Frontiers in Artificial Intelligence

TYPE Original Research
PUBLISHED 06 December 2024
DOI 10.3389/frai.2024.1346700

OPEN ACCESS

EDITED BY
Mohan Sridharan,
University of Edinburgh, United Kingdom

REVIEWED BY
Alberto Ochoa Zezzatti,
Universidad Autónoma de Ciudad Juárez, Mexico
Reza Shahbazian,
University of Calabria, Italy
Muhammad Yousuf Jat Baloch,
Jilin University, China
Anam Nigar,
Changchun University of Science and Technology, China

*CORRESPONDENCE
Chandrasekar Subramanian
sekarnet@gmail.com

RECEIVED 29 November 2023
ACCEPTED 19 November 2024
PUBLISHED 06 December 2024

CITATION
Subramanian C and Ravindran B (2024) Causal contextual bandits with one-shot data integration. *Front. Artif. Intell.* 7:1346700.
doi: 10.3389/frai.2024.1346700

Causal contextual bandits with one-shot data integration

Chandrasekar Subramanian[1,2]* and Balaraman Ravindran[1,2]

[1]Robert Bosch Center for Data Science and Artificial Intelligence, Indian Institute of Technology Madras, Chennai, India, [2]Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India

We study a contextual bandit setting where the agent has access to causal side information, in addition to the ability to perform multiple targeted experiments corresponding to potentially different context-action pairs—simultaneously in one-shot within a budget. This new formalism provides a natural model for several real-world scenarios where parallel targeted experiments can be conducted and where some domain knowledge of causal relationships is available. We propose a new algorithm that utilizes a novel entropy-like measure that we introduce. We perform several experiments, both using purely synthetic data and using a real-world dataset. In addition, we study sensitivity of our algorithm's performance to various aspects of the problem setting. The results show that our algorithm performs better than baselines in all of the experiments. We also show that the algorithm is sound; that is, as budget increases, the learned policy eventually converges to an optimal policy. Further, we theoretically bound our algorithm's regret under additional assumptions. Finally, we provide ways to achieve two popular notions of fairness, namely counterfactual fairness and demographic parity, with our algorithm.

KEYWORDS

causality, fairness, causal contextual bandits, causal bandits, contextual bandit algorithm

1 Introduction

Learning to make decisions that depend on context has a wide range of applications—software product experimentation, personalized medical treatments, recommendation systems, marketing campaign design, etc. Contextual bandits (Lattimore and Szepesvári, 2020) have been used to model such problems with good success (Liu et al., 2018; Sawant et al., 2018; Bouneffouf et al., 2020; Ameko et al., 2020). In the contextual bandit framework, the agent interacts with an environment to learn a near-optimal policy that maps a context space to an action space.[1]

A major challenge in wider application of contextual bandits to real world problems is the need for a large number samples, which is often prohibitively costly to obtain; indeed, this is a challenge with reinforcement learning in general (Dulac-Arnold et al., 2021). Typically, this is mitigated by considering special cases and exploiting the structures present in those settings to obtain better algorithms. Subramanian and Ravindran (2022) provided the first approach (and the only one so far) for contextual bandits where a causal graph was present as side information, in addition to the agent having the ability to perform interventions targeted on subgroups.

1 Actions in a contextual bandit setting can also be interpreted as Pearl $do()$ interventions (Zhang and Bareinboim, 2017; Lattimore et al., 2016); so we use the term "action" and "intervention" interchangeably.

Specifically, in many real world scenarios, we often have some causal side information available from domain knowledge. For instance, in software product experimentation, we might know that `os` has a causal effect on `browser`, but not the other way around. Subramanian and Ravindran (2022) showed that exploiting this causal knowledge can yield better contextual bandit policies.[2] They also introduced the notion of targeted interventions, which are interventions targeted on a specific subgroup specified by a particular assignment of values to the context variables.

This work proposes and studies a causal contextual bandits framework that has similarities with the one proposed in Subramanian and Ravindran (2022) where the agent has access to causal side information and can perform targeted interventions; the agent's objective is to minimize *simple regret* (Lattimore and Szepesvári, 2020). However, in contrast to the purely interactive setting in their work, we consider a setting where *multiple targeted interventions can be performed simultaneously in one-shot* at a cost. This, as we will see, fundamentally changes the agent's optimization problem due to the shared causal graph across the different interventions. However, importantly, it also opens up causal contextual bandits to new areas of application where it is possible to acquire additional data in one-shot at a cost and within a budget; some examples are discussed in Section 1.1. Section 1.2 provides a non-mathematical description of our framework, while Section 2.1 provides a mathematical specification of the problem. Section 1.4 provides a comparison of this work with related work.

To the best of our knowledge, there has not been any investigation on what the best way is for causal contextual bandit agents is to obtain additional experimental data in one shot and incorporate it with aim of learning a good policy.

1.1 Motivating examples

In software product development, product teams are frequently interested in learning the best software variant to show or roll out to each subgroup of users. To achieve this, it is often possible to conduct targeting at scale *simultaneously*, i.e., in *one shot*, for various combinations of contexts (representing user groups) and actions (representing software variants)—instead of one experiment at a time.[3] These targeted experiments can be used to compute relevant metrics (e.g., click-through rate) for each context-action pair. Further, we might have some qualitative domain knowledge of how some context variables are causally related to others; for example, we might know that `os` has a causal relationship to `browser`; we would like to exploit this knowledge to learn better policies. This can be naturally modeled as a question of "given the causal graph, what *table* of targeted experimental data needs to be acquired and how to integrate that data, so as to learn a good policy?" here the table's rows and columns are contexts and actions, and each cell specifies the number of data samples (zero or more) required corresponding to the context-action pair. At the end of the above training phase, the agent moves to an evaluation phase (e.g., when it is deployed), where it observes contexts, and takes actions using the learned policy.

As another example, in online marketing campaign strategy, the objective is to learn the best campaign to show to each group of people.[4] This is done by conducting experiments where different groups of people are shown different marketing campaigns simultaneously, and learning from the resultant outcomes. Further, letting context variables model the features of groups, we might have some knowledge about how some of the features are causally related to others; for example, we might know that `country` causally affects `income`. Our framework provides a natural model for this setting. Further, we might also be interested in ensuring that the campaigns meet certain criteria for fairness. For example, there might be some variables such as `race` that could be sensitive from a fairness perspective, and we would not want the agent to learn policies that depends on these variables. We discuss fairness implications of our algorithm in Section 5. These are just two of many scenarios where this framework provides a natural model. Two additional examples include experimental design for ads[5] and recommendation systems.[6]

1.1.1 Remark

The ideas and approach provided in this paper are not restricted to the examples mentioned above, and can be more generally applied by the wider scientific community. The scientific method, especially in the natural and social sciences, often involves performing experiments with real world entities such as people, animals or objects. Whenever there are opportunities for conducting multiple experiments in parallel (e.g., multiple people conducting social interventions at the same time on different groups) and there is some knowledge of causal relationships between the attributes of those entities (e.g., relationships between demographic attributes of beneficiaries), the framework and approach mentioned in this paper can be considered. The algorithm in this paper provides an efficient approach for designing the parallel experiments in these settings. However, the ethical aspects of such experiments have to be accounted for, before using in practice in such cases.

