N}} \mathcal{C}_i = \Delta_M$. If $\mathcal{C}_i = \emptyset$, action $i$ is called dominated. If action $i$ is not dominated, but $\mathcal{C}_i \subset \mathcal{C}_j$ for some action $j$, then $i$ is called degenerate.

Recall row $i$ of the feedback matrix $H$ consists of exactly $\sigma_i$ unique symbols from the set $\{1, \cdots, \sigma_i\}$. Let $k$ be such a symbol and let $Y_i(k) = e_k$, where $e_k$ is the $k$th standard basis vector in $\mathbb{R}^{\sigma_i}$. The signal matrix of action $i$ is given by $S_i = \left( Y_i(H_{i,1}) \mid \cdots \mid Y_i(H_{i,M}) \right) \in \mathbb{R}^{\sigma_i \times M}$. Intuitively, the signal matrix for action $i$ maps outcome distributions $K_{j\ell}$ into observation distributions, given that action $i$ was played.

Action $i$ is called Pareto-optimal whenever $\mathcal{C}_i$ is an $(M-1)$-dimensional polytope. The set of Pareto-optimal actions is denoted by $\mathcal{P}$. Two Pareto-optimal actions $i$ and $j$ are called neighbours whenever $\mathcal{C}_i \cap \mathcal{C}_j$ is an $(M-2)$-dimensional polytope. Denote the set of all (unordered) neighbouring action pairs $\{i, j\}$ by $\mathcal{K}$. The neighbourhood action set of actions $i, j \in \mathcal{P}$ is the set of actions $K_{ij}^+ = \{k : \mathcal{C}_i \cap \mathcal{C}_j \subset \mathcal{C}_k\}$. Intuitively, this set includes all actions that could yield further information about the observation distributions without "harming" us more than strictly playing $i$ and $j$, irrespective of the outcome distribution.

$G$ is said to be globally observable if, for all $i, j \in \mathcal{N}$, there exists, for each $k \in \mathcal{N}$, an observer vector $v_{i,j,k} \in \mathbb{R}^{\sigma_k}$ such that $\ell_i - \ell_j = \sum_{k \in \mathcal{N}} S_k^T v_{i,j,k}$. A pair of neighbouring actions $i$ and $j$ is said to be locally observable whenever there exists, for each $k \in K_{ij}^+$, a vector $v_{i,j,k} \in \mathbb{R}^{\sigma_k}$ such that $\ell_i - \ell_j = \sum_{k \in K_{ij}^+} S_k^T v_{i,j,k}$. Observer vectors act so as to tie the observed feedback to the expected losses. Denote the set of unordered locally observable action pairs by $\mathcal{L}$. $G$ is called locally observable whenever $\mathcal{L} = \mathcal{K}$.

The set of observer actions underlying actions $\{i, j\}$ is a set $\mathcal{V}_{ij} \subset \mathcal{N}$ satisfying all of the following constraints:

1. $\ell_i - \ell_j = \sum_{k \in \mathcal{V}_{ij}} S_k^T v_{i,j,k}$.
2. $\mathcal{V}_{ij} \supset K_{ij}^+$.
3. If $i$ and $j$ are a locally observable pair, then $\mathcal{V}_{ij} = K_{ij}^+$.

Note there may still be several ways of selecting the set $\mathcal{V}_{i,j}$, but in practice it is preferable to choose the smallest possible set. Denote the observer order of an action $k$ by $W_k = \max_{i,j,k \in \mathcal{V}_{ij}} \|v_{i,j,k}\|_2$. 

11
Chapter 4

The CBP-SIDE Algorithm

In this chapter, we flesh out the details of the CBP-SIDE algorithm by Bartók and Szepesvári [9] by providing the pseudocode (see Algorithm 1) as well as giving a high level description of the regression and action-selection stages. Afterwards, we give a short summary of the regret bounds and conjecture that CBP-SIDE attains sublinear regret on all partial monitoring games having sublinear minimax regret.

Note that in Bartók and Szepesvári [9], it is assumed that all actions are in $\mathcal{P}$ for ease of presentation. We do away with this assumption and incorporate the extensive functionality adopted from CBP [11].

4.1 Fitting a Linear Model

Recall the model assumption: the outcome distribution at time $t$ is realized by the distribution $Kx_t$ for a given stochastic matrix $K$. So, the distribution over observations, given that action $i$ was played at time $t$, is given by $\mathbb{E}[Y_t(J_t) | x_t] = S_iKx_t$. Then, there must exist some matrix $\pi^*_i = S_iK \in \mathbb{R}^{\sigma_i \times D}$ mapping side information to observation distributions. It is these parameters $\pi^*_i$ that we wish to learn.

Let $Y_t = Y_t(J_t)$. The function fitLINEARMODEL uses regularized least squares to determine the estimator for $\pi^*_i$, given by

$$
\hat{\pi}_i(t) = \arg \min_{\pi \in \mathbb{R}^{\sigma_i \times D}} \sum_{s=1}^{n_i(t-1)} \| Y_t(s) - \pi x_t(s) \|_2^2 + \lambda_i \| \pi \|_2^2,
$$

where $t_i(s)$ is the time step where action $i$ was played the $s^{th}$ time, $n_i(t)$ is the total number of times $i$ was played up to and including $t$ and the $\lambda_i$'s are tuning parameters that control the complexity of the model. The closed
Algorithm 1: CBP-SIDE

**Input:** loss matrix \( L \), feedback matrix \( H \), \( f_k \)

**Output:** Sequence of actions \((I_1, \ldots, I_T)\)

Compute \( \mathcal{P}, \mathcal{K}, \mathcal{V}_{i,j}, v_{i,j,k}, W_k, r = 0 \)

**for** \( t = 1 \) \( \text{to} \) \( N \) \( \text{do} \)
- Receive side information \( x_t \)
- Select action \( I_t \leftarrow t \)
- Select action \( I_t \leftarrow 1 \)
- Observe feedback \( J_t \)
- \( Y_t \leftarrow Y_t(I_t) \)
- \( \tilde{\pi}_t \leftarrow \text{fitLINEARMODEL}(x_t, Y_t) \)

**for** \( t = N + 1 \) \( \text{to} \) \( T \) \( \text{do} \)
- Receive side information \( x_t \)
- for \( i \in \mathcal{N} \) \( \text{do} \)
  - \( \tilde{q}_i \leftarrow \tilde{\pi}_i x_t \)
  - \( w_i \leftarrow \text{GETCONFIDENCEWIDTH}(i, \mathcal{H}_{t-1}, x_t) \)
- for \( \{i, j\} \in \mathcal{K} \) \( \text{do} \)
  - \( \tilde{d}_{ij} \leftarrow \sum_{k \in \mathcal{K}_{ij}} v_{i,j,k}^T \tilde{q}_k \)
  - \( c_{ij} \leftarrow \sum_{k \in \mathcal{K}_{ij}} \|v_{i,j,k}\|_2 w_k \)
  - if \( |\tilde{d}_{ij}| \geq c_{ij} \) then
    - \( \mathcal{P}(t), \mathcal{K}(t) \leftarrow \text{HALFSPACE}(i, j, \text{sgn}(\tilde{d}_{ij}), \mathcal{P}(t)) \)
  - \( \mathcal{K}^+(t) \leftarrow \cup_{(i,j) \in \mathcal{K}(t)} \mathcal{K}^+_{ij} \)
  - \( \mathcal{V}(t) \leftarrow \cup_{(i,j) \in \mathcal{K}(t)} \mathcal{V}_{ij} \)
  - \( \mathcal{R}(t) \leftarrow \{k \in \mathcal{V}(t) : r \leq f_k(t)\} \)
  - \( \mathcal{S}(t) \leftarrow \mathcal{P}(t) \cup \mathcal{K}^+(t) \cup \mathcal{R}(t) \)
- Select action \( I_t \leftarrow \arg \max_{j \in \mathcal{S}(t)} W_i w_j \)
- if \( I_t \in \mathcal{V}(t) \setminus (\mathcal{P}(t) \cup \mathcal{K}^+(t)) \) then
  - \( r \leftarrow r + 1 \)
- Observe feedback \( J_t \)
- \( Y_t \leftarrow Y_t(I_t) \)
- \( X_{I_t,t} \leftarrow \begin{pmatrix} x_{I_t(1)} & \cdots & x_{I_t(n_t(t))} \end{pmatrix} \)
- \( Y_{I_t,t} \leftarrow \begin{pmatrix} Y_{I_t(1)} & \cdots & Y_{I_t(n_t(t))} \end{pmatrix} \)
- \( \tilde{\pi}_t \leftarrow \text{fitLINEARMODEL}(X_{I_t,t}, Y_{I_t,t}) \)
4.2. Reducing the Action Set

Denote the expected loss difference between $i$ and $j$ by $d_{ij}(t) = (\ell_i - \ell_j)^\top Kx_t$. Rather than selecting actions based on individual expected losses, CBP-SIDE computes expected loss difference estimates $\tilde{d}_{ij}(t)$ between neighbouring actions. Suboptimal actions having large expected loss difference estimates are eliminated. If the game is globally observable then, for each neighbouring action pair $\{i, j\}$, $V_{ij} = K_{ij}^+$ and so the expected loss difference is

$$d_{ij}(t) = (\ell_i - \ell_j)^\top Kx_t = \sum_{k \in K_{ij}^+} v_{ij,k}^\top S_k Kx_t = \sum_{k \in K_{ij}^+} v_{ij,k}^\top \pi^*_k x_t.$$

Here, the mapping $\pi^*_k$ is unknown, but we can plug in our estimator $\tilde{\pi}_k$ to obtain our loss difference estimate $\tilde{d}_{ij} = \sum_{k \in K_{ij}^+} v_{ij,k}^\top \tilde{\pi}_k x_t$. Bartók and Szepesvári [9] showed that by setting the provided confidence width for $\tilde{d}_{ij}$ to $c_{ij} = \sum_{k \in K_{ij}^+} \|v_{ij,k}\|_2 \tilde{w}_k$, the sublinear regret bound discussed in the following section must hold.

It is obvious that if $d_{ij}(t) > 0$, then all actions $k$ having $C_k \subset \{p \in \Delta_M : (\ell_i - \ell_j)^\top p < 0\}$ are suboptimal and can be eliminated. The algorithm removes from consideration all such “suboptimal open half-spaces” with sufficiently large loss difference estimates. All Pareto-optimal actions whose

\[ \tilde{\pi}_i(t) = \mathcal{Y}_{ij}^\top (\lambda I_D + X_{ij}^\top X_{ij})^{-1}, \]

where

$$X_{ij} = (x_{ij(1)} \mid \cdots \mid x_{ij(n_i(t-1))}), \quad \mathcal{Y}_{ij} = (Y(J_{ij(1)}) \mid \cdots \mid Y(J_{ij(n_i(t-1))})).$$

With this, we can estimate our outcome distribution as $\tilde{\pi}_i x_t$ and compute the respective confidence widths $w_i$ as done in getConfidenceWidth. Larger confidence widths $w_i$ indicate more uncertainty in the estimate $\tilde{\pi}_i$. There are number of different ways to specify the widths. Indeed, we offer comparisons of two such formulations, but defer the details to the following chapter. For the remainder of this chapter, however, we assume $w_i$ is as appears in Bartók and Szepesvári [9] such that the theoretical regret guarantees are fulfilled. Specifically, we assume $w_i = D(\sqrt{(D + 1) \log t + \sigma_i}) x_t^\top (\lambda I_D + X_{ij} X_{ij}^\top) x_t$.  

\[^1\text{Tragically, the notation for the identity matrix } I_D \in \mathbb{R}^D \text{ overlaps with that of chosen actions } l_i, \text{ but the distinction should be clear from the context.}\]

\[^2\text{Actually, the paper gives a larger width setting, but it was confirmed with the authors that the one shown here is correct.}\]
cells intersect with the remaining closed polytope \( \hat{C}(t) \) are added to the set of candidate Pareto-optimal actions \( \mathcal{P}(t) \) from which the algorithm will select \( I_t \). Furthermore, all neighbouring pairs of actions \( \{i, j\} \in \mathcal{K} \) whose cell intersection \( C_i \cap C_j \) intersects with \( \hat{C}(t) \) are added to \( \mathcal{K}(t) \).

We discourage “harmful exploration” by maintaining a counter variable \( r \) which is incremented whenever actions whose cells are strictly outside of \( \hat{C} \) are played. Whenever \( r \) exceeds \( f_k(t) \) for a given action \( k \), only actions from \( k \)'s observer set whose cells lie within \( \mathcal{P}(t) \) or \( K^+(t) \) are considered.

Of the remaining candidate actions, we select action \( I_t \) with the largest uncertainty as measured by \( w_i \) and \( c_{i,j} \).

### 4.3 Theoretical Guarantees

Using the confidence widths specified above, CBP-SIDE attains a regret bound of order \( O(N + N^{3/2}D^2 \sqrt{T \log T}) \) [9]. Note this bound only applies to locally-observable partial monitoring games. This result matches the tight minimax regret bound of \( \tilde{\Theta}(\sqrt{T}) \) ([2], [10]).

No sublinear regret bound has yet been provided for CBP-SIDE on globally observable, non-locally observable games, though such games were proven to have minimax regret \( \Theta(T^{2/3}) \) ([10], [13]). We suspect, however, that the regret is indeed in \( O(T^{2/3}) \) as the algorithm quite closely resembles CBP for which such bounds have been proven [11]. Furthermore, empirical tests run on a variety of different non-locally observable games indicate sublinear regret growth, as we shall see in Chapters 6 and 7.
The implementation of CBP-SIDE was written in MATLAB and used the existing implementation of the CBP algorithm [11] as a basis. In this chapter, we outline the major extensions added for the purposes of the thesis. Our contributions range from runtime optimizations developed for speeding up the regression as well as determining the dimensionality of certain cell polytopes, caching of environment states to reduce the number of linear programs, and optimizing learning parameters to further empirically minimize regret.

5.1 Runtime Optimizations

The most computationally heavy portions of CBP occur in sections where cell dimensions are determined. When computing each cell’s dimension, a total of $2M + 1$ linear programs are run. One program is used to determine whether the cell is empty while two are required for each dimension of the cell to determine whether the “width” with respect to that coordinate of the cell is nonzero. This computation happens in the function `getCellDimension`.

Determining $\mathcal{P}$, $\mathcal{K}$ and $K_{ij}^+$ all require techniques to determine the dimension of individual cells or cell intersections. In the case of $\mathcal{P}$, we simply need to iterate over each action in $\mathcal{N}$ and find those cells having dimension $M - 1$. Determining $\mathcal{K}$ requires calling `getCellDimension` on $C_i \cap C_j$ for each $i, j \in \mathcal{P}$. Computing $K_{ij}^+$ requires calling `getCellDimension` on the intersection $C_i \cap C_j \cap C_k$ for up to $N - 2$ possible resulting polytopes.

We reduce the runtime of this initial preprocessing phase as follows. First, we introduce a matrix $D \in \mathbb{R}^{N \times N}$ to store precomputed dimensions of cell
intersections. That is, we store the dimension of \( C_i \cap C_j \) in \( D_{ij} \). A total of \( N(N + 1) \) calls to \textsc{getCellDimension} must be made to do this, but determining cell dimensions thereafter becomes trivial. We can simply select actions \( i \) such that \( D_{ii} = M - 1 \). Determining \( K \) becomes similarly lightweight. For each \( i \in \mathcal{P}, j \in \mathcal{P}, i < j, \{i, j\} \) is added to \( K \) whenever \( D_{ij} = M - 2. \)

Fix \( \{i, j\} \in K \). Where \( D \) is exploited to improve runtime performance is in the function that determines \( K_{ij}^- \). In CBP, exactly \( N - 2 \) calls to \textsc{getCellDimension} are issued — one to determine \( C_i \cap C_j \cap C_k \) for each \( k \in \mathcal{N} \setminus \{i, j\} \). In an attempt to reduce this number, we exploit two observations. The first is that since \( \{i, j\} \in K \), we have that \( C_i \cap C_j \) is an \( N - 2 \) dimensional polytope by definition. The candidate actions \( k \) eligible for membership in \( K_{ij}^- \) must be such that \( C_i \cap C_j \subset C_k \). Assuming \( \ell_i \neq \ell_j \) for all \( i \neq j \), it is impossible to yield an \((M - 2)\)-dimensional polytope by intersecting three Pareto-optimal actions, so we can consider only actions \( k \in \mathcal{N} \setminus \mathcal{P} \). The second observation is the fact that for \( C_i \cap C_j \cap C_k \neq \emptyset \) to hold, it is necessary that both \( C_i \cap C_k \neq \emptyset \) and \( C_j \cap C_k \neq \emptyset \) also hold. Both predicates can be checked in \( O(1) \) time assuming the dimension matrix \( D \) is available to us. Only when both predicates are checked do we make a single call to \textsc{getCellDimension} on the full intersection. In problem settings where there are few degenerate actions — which happen to the ones most interesting to us — this initialization phase has a reduction in the number of calls to \textsc{getCellDimension} of order \( |\mathcal{P}|^2 N \) after these optimizations.