1.2 Our framework

Our framework captures the various complexities and nuances described in the examples in Section 1.1. We present an overview here; please see Section 2.1 for the mathematical formalism. The agent's interactions consist of a learning phase where the agent incurs no regret, followed by an evaluation phase where regret is measured; this is also called *simple regret* (Lattimore and Szepesvári, 2020). At the start of the learning phase, the agent is given a

2 Similar results for multi-armed bandits for best arm identification problems are shown in works such as Lattimore et al. (2016).

3 See: https://firebase.google.com/docs/ab-testing for an example.

4 See: https://www.persado.com/articles/the-power-of-experimental-design-to-deliver-marketing-insights/ for an example.

5 For example, see: https://support.google.com/google-ads/answer/1704368

6 For example, see: https://scale.com/blog/netflix-recommendation-personalization

(possibly empty) log of offline data—consisting of context-action-reward tuples—generated from some unknown policy. The context variables are partitioned into two sets—the main set and the auxiliary set (possibly empty). The agent observes all context variables during this phase, but learns a policy that only depends on the main set of context variables; this also provides a way to ensure that the learned policy meets certain definitions of fairness (see Section 5 for a more detailed discussion). Further, the agent also has some qualitative[7] causal side-information available, likely from domain knowledge. This causal side-information is encoded as a causal graph between contextual variables. A key implication of the causal graph is information leakage (Lattimore et al., 2016; Subramanian and Ravindran, 2022)—getting samples for one context-action pair provides information about other context-action pairs because of shared pathways in the causal graph.

Given the logged data and the causal graph, the agent's problem is to decide the set of targeted experimental samples to acquire within a budget, and then integrate the returned samples. More specifically, the agent is allowed to make a *one-shot* request for data in the form of a table specifying the number of samples it requires for each context-action pair, subject to the total cost being within the budget. The environment then returns the requested samples after conducting the targeted interventions, and the agent integrates those samples to update its internal beliefs and learned policy. After this learning phase, the agent moves to an inference or evaluation phase, where it returns an action (according to the learned policy) for every context it encounters; its regret is measured at this point. The core problem of the agent is to choose these samples in a way that straddles the trade-off between choosing more samples for context-action pairs it knows is likely more valuable (given its beliefs) and choosing more samples to explore less-seen context-action pairs—while taking into account the budget and the information leakage across all obtained samples arising from the causal graph.

1.3 Contributions

1. This is the first work to study how to *actively obtain and integrate* a table of *multiple samples in one-shot* in a contextual bandit setting. Further, we study this in the presence of a causal graph, making it one of the very few works to study the *utilization of causal side-information* by contextual bandit agents. See Section 1.4 for a more detailed discussion on related work, and Section 2.1 for the mathematical formalism of the problem.
2. We propose a novel algorithm (Section 2.2.3) that works by minimizing a new entropy-like measure called $\Upsilon(.)$ that we introduce. See Section 2.2 for a full discussion on the approach.
3. We show results of extensive experiments using purely synthetically generated data and an experiment inspired by real-world data, that demonstrate that our algorithm performs better than baselines. We also study sensitivity of the results to key aspects of the problem setting. See Section 4.
4. We also show some theoretical results. Specifically, we show that the method is sound – that is, as the budget tends to infinity, the algorithm's regret converges to 0 (Section 3.2). Further, we provide a bound on regret for a limited case (Section 3.1).
5. We discuss fairness implications of our method in Section 5. We show that it can achieve counterfactual fairness. Further, while the algorithm does not guarantee demographic parity, we provide a way to recover this notion of fairness, but with a reduction in performance.

1.4 Related work

Causal bandits have been studied in the last few years (e.g., Lattimore et al., 2016; Yabe et al., 2018; Lu et al., 2020), but they study this in a multi-armed bandit setting where the problem is identification of one best action. There is only one work (Subramanian and Ravindran, 2022) studying causal *contextual* bandits—where the objective is to learn a *policy* mapping contexts to actions—and this is the closest related work. While we do leverage some ideas introduced in that work in our methodology and in the design of experiments, our work differs fundamentally from this work in important ways. Subramanian and Ravindran (2022) consider a standard interactive setting where the agent can repeatedly act, observe outcomes and update its beliefs, whereas in our work the agent has a one-shot data collection option for samples from *multiple* context-action pairs. This fundamentally changes the nature of the optimization problem as we will see in Section 2.2; it also makes it a more natural model in a different set of applications, some of which were discussed in Section 1.1. Further, our work allows for arbitrary costs for collecting those samples, whereas they assume every intervention is of equal cost.

Contextual bandits in purely offline settings, where decision policies are learned from logged data, is a well-studied problem. Most of the work involves inverse propensity weighting based methods (such as Swaminathan and Joachims, 2015b,a; Joachims et al., 2018). Contextual bandits are also well-studied in purely interactive settings (see Lattimore and Szepesvári, 2020 for a discussion on various algorithms). However, in contrast to our work, none of these methods can integrate causal side information or provide a way to actively acquire and integrate new targeted experimental data.

Active learning (Settles, 2012) studies settings where an agent is allowed to query an oracle for ground truth labels for certain data points. This has been studied in supervised learning settings where the agent receives ground truth feedback; in contrast, in our case, the agent receives outcomes only for actions that were taken ("bandit feedback"). However, despite this difference, our approach can be viewed as incorporating some elements of active learning into contextual bandits by enabling the agent to acquire additional samples at a cost. There has been some work that has studied contextual bandits with costs and budget constraints (e.g., Agrawal and Goyal, 2012; Wu et al., 2015). There has also been work that has explored contextual bandit settings where the agent can not immediately integrate feedback from the environment, but can do so only in batches (Zhang et al., 2022; Ren et al., 2022; Han et al., 2020). However, all these works consider settings where the samples

7 By qualitative, we mean that the agent can know the causal *graph*, but not the conditional probability distributions of the variables. See Section 2.1 for a more detailed discussion.

TABLE 1 Summary of key notation.

Notation	Meaning
X	Action variable
Y	Reward variable
$\mathcal{C}^A, \mathcal{C}^B$	Set of main context variables and set of auxiliary context variables, respectively; so the set of all context variables is $\mathcal{C} = \mathcal{C}^A \cup \mathcal{C}^B = \{..., C_i, ...\}$.
Capital letters	A random variable; e.g., C_1 or X
Small letters	A random variable's value; e.g., c_1 or x
Small bold font	An assignment of values to a set of random variables; for example, $\mathbf{c}$ denotes a specific choice of values taken by variables in $\mathcal{C}$
$\hat{\mathbb{P}}, \hat{\mathbb{E}}$	Estimate of distribution $\mathbb{P}$ and expectation $\mathbb{E}$ based on current beliefs
$\mathsf{val}(V), \mathsf{val}(\mathcal{V})$	Set of values taken by the variable V, and set of variables $\mathcal{V}$, respectively.
$\hat{\phi}, \phi^*$	The learned policy and an optimal policy, respectively
Υ	Entropy-like measure used in our algorithm; defined in Equation 3
$\mathbf{pa}_V$	Value of variables in PA_V, the parents of V
$N_{x,\mathbf{c}^A}$	Number of samples requested corresponding to $X = x$ and $\mathcal{C}^A = \mathbf{c}^A$
$\beta(x, \mathbf{c}^A, N_{x,\mathbf{c}^A})$	Cost of acquiring $N_{x,\mathbf{c}^A}$ samples corresponding to $X = x$ and $\mathcal{C}^A = \mathbf{c}^A$
B	budget
$\mathbf{a}\langle\mathcal{B}\rangle$	If $\mathbf{a}$ is an assignment of values to $\mathcal{A}$, then $\mathbf{a}\langle\mathcal{B}\rangle$ is assignment of those values to respective variables $\mathcal{B}$; $\mathbf{a}\langle\mathcal{B}\rangle = \emptyset$ if $\mathcal{A} \cap \mathcal{B} = \emptyset$.

are obtained through a standard contextual bandit interaction—observe a context, choose an intervention, receive a reward; in contrast, our work considers *targeted* interventions where context values to determine the targeted subgroup is specified along with the intervention. Further, importantly, none of these works provide a way to integrate causal side information.