### 5.2 State Caching

Recall at each time step \( t \), the algorithm makes a pass over all unordered pairs of actions \( \{i, j\} \in K \) and updates the loss difference estimates along with their confidence widths accordingly. If we are sufficiently confident that \( (\ell_i - \ell_j)^\top K x_t > 0 \), the open half-space \( \{p \in \Delta_K : (\ell_i - \ell_j)^\top p < 0 \} \) is removed from our consideration for the remainder of the round. The removal of this set of points requires recomputing \( \mathcal{P} \) as well as \( K \) which, as we have seen in the previous section, is rather expensive.

In CBP, a cache is employed to map the current “state” of the closed polytope \( \tilde{C}(t) \) of candidate cells to the active set of Pareto-optimal actions as well as the active neighbouring action pairs. We make this more succinct. Let \( \mathcal{W}(t) = \{(i, j) \in \mathcal{P} \times \mathcal{P} : \tilde{d}_{ij}(t) > c_{ij}(t)\} \) and let \( k(i', j') \) be the index of element \( \{i', j'\} \) in an arbitrary but fixed ordering over \( \{(i, j) : \{i, j\} \in K\} \). A bitmask \( b(t) = 1 + \sum_{(i,j)\in \mathcal{W}(t)} 2^{k(i,j)} \) is maintained to indicate exactly what half-spaces have been eliminated in this round.

CBP allocates two arrays, each of size \( 2^{|K|} \), so that whenever the bitmask \( b(t) \) has been seen previously, one can simply do a lookup in the \( b(t) \)-th slot of each array to obtain \( \mathcal{P}(t) \) and \( K(t) \), respectively, without having to
compute cell intersections. Of course, in practice, this does not accommodate games with even a moderately large number of actions as $K$ can have as many as $\Theta(N^2)$ many elements. In CBP-SIDE, we continue to utilize bitmask $b(t)$, but instead of maintaining two very large arrays, we maintain two hash tables — one for cached Pareto-optimal action sets and another for cached neighbouring action pairs. Both tables use bitmasks as keys.

5.3 Regression

Recall from the previous chapter the closed-form formulation for the estimators associated with action-specific observation distribution mappings:

$$\tilde{\pi}_i(t) = Y_i^T V_i^{-1}(\lambda_i I_D + X_i^T X_i)^{-1}.$$ 

Let $V_i = \lambda_i I_D + X_i^T X_i$. To avoid the expenses associated with matrix inversion, we made use of Sherman-Morrison updates [23] which allow us the following update rule:

$$V^{-1}_{i(t)} = V^{-1}_{i(s-1)} - \frac{V^{-1}_{i(t)} x_i x_i^T V^{-1}_{i(t)} - V^{-1}_{i(t)} x_i x_i^T V^{-1}_{i(t)}}{1 + x_i V^{-1}_{i(T)} V^{-1}_{i(t)} x_i}.$$

These updates are performed in quadratic time rather than cubic.

5.4 Performance Improvements

It was shown by Bartók and Szepesvári [9] that, given a locally-observable partial monitoring game satisfying $J_t \sim K x_t$, the regret of CBP-SIDE using least-squares estimators $\tilde{\pi}_i$ equipped with action confidence widths $w_i(t) = D(\sqrt{(D+1) \log T} + \sigma_i)x_i^T V_i^{-1} x_i$, gives

$$\mathbb{E}[R_{CBP-SIDE}^T] \leq C_1 N + C_2 N^{3/2} D^2 \sqrt{T} \log T$$

for constants $C_1$ and $C_2$, both dependent on $L$ and $H$. As our contribution, we introduce a new flavour of CBP-SIDE, which makes use of reduced confidence widths whose decay depends solely on past observed side information. These updated widths are specified as

$$w'_i(t) = x_i V^{-1}_{i(t)} x_i.$$ 

Note the width can never increase over time as was the case before.

Despite breaking the theoretical guarantees that were proven by Bartók and Szepesvári [9], the introduction of these widths has been shown to yield performance improvements by many orders of magnitude when run across
5. Extensions

![Figure 5.1: A $2 \times 2$ locally observable game.](image)

![Figure 5.2: A $3 \times 3$ non-locally observable game.](image)

our synthetic experiments. Such experiments included both locally as well as non-locally observable environments.

Figures 5.1 and 5.2 depict the expected regret of both versions of the algorithm run on two different problem instances. Side information was generated u.a.r. from $\Delta_M$ and mapped to the outcome space via the identity. That is, the outcome distribution and side information coincided. To distinguish between the two versions of the algorithm, we refer to the version using conservative widths $w_i$ as CBP-SIDE$^{\text{cons}}$, and the version using empirically-observed high performance widths $w'_i$ as CBP-SIDE$^{\text{perf}}$. Similar results were observed for higher dimensional spaces.

For the remainder of this study, we restrict our attention to the CBP-SIDE$^{\text{perf}}$ algorithm and use the names CBP-SIDE and CBP-SIDE$^{\text{perf}}$ interchangeably.
Chapter 6

Contextual Bandits as Partial Monitoring Games

Here we introduce the problem of contextual bandits with linear payoff functions [14]. This problem can be thought of as a “stepping stone” to partial monitoring games. In what follows, we model the contextual bandit problem as a partial monitoring game so that we may observe how CBP-SIDE performs against leading-edge algorithms within this theoretically easier class of problems before moving on to more challenging ones.

In the experimental sections of this thesis, we compare performance against the LINUCB algorithm [20], designed for general contextual bandit problems with linear side information and proven to achieve asymptotically optimal regret within polylogarithmic factors. We therefore can only hope that the performance of CBP-SIDE is within the same order of magnitude. We show that the performance of the two algorithms on a generic binary loss setting is similar for high-dimensional side information.

We first give a formal description of contextual bandits with linear payoffs and then describe how they can be formulated as a partial monitoring game. Lastly, we provide some experimental comparisons of CBP-SIDE and LINUCB in this domain.

6.1 Contextual Bandits with Linear Payoffs

Contextual bandits with linear payoffs are given as follows. For each round \( t \) and each action (or arm) \( i \), the learner observes feature vectors \( x_{t,i} \in \mathbb{R}^d \) with \( \|x_{t,i}\|_2 \leq 1 \). Then, the learner selects action \( I_t \) while the environment simultaneously selects outcome \( J_t \). We assume there exists an unknown weight vector \( \theta = (\theta_1 \cdots \theta_d)^\top \in \mathbb{R}^d \) with \( \|\theta\|_2 \leq 1 \) so that

\[
\mathbb{E}[s_{I_t,J_t}|x_{t,i}] = \theta^\top x_{t,i}
\]  

(6.1)
for all $i$, where $s_{i,j}$ is the loss of action $i$ at time $t$. We assume the learner receives $s_{i,j'}$ as feedback if and only if he plays action $i'$ at time $t$.

### 6.2 Reconciling the Models

Our goal is to formulate any arbitrary contextual bandit instance as a partial monitoring problem. In this section, we develop one possible interpretation of the outcomes for a general contextual bandit game. We also provide a representation of the side information $x_t$, which can accommodate each arm-specific side information $x_{t,i}$, and a corresponding stochastic matrix $K$ such that $\theta^T x_t = \ell_i K x_t$. For the remainder of this chapter, we omit the subscript $t$ from $x_t, x_t, i, s_{t,i}$, unless the context requires it.

For the model to fulfill the necessary requirements, we let the set of outcomes $\mathcal{M}$ correspond to the set $\{-1, 1\}^N$. Thus, $\mathcal{M} = |\{-1, 1\}^N| = 2^N$. We represent the function mapping action-outcome pairs to losses at each timestep by $L \in \mathbb{R}^{N \times 2^N}$ where the $j$-th column of $L$ is a unique assignment of elements in $\{-1, 1\}$ to each action $i$ in row $i$. In the bandit setting, the losses are provided as feedback and so $H = L$. Further assume the columns of $L$ are lexicographically ordered from left to right, where rows with smaller indices are more significant than those with larger indices. For example, the following loss matrix would be used in the 3-action setting:

$$L = \begin{pmatrix}
-1 & -1 & -1 & 1 & 1 & 1 & 1 \\
-1 & -1 & 1 & 1 & -1 & -1 & 1 \\
-1 & 1 & -1 & -1 & 1 & -1 & 1 \\
-1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}.$$

In plain English, we treat actions as either being “harmful” or “beneficial” under a certain outcome and each outcome is a possible configuration of the world.

Let $x = (x_1^T \cdots x_N^T)^T \in \mathbb{R}^{Nd}$. This representation allows us to “bundle” all arm-specific side information vectors into a single vector as required in the partial monitoring analog. All that remains is to define an appropriate $K$.

Let

$$K = 2^{-N} \left( \theta_1 \ell_1 + 1 \cdots \theta_d \ell_1 + 1 \cdots \theta_1 \ell_N + 1 \cdots \theta_d \ell_N + 1 \right),$$

where $1 \in \mathbb{R}^{2^N}$ is the all-one vector. The next theorem establishes that $K$ is indeed stochastic and that the expected losses under $K$ are equivalent to those of the contextual bandit case.

**Theorem 6.1** Let $K$ be as above and let $(p_1 \cdots p_{2^N})^T = Kx$. Then, all of the following desiderata hold:
6.3. Discussion

In this chapter we provided a representation of the contextual bandit problem as a partial monitoring one by representing the outcomes as the set of all possible “binary loss assignments” to elements of the action set. One limitation of this setup is the difference between the number of degrees of

Figure 6.1: CBP-SIDE versus LinUCB on contextual bandit games with binary losses.

1. $\theta^T x_i = \ell_i^T K x$ for each action $i$,
2. $\sum_{j=1}^{2^N} p_j = 1$,
3. $0 \leq p_j \leq 1$ for each outcome $j$.

The proof of Theorem 6.1 can be found in Appendix A.1.

See Figure 6.1 for a comparison between LinUCB and CBP-SIDE using this newly defined model. The results indicate a drastic advantage of LinUCB over CBP-SIDE as expected.

### 6.3 Discussion

In this chapter we provided a representation of the contextual bandit problem as a partial monitoring one by representing the outcomes as the set of all possible “binary loss assignments” to elements of the action set. One limitation of this setup is the difference between the number of degrees of
freedom inherent to the problem as contrasted with the size of the estimators to be learned. An interesting problem would be to determine whether, given additional structure on $L$ as we have encountered in this chapter, the learning can be sped up via an appropriate representation of estimators.

Recall the regularized least squares estimator

$$\tilde{\pi}_i(t) = \arg\min_{\pi \in \mathbb{R}^\sigma_i \times d} \sum_{s=1}^{n_i(t-1)} \|Y_{t_i(s)} - \pi x_{t_i(s)}\|_2^2 + \lambda_i \|\pi\|_2^2.$$ 

In the contextual bandit setting, the true parameter to be estimated would be

$$\pi^*_i = S_i K = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \theta^T \begin{pmatrix} 0_{d \times d} & \cdots & 0_{d \times d} & I_{d \times d} & 0_{d \times d} & \cdots & 0_{d \times d} \end{pmatrix}.$$ 

Further research into determining what estimators to use as well how their respective confidence widths are to be set would be of particular interest for the general setting where, as in contextual bandits, $\pi^*_i$ is simply a linear transformation of a matrix or vector having few degrees of freedom.

Also note that our conversion was one of potentially many and there may exist a more compact transformation which may improve the learning rate.
Experimental Results on Synthetic Games

In this chapter, we compare CBP-SIDE against several baselines in various synthetic settings. We are chiefly interested in how the algorithm performs given that all of the model assumptions are satisfied, namely that a linear relationship exists between side information and the outcome distribution. In what follows, we give a brief high-level description of the settings and their respective representations within partial monitoring. We then survey two baseline algorithms: Tit-For-Tat, a naive bargainer that does not learn in any sophisticated way, and LINUCB, an upper confidence bound algorithm that we’ve looked at in the previous chapter.

7.1 Settings

We outline two classes of partial monitoring games that we felt would be both practical as well as maximally informative for extrapolating a general classification of regret performance of CBP-SIDE relative to other algorithms. The first game, known as apple tasting, is a locally observable game but consists of some hidden information about the losses. The second game, dynamic pricing, belongs to the hard class of games due to being non-locally observable and globally observable.

7.1.1 Apple Tasting

The apple tasting problem is one of the simplest partial monitoring instances. First proposed in [17], apple tasting can be mapped to a bandit game [2] and, thus, has a minimax regret of $O(\sqrt{NT})$. For our purposes, we extend the setting to include side information.

The idea is as follows. Consider an orchard owner (learner) who is looking to sell his apples to a stream of customers. Each apple can be considered
either “rotten” or “good” and has a particular colour which may provide additional information about the apple’s quality. The owner would like to sell as many good apples and as few rotten apples as he can. To achieve this goal, the owner must either sell an apple or taste an apple. Selling an apple will not provide any information about the probability of an apple being good and will result in the customer being either satisfied or unsatisfied. No customer feedback is presented to the learner. Tasting an apple will provide information about the apple’s quality, but will result in wasted product if the apple was indeed good.

In partial monitoring terms,

\[
L = \begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix}, \quad H = \begin{pmatrix} a & a \\ b & c \end{pmatrix},
\]

(7.1)

where the first row of each matrix corresponds to selling the apple and the second to tasting the apple. The first column in this case corresponds to a rotten apple and the second to a good apple. The numbers \( x > 0 \) and \( y > 0 \) are fixed losses incurred when a bad apple is sold and a good apple is tasted, respectively. The side information (that is, the apple’s colour) is presented in the vector \( x_t \) (for example, this can be a 3-dimensional RGB vector) which is mapped to some subset of outcome distributions via an appropriate linear map \( K \).

To characterize the “hardness” of this game, we require the following.

**Definition 7.1** An admissible transformation to a game \( G = (L, H, K) \) is an operation on \( G \) that performs one of the following:

1. Adds the same value to each entry of a column of \( L \).

2. Replaces all occurrences of a symbol \( s \) in row \( i \) of \( H \) with a different symbol \( t \).

Bartók [8] showed that the minimax regret of a game \( G \) is no larger than the game \( G' \) having been mapped from \( G \) using only admissible transformations. In this case, we call \( G \) easier than \( G' \). If each application of operation 2 preserves differences between symbols in the affected row, the minimax regret of \( G' \) will be the same as \( G \). Here, we say that \( G \) and \( G' \) are equivalent games.

We show that apple tasting is equivalent to a bandit game by simply subtracting \( x \) from the first column of \( L \) and applying the function \( f \) having \( f(a) = 0 \), \( f(b) = -x \) and \( f(c) = y \) to the entries of \( H \).

\[
L' = \begin{pmatrix} 0 & 0 \\ -x & y \end{pmatrix}, \quad H' = \begin{pmatrix} 0 & 0 \\ -x & y \end{pmatrix}
\]

(7.2)

One can also easily verify that apple tasting is a locally observable game.
One may devise a generalized apple tasting problem also consisting of many informative actions and a single uninformative one. Namely,

\[
L_k = \begin{pmatrix}
    x_1 & 0 & \cdots & 0 \\
    0 & x_2 & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & x_k
\end{pmatrix},
\quad
H_k = \begin{pmatrix}
    a & a & \cdots & a \\
    b_1 & b_2 & \cdots & b_k \\
    \vdots & \vdots & \ddots & \vdots \\
    b_1 & b_2 & \cdots & b_k
\end{pmatrix}.
\] (7.3)

where, for each \( i, x_i > 0 \). This game is locally observable in addition to being easier than a bandit game.

### 7.1.2 Dynamic Pricing

In dynamic pricing, a retailer (the learner) wishes to sell a single product in an imperfect market. We assume the product has zero cost of production. At each time \( t \), a new customer arrives. The customers possess some set of characteristics \( x_t \) (which may encode age, gender, etc.). The retailer must decide on an asking price for the good while the customer simultaneously decides her maximum valuation of the good. We assume the asking prices as well as the valuations come from the same finite set of prices. The retailer has one chance to voice an asking price (there is no haggling) and either the customer accepts or rejects without ever revealing her true valuation. That is, the retailer receives only a single bit of information at the end of each round — whether the transaction was successful or unsuccessful. If the asking price is at most the customer’s valuation, then the transaction is considered successful and a loss equal to the price difference between the asking price and the customer’s true valuation is suffered by the learner. Otherwise, the transaction is deemed unsuccessful and a fixed storage cost \( c \) is incurred.