2 Methodology

2.1 Problem formalism

2.1.1 Underlying model

We model the underlying environment as a causal model $\mathcal{M}$, which is defined by a directed acyclic graph $\mathcal{G}$ over all variables (the "causal graph") and a joint probability distribution $\mathbb{P}$ that factorizes over $\mathcal{G}$ (Pearl, 2009b; Koller and Friedman, 2009). The set of variables in $\mathcal{G}$ consists of the action variable (X), the reward variable (Y), and the set of context variables ($\mathcal{C}$). Each variable takes on a finite, known set of values; note that this is quite general, and accommodates categorical variables. $\mathcal{C}$ is partitioned into the set of main context variables ($\mathcal{C}^A$) and the set of (possibly empty) auxiliary context variables ($\mathcal{C}^B$). That is, $\mathcal{C} = \mathcal{C}^A \cup \mathcal{C}^B$.

The agent knows only $\mathcal{G}$ but not $\mathcal{M}$; therefore, the agent has no *a priori* knowledge of the conditional probability distributions (CPDs) of the variables.

2.1.2 Protocol

In addition to knowing $\mathcal{G}$, the agent also has access to logged offline data, $\mathcal{D}_L = \{(\mathbf{c}_i, x_i, y_i)\}$, where each $(\mathbf{c}_i, x_i, y_i)$ is sampled from $\mathcal{M}$ and x_i is chosen following some unknown policy. Unlike many prior works, such as Swaminathan and Joachims, 2015b, the agent here does *not* have access to the logging propensities.

The agent then specifies in one shot the number of samples $N_{x,\mathbf{c}^A}$ it requires for each pair $(x, \mathbf{c}^A)$.[8] We denote the full table of these values by $\mathbf{N} \triangleq \bigcup_{x,\mathbf{c}^A}\{N_{x,\mathbf{c}^A}\}$. Given a $(x, \mathbf{c}^A)$, there is an arbitrary cost $\beta(x, \mathbf{c}^A, N_{x,\mathbf{c}^A})$ associated with obtaining those samples. The total cost should be at most a budget B. For each $(x, \mathbf{c}^A)$, the environment returns $N_{x,\mathbf{c}^A}$ samples of the form $(\mathbf{c}^B, y) \sim \mathbb{P}(\mathcal{C}^B, Y \mid do(x), \mathbf{c}^A)$.[9] Let's call this acquired dataset $\mathcal{D}_A$. The agent utilizes $\mathcal{D}_A$ along with $\mathcal{D}_L$ to learn a good policy.

2.1.3 Objective

The agent's objective is to learn a policy $\hat{\phi} : \mathsf{val}(\mathcal{C}^A) \to \mathsf{val}(X)$ such that expected simple regret is minimized:

$$Regret \triangleq \sum_{\mathbf{c}^A} \left[\mu^*_{\mathbf{c}^A} - \hat{\mu}_{\mathbf{c}^A}\right] \cdot \mathbb{P}(\mathbf{c}^A)$$

where ϕ^* is an optimal policy, $\mu^*_{\mathbf{c}^A} \triangleq \mathbb{E}[Y|do(\phi^*(\mathbf{c}^A)), \mathbf{c}^A)]$ and $\hat{\mu}_{\mathbf{c}^A} \triangleq \mathbb{E}[Y|do(\hat{\phi}(\mathbf{c}^A)), \mathbf{c}^A)]$.

Table 1 provides a summary of the key notation used in this paper.

2.1.4 Assumptions

We assume that X has exactly one outgoing edge, $X \to Y$, in $\mathcal{G}$. This is suitable to express a wide range of problems such as personalized treatments or software experimentation where the problem is to learn the best action under a context, but the action or treatment does not affect context variables. We also make a commonly-made assumption (see Guo et al., 2020) that there are no unobserved confounders. Similar to Subramanian and Ravindran (2022), we make an additional assumption that simplifies the factorization in Section 2.2.2: $\{C$ confounds $C' \in \mathcal{C}^A$ and $Y\} \implies C \in \mathcal{C}^A$; a *sufficient* condition for this to be true is if $\mathcal{C}^A$ is ancestral.[10] This last assumption is a simplifying assumption and can be relaxed in the future.

8 In contrast, in a standard contextual bandit setting, the agent would repeatedly observe $\mathbf{c}^A$, respond with an x, and observe outcomes.

9 $do(X = x)$ is a standard operation in causal inference that models interventions, where parents of X are removed, and X is set equal to x. See Pearl (2009b, 2019) for more discussion on the $do()$ operation. $do(X = x)$ is succinctly written $do(x)$.

10 That is, if $\mathcal{C}^A$ contains all its ancestors.

2.2 Solution approach

2.2.1 Overall idea

In our approach, the agent works by maintaining beliefs[11] regarding every conditional probability distribution (CPD) in $\mathcal{M}$. It first uses $\mathcal{D}_L$ to update its initial CPD beliefs; this, in itself, makes use of information leakage provided the causal graph. It next needs to choose $\mathcal{D}_A$, which is the core problem. The key tradeoff facing the agent is the following: it needs to choose between allocating more samples to context-action pairs that it believes are more valuable and to context-action pairs that it knows less about. Unlike Subramanian and Ravindran (2022), it cannot interactively choose and learn, but instead has to choose the whole $\mathcal{D}_A$ in one shot—necessitating the need to account for multiple overlapping information leakage pathways resulting from the multitude of samples. In addition, these samples have a cost to acquire, given by an arbitrary cost function, along with a total budget.

2.2.1.1 Toward solving this

To achieve this, we define a novel function $\Upsilon(\mathbf{N})$ that captures a measure of overall entropy weighted by value. The idea is that minimizing Υ results in a good policy; that is, the agent's problem now becomes that of minimizing Υ subject to budget constraints. In Section 2.2.2, we formally define $\Upsilon(\mathbf{N})$ and provide some intuition. Later, we provide experimental support (see Section 4), along with some theoretical grounding to this intuition (see Section 3).

2.2.2 The optimization problem

Determine $N_{x,\mathbf{c}^A}$ for each $(x, \mathbf{c}^A)$ such that

$$\Upsilon(\mathbf{N})$$

is minimized, subject to

$$\sum_{x,\mathbf{c}^A} \beta(x, \mathbf{c}^A, N_{x,\mathbf{c}^A}) \leq B$$

where $\mathbf{N} \triangleq \bigcup_{x,\mathbf{c}^A} \{N_{x,\mathbf{c}^A}\}$. We will next define $\Upsilon(\mathbf{N})$.

2.2.2.1 Defining the objective function $\boldsymbol{\Upsilon}$(N)

The conditional distribution $\mathbb{P}(V|\mathbf{pa}_V)$ for any variable V is modeled as a categorical distribution whose parameters are sampled from a Dirichlet distribution (the belief distribution). That is, $\mathbb{P}(V|\mathbf{pa}_V) = \mathsf{Cat}(V; b_1, ..., b_r)$, where $(b_1, ..., b_r) \sim \mathsf{Dir}(\theta_{V|\mathbf{pa}_V})$, and $\theta_{V|\mathbf{pa}_V}$ is a vector denoting the parameters of the Dirichlet distribution.

Actions in a contextual bandit setting can be interpreted as *do*() interventions on a causal model (Zhang and Bareinboim, 2017; Lattimore et al., 2016). Therefore, the reward Y when an agent chooses action x against context $\mathbf{c}^A$ can be thought of as being sampled according to $\mathbb{P}[Y|do(x), \mathbf{c}^A]$. Under the assumptions described in Section 2.1.4, we can factorize as follows:

$$\mathbb{E}[Y|do(x), \mathbf{c}^A] = \sum_{\mathbf{c}^B \in \mathsf{val}(\mathcal{C}^B)} \left[\mathbb{P}(Y = 1|x, \mathbf{c}\langle PA_Y \rangle) \prod_{c \in \mathbf{c}^B} \mathbb{P}(C = c|\mathbf{c}\langle PA_C \rangle) \right] \quad (1)$$

Crucially, note that our beliefs about each CPD in Equation 1 are affected by samples corresponding to multiple $(x, \mathbf{c}^A)$ due to the shared terms in the factorization. To capture this, we construct a CPD-level uncertainty measure which we call $Q(.)$:

$$Q(\mathbb{P}[V|\mathbf{pa}_V], \mathbf{N}) \triangleq \sum_{x,\mathbf{c}^A} \left(\frac{1}{1 + \ln(N_{x,\mathbf{c}^A} + 1)} \right)$$
$$\mathsf{Ent}^{new}(\mathbb{P}[V|\mathbf{pa}_V]) \Big|_{x\langle PA_V \rangle = \mathbf{pa}_V\langle X \rangle,\ \mathbf{c}^A\langle PA_V \rangle = \mathbf{pa}_V\langle \mathcal{C}^A \rangle} \quad (2)$$

Here Ent^{new} is defined in the same way as in Subramanian and Ravindran (2022), which we reproduce here. Let the length of the vector $\theta_{V|\mathbf{pa}_V}$ be r. Let $\theta_{V|\mathbf{pa}_V}[i]$ denote the i'th entry of $\theta_{V|\mathbf{pa}_V}$. We define an object called Ent that captures a measure of our knowledge of the CPD:

$$\mathsf{Ent}(\mathbb{P}(V|\mathbf{pa}_V)) \triangleq -\sum_i \left[\frac{\theta_{V|\mathbf{pa}_V}[i]}{\sum_j \theta_{V|\mathbf{pa}_V}[j]} \ln \left(\frac{\theta_{V|\mathbf{pa}_V}[i]}{\sum_j \theta_{V|\mathbf{pa}_V}[j]} \right) \right]$$

We then define

$$\mathsf{Ent}^{new}(\mathbb{P}(V|\mathbf{pa}_V)) \triangleq \frac{1}{r} \sum_i \mathsf{Ent}(\mathsf{Cat}(b_1', ..., b_r'))$$

where $(b_1', ..., b_r') \sim \mathsf{Dir}\left(..., \theta_{V|\mathbf{pa}_V}[i-1],\ \theta_{V|\mathbf{pa}_V}[i] + 1,\ \theta_{V|\mathbf{pa}_V}[i+1], ...\right)$.

Finally, we construct $\Upsilon(\mathbf{N})$ as:

$$\Upsilon(\mathbf{N}) \triangleq \sum_{x,\mathbf{c}} \left[\left[\sum_{V \in \mathcal{C}^B \cup \{Y\}} Q(\mathbb{P}[V|\mathbf{c}\langle PA_V \rangle], \mathbf{N}) \right] \cdot \hat{\mathbb{P}}(\mathbf{c}) \cdot \hat{\mathbb{E}}\left[Y|x, \mathbf{c}\langle PA_Y \rangle\right] \right] \quad (3)$$

2.2.2.2 Intuition behind Q(.) and $\boldsymbol{\Upsilon}$(.)

Intuitively, Ent^{new} provides a measure of entropy if *one* additional sample corresponding to $(x, \mathbf{c}^A)$ is obtained and used to update beliefs. $Q(.)$ builds on it and captures the fact that the beliefs regarding any CPD $\mathbb{P}[V|\mathbf{pa}_V]$ can be updated using information leakage[12] from samples corresponding to *multiple* $(x, \mathbf{c}^A)$; it does this by selecting the relevant $(x, \mathbf{c}^A)$ pairs making use of the causal graph $\mathcal{G}$ and aggregating them. In addition, $Q(.)$ also captures the fact that entropy reduces non-linearly with the number of samples. Finally, $\Upsilon(\mathbf{N})$ provides an aggregate (weighted) resulting uncertainty from choosing $N_{x,\mathbf{c}^A}$ samples of each $(x, \mathbf{c}^A)$. The weighting in $\Upsilon(.)$ provides a way for the agent to relatively prioritize context-action pairs that are higher-value according to its beliefs.

For further intuition, note that $Q(\mathbb{P}[V|\mathbf{pa}_V], \mathbf{N})$ captures the resultant uncertainty in the agent's knowledge of $\mathbb{P}[V|\mathbf{pa}_V]$ if samples as specified by $\mathbf{N}$ are obtained and integrated into its beliefs. $\Upsilon(\mathbf{N})$ not only aggregates the CPD-level measure $Q(.)$ to sum over all CPDs, but also adds a weighting factor $\hat{\mathbb{P}}(\mathbf{c})\hat{\mathbb{E}}\left[Y|x, \mathbf{c}\langle PA_Y \rangle\right]$. This term ensures that the algorithm does not overallocate samples to improve knowledge of CPDs that are less "valuable" (i.e., either the context is very unlikely to be seen, or the rewards are too low).

11 "Belief" is a commonly used term to mean the posterior distribution of the parameters of the distribution of interest (Russo et al., 2017).

12 Information leakage arises from shared pathways in $\mathcal{M}$; or equivalently, due to shared CPDs in the factorization of $\mathbb{P}$.

2.2.3 Algorithm

The full learning algorithm, which we call CoBA, is given as Algorithm 1 (Figure 1A). After learning, the algorithm for inferencing on any test context (i.e., returning the action for the given context) is given as Algorithm 2 (Figure 1B). The core problem (Step 2 in Algorithm 1) is a nonlinear optimization problem with nonlinear constraints and integer variables. It can be solved using any of the various existing solvers; refer Section 2.2.3.1 for details on the solver used in our experiments.