In this instance, we assume the set of possible asking prices is given by \( N = \{0, 1/(N - 1), \ldots, (N - 2)/(N - 1), 1\} \) so that the corresponding partial monitoring game looks as follows:

\[
L = \begin{pmatrix}
    0 & 1/(N - 1) & 2/(N - 1) & \cdots & 1 \\
    c & 0 & 1/(N - 1) & \cdots & (N - 2)/(N - 1) \\
    c & c & 0 & \cdots & (N - 3)/(N - 1) \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    c & c & c & \cdots & 0
\end{pmatrix} \in \mathbb{R}^{N \times N},
\]

\[
H = \begin{pmatrix}
    y & y & \cdots & y \\
    n & y & \cdots & y \\
    \vdots & \vdots & \ddots & \vdots \\
    n & \cdots & n & y
\end{pmatrix} \in \mathbb{R}^{N \times N}.
\] (7.4)

The \( i \)-th action in this game corresponds to offering a price of \( (i - 1)/(N - 1) \) to the customer, while the \( j \)-th outcome corresponds to the scenario where
her true valuation is \((j - 1)/(N - 1)\). Symbols \(y, n \in \Sigma\) represent successful and unsuccessful transactions respectively.

Simple linear algebra reveals that this game is non-locally observable. However, it is globally observable and thus has a minimax regret of \(\Theta(T^{2/3})\).

7.2 Baselines

At the time of this writing, no algorithms designed for partial monitoring games with stochastic side information were known to exist. Obviously, algorithms tailored for those games with a fixed distribution over outcomes would suffer linear regret for appropriately sampled \(x_t\). Thus, it would make little sense to include such algorithms in a comparison with CBP-SIDE. Instead, we focus on baselines that operate under stronger model assumptions, but that also consider \(x_t\) when determining the best action.

7.2.1 Tit-For-Tat

Tit-For-Tat (TFT) is a naive algorithm we developed with the intention to paint a picture of the level of difficulty associated with dynamic pricing as an online learning problem. Of the synthetic experiments we look at in this chapter, we use TFT as a baseline for online dynamic pricing, but we shall see in the following chapter that it also maps nicely to the procurement domain.

The essence of TFT is as follows. Assume \(x_t \in \mathbb{R}^2\). We discretize the side information space into a matrix \(B \in \mathbb{R}^{N-1 \times N-1}\) of buckets. If the first and second components of \(x_t\) fall in the intervals \([(i - 1)/(N - 1), i/(N - 1))\) and \([(j - 1)/(N - 1), j/(N - 1))\) respectively, then \(x_t\) is “assigned” the index \((i, j)\) and our asking price is set to \(B_{ij}\). If the transaction fails, we decrement \(B_{ij}\) for posterity. Similarly, we increment if the transaction succeeds.

Note that TFT is a very lightweight algorithm as it does not need to store the game’s history or perform any type of regression. By the same token, TFT does not learn anything about the outcome distribution, so one would expect it to perform poorly in general.

7.2.2 LinUCB

As a “serious competitor”, we again include in our performance evaluation the LinUCB algorithm, which assumes a contextual bandit model. Recall in section 6.1 we introduced the contextual bandit problem. In this model, we assume that at each time step \(t\) we receive \(N\) side information vectors \(x_{t,i} \in \mathbb{R}^d\). We further assume a linear relationship between the expected reward of action \(i\) and \(x_{t,i}\), realized by the action-independent weight vector
\[ \theta \in \mathbb{R}^d \text{ satisfying} \]
\[ \mathbb{E}[1 - L_{i,j}|x_{t,i}] = \theta^\top x_{t,i}. \quad (7.5) \]

In section 6.2, we wished to translate a contextual bandit problem into a partial monitoring one. Our goal is now reversed. Since we’ve established that partial monitoring is a richer model, we must accept that information loss is inevitable over the course of this translation.

Where we require a bit of ingenuity is in sensibly distributing our single side information vector \( x_t \in \mathbb{R}^D \) considered in partial monitoring across \( N \) different actions to obtain \( x_{t,i} \in \mathbb{R}^d \) for each action \( i \). We assume \( d = (D + 1)N \), and that
\[
(i - 1) \cdot (D + 1) \quad (N - i) \cdot (D + 1)
\]
\[ x_{t,i} = \left( \begin{array}{c} \sum_{j \in \mathcal{U}(1,H_{11})} L_{1j} / |\mathcal{U}(1,H_{11})| \\
\vdots \\
\sum_{j \in \mathcal{U}(N,H_{N1})} L_{Nj} / |\mathcal{U}(N,H_{N1})| \\
\end{array} \right) (\begin{array}{c} x_t^\top \\
0 \cdots 0 \\
\end{array})^\top \quad (7.6) \]

so that the side information vectors reside in homogeneous coordinates and the expected linear payoff of arm \( i \) can be expressed via action-specific weight vectors \( \theta_i \in \mathbb{R}^{D+1} \), where \( \theta^\top = (\theta_1^\top \ldots \theta_N^\top) \). Of course, there may be other ways of representing side information, but this model has enough degrees of freedom to learn \( \theta_i = K^\top \ell_i \) exactly. This comes at a cost of working in a potentially higher-dimensional space, resulting in a slower confidence width decay.

As we’ve seen, apple tasting games are equivalent to bandit games so LIN-UCB is foreseen to have expected regret \( \tilde{O}(\sqrt{T}) \) when fed the losses given in (7.2). However, no such claim exists for dynamic pricing games as these are not even locally observable. We can, however, attempt to “mould” any dynamic pricing problem instance into a contextual bandit problem. The only requirement an approximation \( \mathbf{L}' \) of \( \mathbf{L} \) must make is that \( \mathbf{L}'_{ij} = \mathbf{L}_{ik}' \) whenever \( H_{ij} = H_{ik} \). Our solution is to average over loss values in such a way that the problem then becomes easier than bandit.

**Definition 7.2** Let \( \mathbf{G} = (\mathbf{L}, \mathbf{H}, K) \) be a partial monitoring game. Let \( \mathcal{U}(i,s) = \{ j \in \mathcal{M} : H_{ij} = s \} \). The bandit approximation to \( \mathbf{G} \) is described by \( \mathbf{G}' = (\mathbf{L}', \mathbf{H}', K) \), where
\[
\mathbf{L}' = \left( \begin{array}{cccc}
\sum_{j \in \mathcal{U}(1,H_{11})} L_{1j} / |\mathcal{U}(1,H_{11})| & \cdots & \sum_{j \in \mathcal{U}(1,H_{1M})} L_{1j} / |\mathcal{U}(1,H_{1M})| \\
\vdots & \ddots & \vdots \\
\sum_{j \in \mathcal{U}(N,H_{N1})} L_{Nj} / |\mathcal{U}(N,H_{N1})| & \cdots & \sum_{j \in \mathcal{U}(N,H_{NM})} L_{Nj} / |\mathcal{U}(N,H_{NM})| \\
\end{array} \right)
\]
and \( \mathbf{H}' = \mathbf{L}' \).

Thus, the matrix \( \mathbf{L}' \) in a bandit approximation is simply the result of the averages of the losses within a row that *would have been accrued* given each
feedback symbol in the original feedback matrix $H$. Take dynamic pricing as an example. Whenever we receive feedback $n$, we can deduce that a fixed cost $c$ was incurred since this information is fully revealed by the feedback. However, since receiving the bit $y$ could imply any number of possible losses for a particular action $i$, we average over each one and treat that as the explicit loss.

LinUCB will use the bandit approximations of all non-locally observable games that we will look at, including dynamic pricing in the next section.

7.3 Experiments

We conducted experiments on all of the problems listed above. We describe the setup and provide our results in the remainder of this section.

7.3.1 Apple Tasting

We compared the performance of CBP-SIDE and LinUCB using the mapped bandit losses on four instances of the apple tasting problem. Two instances of the smaller $2 \times 2$ problem from equation (7.1) were used, one where $x = y$ and one where $x = y/10$. We also tested on a larger $3 \times 3$ matrix of the form given in equation (7.3), one where $x = y = z$ and another where $x = y/10$ and $y = z$. We chose these losses so that we could see how the algorithms performed in settings where more informative actions were equally as risky as uninformative ones versus settings where the learner must “pay a price” to learn. See Figure 7.1 for a breakdown of these results. Our side information vectors were generated uniformly at random from $\Delta_M$ and mapped identically over the outcomes.

The results show that CBP-SIDE performs similarly in settings with risk equality and grossly outperforms LinUCB in the harsher learning environment. It is important to note the difference in the order of the expected regret versus the time horizon in all cases. Taking this into consideration, one can see that both algorithms are highly competitive.