2.2.3.1 Solving the optimization problem in Step 2 of Algorithm 1

In our experiments in Section 4, we use the `scipy.optimize.differential_evolution` solver[13] from the `scipy` Python library to solve the problem in Section 4.2. This solver implements differential evolution (Storn and Price, 1997), which is an evolutionary algorithm which makes very few assumptions about the problem. The `scipy.optimize.differential_evolution` method is quite versatile; for example, it allows the specification of the objective function as a Python callable, and also allows arbitrary nonlinear constraints of type `scipy.optimize.NonlinearConstraint`. However, a practitioner can use any suitable optimization algorithm or heuristic to solve this problem.

3 Theoretical results

We first show a regret bound for our algorithm under additional assumptions; we leave a more general bound for future work. In addition, we also show a soundness result – without any assumptions – guaranteeing convergence in the limit. Importantly, in Section 4, we perform extensive experiments where we relax the assumptions made for the regret bound and demonstrate our algorithm's empirical performance.

3.1 Regret bound (under additional assumptions)

In Theorem 3.1, we are interested in the case where B is finite. This is of interest in practical settings where the budget is usually small. We prove a regret bound under additional assumptions (A2) which we describe below. Define $m \triangleq \min_{x,\mathbf{c}^A} \pi(x|\mathbf{c}^A)$, where π is the (unknown) logging policy that generated $\mathcal{D}_L$; and $M_{\mathcal{V}} \triangleq |\text{val}(\mathcal{V})|$.

Theorem 3.1 (Regret bound). Under the additional assumptions (A2) mentioned below, for any $0 < \delta < 1$, with probability $\geq 1-\delta$,

$$Regret \in O\left(\sqrt{\left(\frac{M_{\mathcal{C}}}{mB-\epsilon}\right) \ln \frac{M_X M_{\mathcal{C}}}{\delta}}\right)$$

where $\epsilon \in O\left(\sqrt{B \ln (M_X M_{PA_Y} M_{\mathcal{C}}/\delta)}\right)$, ignoring terms that are constant in B, m, δ, $|\mathcal{C}|$ and the number of possible context-action pairs.

Proof. The proof closely follows the regret bound proof in Subramanian and Ravindran (2022) and adapts it to our setting. The purpose of the proof is to establish an upper bound on performance, and not to provide a tight bound. Bounding regret without these additional assumptions (A2) is left for future work.

First, we define the assumptions (A2) under which the theorem holds:

1. There is some non-empty past logged data ($|\mathcal{D}_L| > 0$), and it was generated by an (unknown) policy π where every action has a non-zero probability of being chosen ($\pi(x|\mathbf{c}^A) > 0, \forall x, \mathbf{c}^A$). The latter is a commonly made assumption, for example, in inverse-propensity weighting based methods.
2. $|\mathcal{D}_L| \geq \alpha B$, for some constant $\alpha > 0$. This is generally achievable in real world settings since we usually have fairly large logged datasets (or it is quite cheap to acquire logged data; for example, think of search logs), and for the bound to hold we technically can have a very small α as long as it is >0. We also assume that B is finite, as discussed earlier.
3. The cost function β is constant; without loss of generality, we let this constant be equal to 1. This is a common case in real world applications, especially when we do not have estimates of cost; in those cases, we typically assign a fixed cost to all targeted experiments.

3.1.1 Expression for overall bound

First, note that Equation 3 in Subramanian and Ravindran (2022) remains the same even for our case. This is because it depends only on the factorization of $\mathbb{E}[Y|do(x), \mathbf{c}^A]$ (see Equation 1 in the main paper) and on the fact that in the evaluation phase the agent uses expected parameters of the CPDs (derived from its learned beliefs) to return an action for a given context.

Therefore, suppose, with probability $\geq 1 - \delta_{X,\mathbf{pa}_Y}$,

$$|\forall x, \hat{\mathbb{P}}(Y=1|X,\mathbf{pa}_Y) - \mathbb{P}(Y=1|X,\mathbf{pa}_Y)| \leq \epsilon_{X,\mathbf{pa}_Y}$$

and with probability $\geq 1 - \delta_{C|\mathbf{pa}_C}$,

$$\forall c,\ |\hat{\mathbb{P}}(C=c|\mathbf{pa}_C) - \mathbb{P}(C=c|\mathbf{pa}_C)| \leq \epsilon_{C|\mathbf{pa}_C}$$

where the expressions for $\delta_{X,\mathbf{pa}_Y}$, $\delta_{C|\mathbf{pa}_C}$, $\epsilon_{X,\mathbf{pa}_Y}$ and $\epsilon_{C|\mathbf{pa}_C}$ will be derived later in this section.

Then with probability $\geq 1 - \sum_{\mathbf{pa}_Y} \delta_{X,\mathbf{pa}_Y} - \sum_{C\in\mathcal{C}} \sum_{\mathbf{pa}_C} \delta_{C|\mathbf{pa}_C}$, for any given $\mathbf{c}^A$,

$$\text{Regret}(\mathbf{c}^A) = \mathbb{E}[Y|do(a^*), \mathbf{c}^A] - \mathbb{E}[Y|do(a_{alg}), \mathbf{c}^A] \leq 2\epsilon'_X + 3 \sum_{C\in\mathcal{C}^B} \epsilon'_C \tag{4}$$

where we define

$$\epsilon'_X \triangleq \sum_{\mathbf{pa}_Y} \mathbb{P}(\mathbf{pa}_Y|\mathbf{c}^A)\epsilon_{X,\mathbf{pa}_Y}$$

and

$$\epsilon'_C \triangleq \sum_{\mathbf{pa}_C} \mathbb{P}(\mathbf{pa}_C|\mathbf{c}^A)\epsilon_{C|\mathbf{pa}_C}$$

13 https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html

A

```
Algorithm 1: Learning phase of CoBA
  Data: Causal graph $\mathcal{G}$; initial dataset $\mathcal{D}_L$
  Initialization: For all $V \in \mathcal{C} \cup \{Y\}$ and for all $\mathbf{pa}_V$, set $\theta_{V|\mathbf{pa}_V} = (1, ..., 1)$.
  Update all beliefs using initial dataset $\mathcal{D}_L$ by calling update_beliefs($\mathcal{D}_L, \mathcal{C}$)
  Solve the following
      $$\mathbf{N} = \arg\min_{\mathbf{N}'} \Upsilon(\mathbf{N}')$$
    subject to
      $$\sum_{x,\mathbf{c}^A} \beta(x, \mathbf{c}^A, N'_{x,\mathbf{c}^A}) \leq B$$
  Place data request $\mathbf{N}$; environment returns dataset $\mathcal{D}_A$ as discussed in Section 2.1.
  Update all beliefs using $\mathcal{D}_A$ by calling update_beliefs($\mathcal{D}_A, \mathcal{C}^B$)
  Result: Final set of beliefs for all $V, \mathbf{pa}_V$: $\{..., \theta_{V|\mathbf{pa}_V}, ...\}$
  Procedure update_beliefs($\mathcal{D}, \mathcal{V}$)
    for each sample $(x, \mathbf{c}, y) \in \mathcal{D}$ do
      let $\tilde{\mathbf{c}} \triangleq \mathbf{c} \cup \{x, y\}$
      for each $V \in \mathcal{V} \cup \{Y\}$ do
        $\theta_{V|\tilde{\mathbf{c}}\langle PA_V\rangle}[\tilde{\mathbf{c}}\langle V\rangle] \leftarrow \theta_{V|\tilde{\mathbf{c}}\langle PA_V\rangle}[\tilde{\mathbf{c}}\langle V\rangle] + 1$
      end
    end
```

B

```
Algorithm 2: Inference phase of CoBA
  Data: Causal graph $\mathcal{G}$, learned beliefs $\{..., \theta_{V|\mathbf{pa}_V}, ...\}$, test context $\mathbf{c}^A$
  for every $V$, $\mathbf{pa}_V$ do
    for $v \in \mathsf{val}(V)$ do
      Set $\hat{\mathbb{P}}(V = v|\mathbf{pa}_V) = \frac{\theta^{(v)}_{V|\mathbf{pa}_V}}{\sum_{v'} \theta^{(v')}_{V|\mathbf{pa}_V}}$
    end
  end
  for $x \in \mathsf{val}(X)$ do
    Compute $\hat{\psi}(x, \mathbf{c}^A) \triangleq \hat{\mathbb{E}}[Y|do(x), \mathbf{c}^A]$ using $\hat{\mathbb{P}}$ in Equation (1)
  end
  Result: Return $\hat{\phi}(\mathbf{c}^A) \triangleq \arg\max_x \hat{\psi}(x, \mathbf{c}^A)$
```

FIGURE 1
The learning **(A)** and inference **(B)** phases of our algorithm CoBA. After the learning phase is complete, the agent can be deployed to perform inference as per the inference algorithm.

3.1.2 Expressions for $\delta_{C|\mathsf{pa}_C}$ and $\epsilon_{C|\mathsf{pa}_C}$

Denote $M_{\mathcal{V}} \triangleq |\mathsf{val}(\mathcal{V})|$. Let $L_{\mathbf{pa}_C}$ be the number of samples in $\mathcal{D}_L$ where $PA_C = \mathbf{pa}_C$. Now, our starting estimate of $\hat{\mathbb{P}}(C = 1|\mathbf{pa}_C)$ using $\mathcal{D}_L$ is computed as $(\theta^{(1)}_{C|\mathbf{pa}_C} + 1)/(L_{\mathbf{pa}_C} + 2)$. Since $\mathcal{D}_L$ is built by observing C^A according to the natural distribution and choosing X according to some (unknown) policy, the proof of Lemma A.1 in Subramanian and Ravindran (2022) can be followed if we replace T' by αB since $|\mathcal{D}_L| \geq \alpha B$.

Therefore, suppose, with probability at least $1 - \delta^L_{C|PA_C}$, it is true that

$$\forall \mathbf{pa}_C,\ L_{\mathbf{pa}_C} \geq \alpha B \mathbb{P}(\mathbf{pa}_C, \mathbf{c}^A) - \epsilon^L_{PA_C}$$

If the above event is true, then it is also true that with probability at least $1 - \delta_{C|\mathbf{pa}_C}$, it is true that

$$\forall c,\ |\hat{\mathbb{P}}(c|\mathbf{pa}_C) - \mathbb{P}(c|\mathbf{pa}_C)| \leq \sqrt{\left[\frac{2}{\alpha B \mathbb{P}(\mathbf{pa}_C, \mathbf{c}^A) - \epsilon^L_{C|PA_C}}\right] \ln\left(\frac{2}{\delta_{C|\mathbf{pa}_C}}\right)}$$

where

$$\epsilon^L_{C|PA_C} = \sqrt{\left[\frac{\alpha B}{2}\right] \ln\left(\frac{M_{PA_C}}{\delta^L_{C|PA_C}}\right)},\ \ M_{PA_C} = \prod_{C \in PA_C} M_C$$

Therefore, we have that

$$\epsilon_{C|\mathbf{pa}_C} = \sqrt{\left[\frac{2}{\alpha B\mathbb{P}(\mathbf{pa}_C, \mathbf{c}^A) - \epsilon^L_{C|PA_C}}\right] \ln\left(\frac{2}{\delta_{C|\mathbf{pa}_C}}\right)} \tag{5}$$

3.1.3 Expressions for $\delta_{X,\mathbf{pa}_Y}$ and $\epsilon_{X,\mathbf{pa}_Y}$

Let $L_{x,\mathbf{pa}_Y}$ be the number of samples in $\mathcal{D}_L$ where $(X, PA_Y) = (x, \mathbf{pa}_Y)$. As before, recollect that our estimate of $\hat{\mathbb{P}}(Y = 1|x, \mathbf{pa}_Y)$ is computed as $(\theta^{(1)}_{Y|x,\mathbf{pa}_Y} + 1)/(L_{x,\mathbf{pa}_Y} + 2)$. Further, the mean of $L_{x,\mathbf{pa}_Y}$ is *at least* $|\mathcal{D}_L| \cdot \mathbb{P}(\mathbf{pa}_Y, \mathbf{c}^A) \cdot m > \alpha Bm\mathbb{P}(\mathbf{pa}_Y, \mathbf{c}^A)$, where $m = \min_{x,\mathbf{c}^A} \pi(x|\mathbf{c}^A)$ and π is the unknown logging policy. From our set of assumptions (A2), we have that $\pi(x|\mathbf{c}^A) > 0, \forall x, \mathbf{c}^A$; therefore, $m > 0$.

Given this, the proof of Lemma A.2 in Subramanian and Ravindran (2022) can be followed.

Therefore, suppose, with probability at least $1 - \delta^L_{X,PA_Y}$, it is true that

$$\forall (x, \mathbf{pa}_Y),\ L_{x,\mathbf{pa}_Y} \geq \alpha Bm\mathbb{P}(\mathbf{pa}_Y, \mathbf{c}^A) - \epsilon^L_{X,PA_Y}$$

where

$$\epsilon^L_{X,PA_Y} = \sqrt{\left[\frac{\alpha B}{2}\right] \ln\left(\frac{M_X M_{PA_Y}}{\delta^L_{X,PA_Y}}\right)}$$

If the above event is true, then it is also true that with probability at least $1 - \delta_{X,\mathbf{pa}_Y}$, it is true that

$$\forall x, |\hat{\mathbb{P}}(Y = 1|x, \mathbf{pa}_Y) - \mathbb{P}(Y = 1|x, \mathbf{pa}_Y)| \leq \sqrt{\left[\frac{2}{\alpha Bm\mathbb{P}(\mathbf{pa}_Y, \mathbf{c}^A) - \epsilon^L_{X,PA_Y}}\right] \ln\left(\frac{2M_X}{\delta_{X,\mathbf{pa}_Y}}\right)}$$

Therefore, we have that

$$\epsilon_{X,\mathbf{pa}_Y} = \sqrt{\left[\frac{2}{\alpha Bm\mathbb{P}(\mathbf{pa}_Y, \mathbf{c}^A) - \epsilon^L_{X,PA_Y}}\right] \ln\left(\frac{2M_X}{\delta_{X,\mathbf{pa}_Y}}\right)} \tag{6}$$