7.3.2 Dynamic Pricing

Figure 7.2 shows the results of the comparisons on two dynamic pricing instances, one where the fixed storage cost is “cheap” and the other where it is “costly”. In both instances, $N = 6$.

We included TFT as our second baseline to give an indication of how poorly a naive approach would perform. Note that TFT does not learn the outcome distribution and so, in effect, is “memory-less”. Rather than zeroing in on the best fixed action in hindsight, conditioned on $x_t$, TFT will tend
Figure 7.1: Performance of CBP-SIDE versus LinUCB on apple tasting games.

to constantly oscillate between different actions, causing a linear regret of $\Omega((c + \frac{1}{N^T})T)$.

For LinUCB, we note a “knee” in the regret curve initially, which afterwards tends to flatten out. This is a necessary result of operating with the bandit approximation of $L$, a biased estimate of the true $L$. As the offered price increases, the bandit approximation of the loss becomes closer to the true loss, since failed transactions return the exact losses as feedback, while higher asking prices effectively mean that more outcomes result in failed transactions. In effect, LinUCB learns the expected loss of some actions very rapidly and these correspond to actions where most of the weight of the probability mass function falls on outcomes resulting in failed transactions for that action. In our setting, this happens across the range of more expensive asking prices. This also explains why the knee is more pronounced in the costly version of the game, as higher asking prices tend to result in steep losses, which LinUCB is able to glean onto very quickly. Beyond this, however, LinUCB fails to determine the best side information to fixed action map-
7. Experimental Results on Synthetic Games

![Graphs showing performance comparison between CBP-SIDE, LinUCB, and TFT.](image)

Figure 7.2: Performance of CBP-SIDE versus LinUCB on dynamic pricing games.

ping for lower prices since it fails to leverage the feedback in such a way as to learn the true distribution over outcomes.

Since dynamic pricing is not a locally-observable game, there are as of yet no theoretical guarantees that the expected regret of CBP-SIDE would be sublinear for this problem. However, empirically, we can see this is indeed the case. Clear advantages can be noted in the cheap setting, while convergence on LinUCB’s regret is more delicate in the costly setting. Where CBP-SIDE gains in the long run, it makes up for in the short run, as pairwise estimates must be made over the set of actions, rather than individual (sometimes clearly suboptimal) actions. Nevertheless, in large-scale environments, where the time horizon is on the order of millions, CBP-SIDE is the clear winner.
Case Study: Online Procurement with Worker Utilities

In this chapter, we focus our attention on the problem of online procurement. In the settings of the previous chapter, the learner assumed the role of a merchant while the environment was embodied as a stream of customers. In online procurement, however, the learner wishes to solicit a set of services from a pool of workers. The learner has one chance to state a sufficiently high offer to each worker who may either accept or reject it. The goal of the learner is to consider the past history of the game as well as additional side information to induce the distribution over workers’ private costs. In this chapter, we focus strictly on posted price mechanisms where the learner offers a fixed price and receives either negative or positive feedback. We introduce a novel augmentation to the standard model by introducing worker values. That is, the learner receives some indication of task quality that he may also use to further his cumulative utility.

We first develop the model formally in section (8.1). Then, in section (8.2), we translate it into a partial monitoring game. In doing so, we consider three different methods of interpreting our loss as a learner since the concept of loss is an ambiguous one in this environment. We then prove a surprising result, namely that two seemingly different textbook notions of loss are equivalent under this type of game. We conclude in sections 8.3–8.5 by presenting results of some experiments carried out on both synthetic and real-world data.

8.1 Online Procurement

In online procurement, a requester (the learner) has a set of tasks he wishes to have completed. The tasks are assumed to be of uniform difficulty. There is an ordered stream of $T$ workers willing to work on a single task at a cost
case study: Online Procurement with Worker Utilities

\( c_t \in \mathcal{P} \), where \( \mathcal{P} \subset \mathbb{R}_{\geq 0} \) is a finite set of possible prices. The cost is never revealed to the requester. At each time step \( t \), the requester offers a price \( p_t \in \mathcal{P} \). The worker accepts if \( c_t \leq p_t \) and rejects otherwise. That is, we assume only posted price mechanisms.

Each worker possesses an inherent value (or utility) associated with her ability to perform the task. We denote this value by \( v_t \in \mathcal{V} \), where \( \mathcal{V} \subset \mathbb{R}_{\geq 0} \) is a finite set of possible utilities that the requester may hold for the worker. We assume the prices and values are measured in the same units so that they can be easily compared, though the sets \( \mathcal{P} \) and \( \mathcal{V} \) may be different. If the transaction was successful, the worker completes the task and the requester receives as feedback \( v_t \), an indicator of the quality of the result. The requester gains value \( v_t \) and must pay price \( p_t \). If the transaction was unsuccessful (i.e. the offered price was too low), the requester gains no utility from the worker and is not shown the true value.

As always, the requester receives some side information \( x_t \in \mathbb{R}^D \) about the worker at the beginning of each round.

### 8.2 Procurement as a Partial Monitoring Game

We would like to formulate this problem as a partial monitoring game, but we have yet to actually state our explicit goal as a requester. That is, it must be determined on how to interpret the losses. In this section, we outline three different loss functions we deem to be useful.

First, however, we must specify the structure of the loss and feedback matrices, \( \mathbf{L} \) and \( \mathbf{H} \). We will abuse notation slightly and write \( p_i \) (resp. \( c_i \)) when we mean the \( i \)-th largest price (resp. cost) in \( \mathcal{P} \) and \( v_k \) when we mean the \( k \)-th largest value in \( \mathcal{V} \). The distinction from \( p_t, c_t \) and \( v_t \) should be clear from the context. Recall in dynamic pricing, the \((i,j)\)-th entry of the loss matrix corresponded to the situation where the learner asks for price \( p_i \) and the consumer values it at \( c_j \). The formulation is similar in this setting, however, with the introduction of worker valuations, our matrices blow up by a factor of \(|\mathcal{V}|\) since each price-cost pairing must be considered for each possible value \( v_k \). To accommodate this, we represent our matrices as

\[
\mathbf{L} = \left( \begin{array}{c|c|c|c} L_1 & L_2 & \cdots & L_{|\mathcal{V}|} \end{array} \right), \quad \mathbf{H} = \left( \begin{array}{c|c|c|c} H_1 & H_2 & \cdots & H_{|\mathcal{V}|} \end{array} \right),
\]

where each \( L_k, H_k \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|} \). In this case, our feedback alphabet is \( \Sigma = \{ n, y_1, y_2, \ldots, y_{|\mathcal{V}|} \} \) where \( n \) corresponds to a failed transaction and \( y_k \) to a successful transaction yielding value \( v_k \). The feedback matrix for a given \( v_k \) is given by
As in dynamic pricing, the rows of each matrix correspond to different prices, but the columns are now indexed according to lexicographically ordered \((v_k, c_j)\) pairs. That is, we assume if the worker’s cost is \(c_j\) and her value is \(v_k\), then the loss incurred (resp. feedback received) for offering price \(p_i\) is \(L_i|\mathcal{V}|(k-1)+j\) (resp. \(H_i|\mathcal{V}|(k-1)+j\)).

We now consider three notions of loss that we deem to be interesting. The first two are well-known concepts in economics, known as *accounting loss* and *economic loss*. The third notion treats failed transactions independently by attributing a fixed *idling cost* to rounds where time is “wasted”.

### 8.2.1 Accounting Loss

*Accounting loss* is concerned with only the net loss of a transaction — that is, the difference between the value acquired and the budget spent. In online procurement, the accounting loss for time \(t\) is given by

\[
\ell_{\text{acc}}(p_t, c_t, v_t) = (p_t - v_t) \mathbb{I}_{p_t \geq c_t}
\]

Formulating this as a loss matrix for a fixed value \(v_k\), we get

\[
L_{\text{acc}} = \begin{pmatrix}
    p_1 - v_k & 0 & \ldots & 0 \\
    p_2 - v_k & p_2 - v_k & 0 & \ldots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    p_N - v_k & p_N - v_k & \ldots & p_N - v_k
\end{pmatrix}
\]

### 8.2.2 Economic Loss

As in dynamic pricing, one of our goals over the course of the game is to minimize price inefficiencies. That is, on a particular time step, if the transaction succeeded but the offered price \(p_t\) was larger than the minimum cost \(c_t\) to the worker, a penalty of \(p_t - c_t\) is accrued (of course, this exact quantity is never revealed to the requester). The problem becomes novel when the utilities \(v_t\) are introduced. Here, the issue of minimizing price inefficiencies is coupled with that of maximizing the total acquired value.

We consider profits forgone by an agent having chosen one particular course of action over another more favourable one. *Economic loss* of a given action
8. Case Study: Online Procurement with Worker Utilities

\( i \) is specified as the difference between the accounting loss of \( i \) and the accounting loss of the optimal action. From this definition, we can derive an appropriate version of economic loss for online procurement:

\[
\ell_{\text{eco}}(p_l, c_l, v_l) = \ell_{\text{acc}}(p_l, c_l, v_l) - \min_{p \in P} \ell_{\text{acc}}(p, c_l, v_l)
\]

\[
= (p_l - v_l)I_{p_l \geq c_l} - \min_{p \in P} (p - v_l)I_{p \geq c_l}
\]

\[
= (p_l - v_l)I_{p_l \geq c_l} - ((c_l - v_l) \cdot I_{v_l \geq c_l} + 0 \cdot I_{c_l < v_l})
\]

\[
= (p_l - v_l)I_{p_l \geq c_l} + (v_l - c_l)I_{v_l \geq c_l}
\]

\[
= \max(p_l - v_l, p_l - c_l)I_{p_l \geq c_l} + \max(0, v_l - c_l)I_{p_l < c_l}
\]

Considering economic losses compels the learner to only solicit workers whose cost, if known, would be considered a “bargain”. If \( v_l < c_l \), we treat this as an inferior deal and, so, pricing outside of the worker’s budget is granted 0 loss, while hiring the worker incurs a loss of \( p_l - v_l \) which is guaranteed to be non-negative. If \( v_l \geq c_l \), the transaction should be pursued, so our loss is simply treated as the price differential had the transaction happened and the opportunity cost of \( v_l - c_l \) otherwise.

Fix \( v_k \) such that \( p_{j-1} \leq v_k < p_j \) for some cost \( p_j \). The corresponding economic loss matrix is given by

\[
I_{\text{eco}}^k = \begin{pmatrix}
0 & v_k - p_2 & \cdots & v_k - p_{j-1} & 0 & \cdots & 0 \\
p_2 - p_1 & 0 & \cdots & v_k - p_{j-1} & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
p_{j-1} - p_1 & p_{j-1} - p_2 & \cdots & 0 & 0 & \cdots & 0 \\
p_j - p_1 & p_j - p_2 & \cdots & p_j - p_{j-1} & p_j - v_k & \cdots & p_j - v_k \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
p_{N-1} - p_1 & p_{N-1} - p_2 & \cdots & p_{N-1} - p_{j-1} & p_{N-1} - v_k & \cdots & p_{N-1} - v_k \\
p_N - p_1 & p_N - p_2 & \cdots & p_N - p_{j-1} & p_N - v_k & \cdots & p_N - v_k \\
\end{pmatrix}
\]

8.2.3 Economic Loss with Fixed Idling Costs

Our third model considers settings where time may be limited or where missed transactions incur expenses to the requester. We refer to these expenses as idling costs. If the transaction was successful, we consider the economic losses instead. The loss function for this scenario is given by

\[
\ell_{\text{idle}}(p_l, c_l, v_l) = \ell_{\text{eco}}(p_l, c_l, v_l)I_{p_l \geq c_l} + aI_{p_l < c_l}
\]
for idling cost \( \alpha \). The associated loss matrix is
\[
L^{\text{idle}}_k = \begin{pmatrix}
0 & \alpha & \cdots & \alpha & \alpha & \cdots & \alpha \\
p_2 - p_1 & 0 & \cdots & \alpha & \alpha & \cdots & \alpha \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
p_{j-1} - p_1 & p_{j-1} - p_2 & \cdots & 0 & \alpha & \cdots & \alpha \\
p_j - p_1 & p_j - p_2 & \cdots & p_j - p_{j-1} & p_j - v_k & \cdots & \alpha \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
p_{N-1} - p_1 & p_{N-1} - p_2 & \cdots & p_{N-1} - p_{j-1} & p_{N-1} - v_k & \cdots & \alpha \\
p_N - p_1 & p_N - p_2 & \cdots & p_N - p_{j-1} & p_N - v_k & \cdots & p_N - v_k \\
\end{pmatrix}.
\]

These formulations may be slightly difficult to digest, so we provide a diagram in Figures 8.1 – 8.3. Each graph depicts how our losses are construed by varying the value of our offered price \( p_t \) while keeping the value \( v_t \) and cost \( c_t \) constant.

Now that we have characterized all of the relevant loss functions, we provide a proof that, in fact, two are equivalent from a learning point of view.

**Theorem 8.1** An online procurement game using accounting loss is equivalent to a game using economic loss assuming prices \( P \) and values \( V \) are used in both games. Moreover, both games are easier than a bandit game.

**Proof** To prove the first statement, it suffices to strictly apply operation 1 from Definition 7.1 to \( L^{\text{acc}}_k \) and \( L^{\text{eco}}_k \) to obtain the same matrix. We do this just for an arbitrary \( k \) since each column transformation does not affect the other loss submatrices.

Add \( v_k \) to each column of \( L^{\text{acc}}_k \) to obtain
\[
\hat{L}^{\text{acc}}_k = \begin{pmatrix}
p_1 & v_k & v_k & \cdots & v_k \\
p_2 & p_2 & v_k & \cdots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
p_N & \cdots & \cdots & \cdots & p_N \\
\end{pmatrix}.
\]

To transform \( L^{\text{eco}}_k \), add \( p_j \) to column \( j \) for each index \( j \) such that \( p_{j-1} \leq v_k \). For \( j \) having \( p_j > v_k \), add \( v_k \) to column \( j \). The result \( \hat{L}^{\text{eco}}_k = \hat{L}^{\text{acc}}_k \).

One can easily determine admissible transformations on \( H_k \) to yield a bandit game.

One can further show that games having economic losses with fixed idling costs are not bandit games and furthermore are not locally observable. With these results, we focus on just two flavours of procurement games in our experiments, ones making use of accounting loss and ones using fixed idling costs.
8. Case Study: Online Procurement with Worker Utilities

8.3 Performance on Internally Synthesized Data

In this section, we describe experiments that respected the model assumptions given in Section 3.1.
8.4. Performance on Externally Synthesized Data

8.3.1 Setup

Before moving on to more complex scenarios, we provide results for the case where the side information is generated u.a.r. from the probability simplex and mapped to an outcome distribution using a predefined linear stochastic matrix. Recall “outcomes” as defined in procurement correspond to cost-value pairs associated with a worker.

We ran a total of six experiments on the synthetic data. Half of the experiments were carried out in the setting where the set of values consisted of only a single element, \( V = \{1\} \). The other half consisted of three equally-spaced priced values, \( V = \{1/3, 2/3, 1\} \). In each of these, we ran three experiments: one experiment to measure regret performance using accounting loss as our metric and two experiments using economic loss with fixed idling costs. We vary the idling cost between a “harsh” (\( \alpha = 0.75 \)) and “lenient” setting (\( \alpha = 0.25 \)). The set of prices remained fixed throughout all experiments at \( P = \{0, 1/2, 1\} \).

As in Section 7.3.2, we use LinUCB and TFT as baselines. Again, LinUCB is fed the “zero-padded” side information in homogeneous coordinates. Thus, if \( x_t \in \mathbb{R}^D \), then \( x_{t,i} \in \mathbb{R}^{(D+1)N} \) (see equation (7.6)). LinUCB uses the unaltered (bandit) accounting loss matrices \( L_{\text{acc}}^k \) and the bandit approximations of each instantiation of \( L_{\text{idle}}^k \). TFT operates on procurement in a similar manner as dynamic pricing, but the roles of price increments and decrements are reversed. TFT uses a \( 10 \times 10 \) discretization of the first two dimensions of the side information space.

8.3.2 Results

The results are depicted in Figure 8.4.

Clear performance gains by CBP-SIDE over LinUCB can be noted in most of the cases. As expected, CBP-SIDE performs favourably in non-locally observable settings such as when using fixed idling costs, whereas LinUCB tends to suffer linear loss and, bizarrely, superlinear loss in the “harsh” setting. Remember, LinUCB is using a biased estimate of the losses and so unusual behaviour of its regret with respect to the true loss is possible. To our pleasure, CBP-SIDE performs favourably even on bandit instances — a setting LinUCB is expected to do very well on.