3.1.4 Final bound

Now, we can plug the Equations 5, 6 back into Equation 4, and following the same union bound trick as in Subramanian and Ravindran (2022) and some algebra, we get that for any $0 < \delta < 1$, with probability $\geq 1 - \delta$,

$$\begin{aligned} Regret \leq\ & 3\mathbb{E}_{\mathbf{pa}_Y,\mathbf{c}^A}\left(\sqrt{\left[\frac{2}{\alpha mB\mathbb{P}(\mathbf{pa}_Y, \mathbf{c}^A) - \epsilon^L_{X,PA_Y}}\right] \ln\left(\frac{2M_X(M_\mathcal{C} + |\mathcal{C}|)}{\delta}\right)}\right) \\ & + 3\sum_{C \in \mathcal{C}^B} \mathbb{E}_{\mathbf{pa}_C,\mathbf{c}^A}\left(\sqrt{\left[\frac{2}{\alpha B\mathbb{P}(\mathbf{pa}_C, \mathbf{c}^A) - \epsilon^L_{PA_C}}\right] \ln\left(\frac{2(M_\mathcal{C} + |\mathcal{C}|)}{\delta}\right)}\right) \end{aligned} \tag{7}$$

where

$$\begin{aligned} \epsilon^L_{PA_C} &= \sqrt{\left[\frac{\alpha B}{2}\right] \ln\left(\frac{M_{PA_C}(M_\mathcal{C} + |\mathcal{C}|)}{\delta}\right)}, \\ \epsilon^L_{X,PA_Y} &= \sqrt{\left[\frac{\alpha B}{2}\right] \ln\left(\frac{M_X M_{PA_Y}(M_\mathcal{C} + |\mathcal{C}|)}{\delta}\right)} \end{aligned}$$

It can be simplified as presented in **Theorem 3.1** as:

$$Regret \in O\left(\sqrt{\left(\frac{M_\mathcal{C}}{mB - \epsilon}\right) \ln \frac{M_X M_\mathcal{C}}{\delta}}\right)$$

where

$$\epsilon \in O\left(\sqrt{B \ln \frac{M_X M_{PA_Y} M_\mathcal{C}}{\delta}}\right)$$

This completes the proof.

3.1.5 Discussion

The bound is inversely related to $\sqrt{B}$, which is analogous to the inverse relation to $\sqrt{T}$ that other algorithms (albeit for different, but related, settings) such as Lattimore et al. (2016) and Subramanian and Ravindran (2022) have. The regret is inversely related to m, which can be interpreted as a measure of "hardness" of a problem (a larger m intuitively means the problem instance is easier); similar approaches to defining bounds in terms of hardness of the problem instance has been used in other such as Lattimore et al. (2016), Sen et al. (2017), and Yabe et al. (2018). The size of the space of possible interventions is $M_X M_\mathcal{C}$. The bound grows with $\sqrt{M_\mathcal{C} \ln(M_X M_\mathcal{C})}$. However, note that $m = \min_{x,\mathbf{c}^A} \pi(x|\mathbf{c}^A)$. This means, that even if the logging policy that generated $\mathcal{D}_L$ was random (the best case), then $m = 1/M_X$ and it forces another $\sqrt{M_X}$ term into the bound, thereby making the bound grow at the rate of $\sqrt{M_X M_\mathcal{C} \ln(M_X M_\mathcal{C})}$; if m is smaller than $1/M_X$, then the bound would grow faster.

3.2 Soundness

Section 3.1 looked at the case where B is finite. In contrast, in this section, we consider the case where $B \to \infty$. **Theorem 3.2** demonstrates the soundness of our approach by showing that as the budget increases, the learned policy will eventually converge to an optimal policy.

Theorem 3.2 (Soundness). As $B \to \infty$, $Regret \to 0$.

Proof. We would like to show that $Regret \to 0$ as $B \to \infty$. As $B \to \infty$, in the limit, the problem becomes unconstrained minimization of $\Upsilon(\mathbf{N})$. Note that for all $\mathbf{N}$, $\Upsilon(\mathbf{N}) \geq 0$. Therefore, the smallest possible value of $\Upsilon(\mathbf{N})$ is 0.

First, note that $N_{x,\mathbf{c}^A} \to \infty, \forall (x, \mathbf{c}^A) \implies \Upsilon(\mathbf{N}) \to 0$. This is because $\forall (x, \mathbf{c}^A)$,

$$N_{x,\mathbf{c}^A} \to \infty \implies \frac{1}{1 + \ln(N_{x,\mathbf{c}^A} + 1)} \to 0$$

which, in turn, makes $Q(\mathbb{P}[V|\mathbf{pa}_V], \mathbf{N}) \to 0, \forall (V, \mathbf{pa}_V)$. From Equation 3, it is easy to see that this causes $\Upsilon(\mathbf{N}) \to 0$.

Also note that $\Upsilon(\mathbf{N}) \to 0 \implies N_{x,\mathbf{c}^A} \to \infty, \forall (x, \mathbf{c}^A)$. To see this, let $\Upsilon(\mathbf{N}) \to 0$, and consider the case where there exists a $(x, \mathbf{c}^A)$ such that $N_{x,\mathbf{c}^A}$ is finite. That means that there is at least one term of the form

$$\frac{1}{1 + \ln(N_{x',\mathbf{c}^{A\prime}} + 1)}$$

which occurs in the $Q(.)$ function of at least one CPD $\mathbb{P}[V|\mathbf{pa}_V]$, causing $Q(\mathbb{P}[V|\mathbf{pa}_V],\mathbf{N}) > 0$ since $\mathsf{Ent}^{new} > 0$. This, in turn, causes $\Upsilon(\mathbf{N}) \not\to 0$, resulting in a contradiction. Note that this makes an implicit technical assumption that $\mathbb{P}[\mathbf{c}] > 0, \forall \mathbf{c}$ and that $\min(\mathsf{val}(Y)) > 0$; these are stronger assumptions than necessary, and could be weakened in the future.

Thus, $N_{x,\mathbf{c}^A} \to \infty, \forall(x,\mathbf{c}^A) \iff \Upsilon(\mathbf{N}) \to 0$. In other words, each $(x,\mathbf{c}^A)$ gets a number of samples tending toward infinity if and only if Υ tends to 0. Thus, since $\Upsilon(\mathbf{N}) \geq 0$, Algorithm 1 will allocate $N_{x,\mathbf{c}^A} \to \infty, \forall(x,\mathbf{c}^A)$. Each CPD has at least one $(x,\mathbf{c}^A)$ whose samples will be used to update its beliefs in Algorithm 1 (due to the technical assumption mentioned above). This means that each CPD will have its beliefs updated a number of times approaching infinity. Thus, for any $(V,\mathbf{pa}_V)$, $\hat{\mathbb{P}}[V|\mathbf{pa}_V] \to \mathbb{P}[V|\mathbf{pa}_V]$. As a result, we have that $\hat{\mathbb{P}} \to \mathbb{P}$.

Since the agent's policy constructed using $\mathbb{P}$ will necessarily be optimal, we have that *Regret* $\to 0$. This completes the proof.