8.4 Performance on Externally Synthesized Data

In an effort to more closely mimic the real-world data we had collected, we ran several experiments on synthetic data generated via a parametrized distribution. The difference between this setup and the previous was that no
8. Case Study: Online Procurement with Worker Utilities

Figure 8.4: Performance on internally synthesized games where $|\mathcal{V}| = 1 ((a),(c),(e))$ and $|\mathcal{V}| = 3 ((b),(d),(f))$. 
knowledge of the relationship between side information and outcome distribution was assumed. The data vectors that were generated acted as proxies for the real data while allowing us to play with various dependencies, unlike in the internal case where the outcome distribution is dependent on all of the observed side information.

### 8.4.1 Setup

We generate four real numbers per time step: a two-dimensional feature vector \( x_t = \left( \frac{x_{t,1}}{x_{t,2}} \right) \) whose entries were sampled u.a.r from the unit interval and then renormalized, a cost sampled from \( P = \{0, 0.1, 0.2, \cdots, 1.0\} \) and a value sampled from \( V = \{1/3, 2/3, 1\} \).

In our synthesis, we enforced only a single correlation in the data, namely between \( v_t \) and \( x_{t,1} \). To generate the data, intermediate counterpart random variables \( v'_t \) and \( c'_t \) were sampled according to \( v'_t \sim \mathcal{N}(x_{t,1}, 0.04^2) \) and \( c'_t \sim \mathcal{U}(0, 1) \). \( c'_t \) and \( v'_t \) were then adjusted to be valid elements \( v_t \in V \) and \( c_t \in P \), respectively. Namely, \( v_t = \arg\min_{v \in V} |v - v'_t| \) and similarly for \( c_t \).

### 8.4.2 Results

See Figure 8.5 for a breakdown of the results.

It is clear that, based on the rift between TFT and its competitors in these experiments, significant performance gains can be made by exploiting non-trivial learning algorithms. Beyond this, there is no clear winner in all three cases. We note that LinUCB is clearly better when minimizing accounting loss, while CBP-SIDE does very well with harsh idling costs. There is no statistically significant difference in performance on the lenient idling costs setting.

### 8.5 Performance on Real Data

To conclude, we provide details of a survey we conducted on the Amazon Mechanical Turk (AMT) platform. We describe how our data was represented and resampled and then take a look at how the algorithms performed on this data set.

#### 8.5.1 Setup

Our ultimate goal was to determine whether CBP-SIDE holds up against LinUCB in real-world procurement settings where \( x_t, c_t \) and \( v_t \) are well-defined and easily determined. To this end, we collected user data from surveys conducted via AMT.
The surveys introduced a hypothetical scenario involving incentives for ad posting to user profiles of a major social media site. All users were required to already be members of this site in order to take part in the survey. The surveys were conducted truthfully in the sense that equal pay was offered to each participant with bonuses awarded to the subset of users who provided answers deemed to be most plausible. The questions influencing the data set were as follows:

1. Would you be interested in posting ads to your social media profile if such a system existed?
2. Roughly how many friends do you have on this service?
3. Roughly how many minutes per day do you spend using this service?
4. Between 1 USD and 500 USD, at what monthly price would you be willing to accept the offer of using the banner of your profile page for advertisements?
We sifted the data according to users who gave positive responses to question 1. Responses to questions 2 and 3 were used as side information, while the responses to question 4 were treated as user costs. Since true user values are difficult to obtain in this context, we injected our own values. We decided a strong relationship should exist between the user’s number of friends and their value. The motivation was that the user would have more “market influence” provided she had a large number of friends. We sorted, in decreasing order, the roughly 800 resulting tuples based on the number of friends and assigned a “high” value \( v_t = 1 \) to the top third, a “medium” value \( v_t = 2/3 \) to the second third and a “low” value \( v_t = 1/3 \) to the remaining third.

We noted no substantial correlation between the side information provided and the costs. All user costs were normalized to the unit interval and then rounded to the nearest element of \( P \). We used sampling with replacement on the user data to ensure a large number of rounds.

### 8.5.2 Results

Cumulative losses for each of the ad posting experiments are shown in Figure 8.6. Unlike with internally synthesized data, the performance of CBP-SIDE appears to be problem-dependent. We tend to see generally better results when values come from a singleton set. This is likely due to the added short-term advantage LinUCB gains by honing in on harmful actions more quickly, as was discussed in section 7.3.2.
8. Case Study: Online Procurement with Worker Utilities

Figure 8.6: Performance on real-world data with injected values where $|\mathcal{V}| = 1$ ((a), (c), (e)) and $|\mathcal{V}| = 3$ ((b), (d), (f)).
Chapter 9

Conclusion

To summarize, we implemented and evaluated one particular instantiation of the CBP-SIDE algorithm. The version we considered made use of linear least squares estimators and assumed the outcome distribution to be a stochastic mapping of the side information. We showed that by adjusting the rate at which the confidence widths decay, we obtain drastically reduced expected regret in practice.

We evaluated this optimized version of CBP-SIDE across several synthetic examples. To include competitive baselines, we provided appropriate reductions from the domain of contextual bandits into the more general partial monitoring setup. We showed that CBP-SIDE is successful in synthetic versions of canonical partial monitoring instances.

We concluded with an in-depth examination of online procurement as a partial monitoring problem. We introduced a novel augmentation to the problem by introducing worker values and provided several notions of loss when considering this new problem. We showed that the accounting and economic losses correspond to easier-than-bandit games. We showed that CBP-SIDE succeeds in all settings where the linear model assumption holds and in real-world settings where the set of possible values is a singleton set.

Regarding future work, one may also consider a more compact set of estimators when operating under certain assumptions about the structure of the losses, that was done for contextual bandits as partial monitoring games. While LinUCB must only learn a $d$-dimensional weight vector $\tilde{\theta}$, CBP-SIDE must estimate $\tilde{\pi}_i \in \mathbb{R}^{2 \times N_d}$ for each $i \in \mathcal{N}$, far more degrees of freedom than is inherent to the problem.

We have also provided an extensive evaluation of CBP-SIDE on non-locally observable games satisfying the global observability condition. Recall this class of games is considered “hard” and has a minimax regret of $\Theta(T^{2/3})$. 
9. Conclusion

Our results indicate that, empirically, CBP-SIDE enjoys sublinear regret, however, these results need to be proven rigorously.

We have seen that for large worker value sets CBP-SIDE performs worse than LinUCB even for loss functions that CBP-SIDE is expected to perform better on in the limit. One reason for this may be due to the short-term advantage LinUCB has by using independent action reward estimates rather than estimates of loss differences between neighbouring actions as used in CBP-SIDE. One direction of research would be to somehow leverage the short-term advantages of such an approach with the long-term advantages of considering loss difference estimates.
Appendix A

Appendix

A.1 Proof of Theorem 6.1

Proof To prove requirement 1, first observe that the rows of of \( L \) comprise an orthogonal basis of a subspace of \( \mathbb{R}^{2N} \). That is, \( \ell_i^\top \ell_j = 0 \) for all \( i \neq j \) and \( \ell_i^\top \ell_i = 2^N \). Also note that each row has an equal number of 1’s as -1’s.

Let \( K_j = 2^{-N} \left( \begin{array}{cccc} \theta_1 \ell_j + 1 & \cdots & \theta_D \ell_j + 1 \end{array} \right) \). Then,

\[
\ell_i^\top K_j = 2^{-N} \left( \begin{array}{cccc} \theta_1 \ell_i^\top \ell_j + \ell_i^\top 1 & \cdots & \theta_D \ell_i^\top \ell_j + \ell_i^\top 1 \end{array} \right) = \begin{cases} 0^\top & \text{if } i \neq j \\ \theta^\top & \text{otherwise}. \end{cases}
\]

Let \( \theta^{(i)} = \left( \begin{array}{cccc} 0 \cdots 0 & \theta^\top \cdots 0 \end{array} \right)^\top \). Since \( K = \left( \begin{array}{cccc} K_1 & \cdots & K_N \end{array} \right) \), we have that

\[
\ell_i^\top Kx = (\theta^{(i)})^\top x = \theta^\top x.
\]

Requirement 2 follows immediately from the fact that all of the columns of \( K \) sum to 1 and it is assumed that side information vectors \( x \) are such that \( \|x\|_1 = 1 \). Employing the fact that any convex combination of distributions is again a distribution, we get our desired result.

Since \( \|\theta\|_\infty \leq 1 \) and \( \ell_i^\top 1 = 0 \) for all \( i \), we have that all elements of \( K \) are nonnegative. Combining this fact with Requirement 2, we get Requirement 3 as desired. \( \square \)