4 Experimental results

4.1 Baselines and experimental setup

4.1.1 Baselines

There are no existing algorithms that directly map to our setting. Therefore, we construct a set of natural baselines and study the performance of our algorithm CoBA against them. These baselines cover standard strategies for allocation without a way to capture information leakage explicitly (which our algorithm exploits). EqualAlloc allocates an equal number of samples to all $(x,\mathbf{c}^A)$; this provides a good distribution of samples to all context-action pairs. MaxSum maximizes the *total* number of samples summed over all $(x,\mathbf{c}^A)$. PropToValue allocates a number of samples to $(x,\mathbf{c}^A)$ that is proportional to $\hat{\mathbb{P}}(\mathbf{c}^A)\cdot\hat{\mathbb{E}}[Y|do(x),\mathbf{c}^A]$; this allocates relatively more samples to context-action pairs that are more "valuable" based on the agent's current beliefs. All baselines first involve updating the agent's starting beliefs regarding the CPDs of $\mathcal{M}$ using $\mathcal{D}_L$ (same as Step 1 of Algorithm 1) before allocating samples for active obtainment as detailed above. After $\mathcal{D}_A$ is returned by the environment, all baselines use it update their beliefs (same as Step 4 of Algorithm 1).

4.1.2 Experiments

Similar to Subramanian and Ravindran (2022), we consider a causal model $\mathcal{M}$ whose causal graph $\mathcal{G}$ consists of the following edges: $C_1 \to C_0$, $C_0 \to X$, $C_0 \to Y$, $X \to Y$. We let $\mathcal{C}^A = \{C_1\}$ and $\mathcal{C}^B = \{C_0\}$.

The causal graph is illustrated in Figure 2. We use this causal graph for all experiments except Experiment 3.

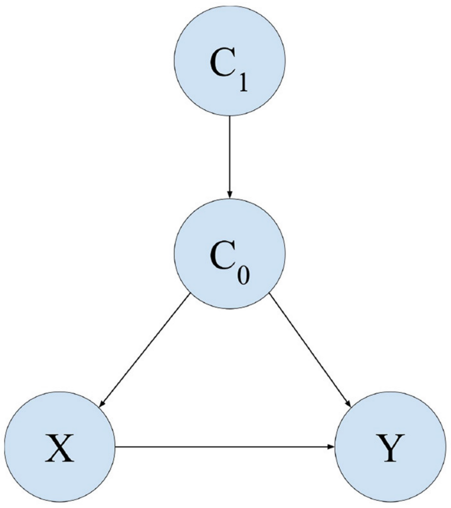

FIGURE 2
Causal graph used in all experiments except Experiment 3.

Experiments 1 and 2 analyze the performance of our algorithm in a variety of settings, similar to those used in Subramanian and Ravindran (2022). Experiment 3 analyzes the performance of the algorithm on a setting calibrated using real-world CRM sales-data provided in Subramanian and Ravindran (2022). For details of all the parameterizations, please refer to the Supplementary material. Experiments 4 through 7 analyze sensitivity of our algorithm's performance to various aspects of the problem setting. In all experiments, except Experiment 6, we set the cost function $\beta(.)$ to be proportional to the number of samples – a natural definition of cost; in Experiment 6, we analyze sensitivity to cost function choice. Section 4.6 reports results of Experiment 1 and 2 for larger values of B (until all algorithms converge), providing empirical evidence of our algorithm's improved *asymptotic behavior*. Further experiments providing more insights into *why* our algorithm performs better than baselines are discussed in Section 4.7. A few additional experiments for intuition are provided in the Supplementary material.

4.1.3 Remark

If the specific parameterization of $\mathcal{M}$ were given *a priori*, it is possible to come up with an algorithm that performs optimally in that particular setting. However, the objective is to design a method that performs well *overall* without this *a priori* information. Consider the relative performance of the baselines in Experiments 2 and 3. We will see that while EqualAlloc performs better than MaxSum and PropToValue in Experiments 3 (Section 4.4), it performs worse than those two in Experiment 2 (Section 4.3). However, our algorithm performs better than all three baselines in all experiments, corroborating our algorithm's overall better performance.

4.2 Experiment 1 (representative settings)

Different parameterizations of $\mathcal{M}$ can produce a wide range of possible settings. Given this, the first experiment studies the performance of our algorithm over a set of "*representative settings*." Each of these settings has a natural interpretation; for example, $\mathcal{C}^A$ could represent the set of person-level features that we are learning a personalized treatment for, or it could represent the set of customer attributes over which we're learning a marketing policy.

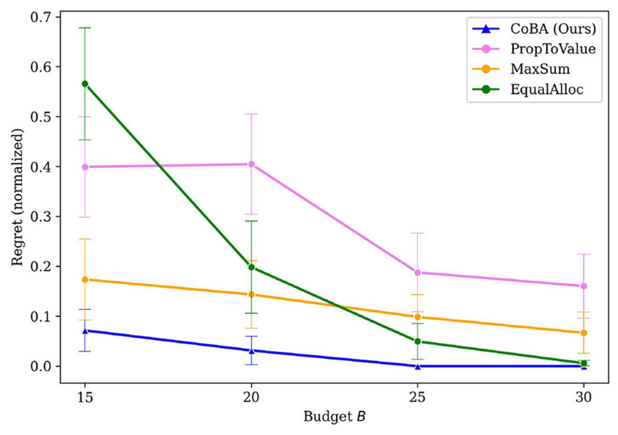

FIGURE 3
Mean regrets normalized to [0, 1] for Experiment 1 (Section 4.2). Our algorithm performs better than all baselines, with gap in performance being higher for smaller budgets.

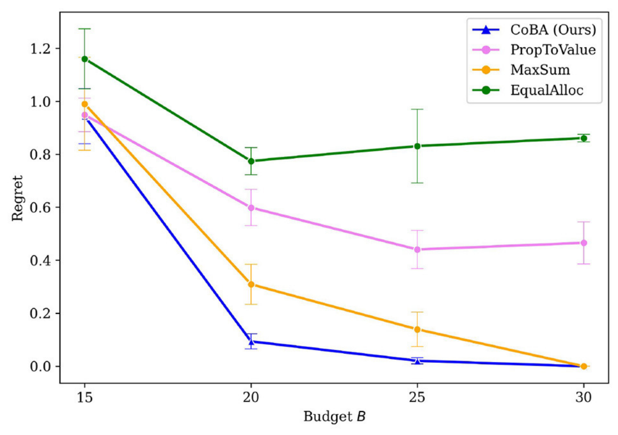

FIGURE 4
Mean regrets for Experiment 2 (Section 4.3). Our algorithm performs better than all baselines.

The settings capture the intuition that high-value contexts (contexts for which, if the optimal action is learned, high expected rewards accrue to the agent) occur relatively less frequently (say, 20% of the time), but that there can be variation in other aspects. Specifically, the variations come from the number of different values of $\mathbf{c}^A$ over which the 20% probability mass is spread, and in how "risky" a particular context is (e.g., difference in rewards between the best and worst actions). For details of the parameterizations, please refer to the Supplementary material. The number of samples in the initial dataset $\mathcal{D}_L$ is kept at $0.5 \cdot |\mathsf{val}(\mathcal{C}^A)| \cdot |\mathsf{val}(X)|$. We consider a uniformly exploring logging policy for $\mathcal{D}_L$; that is, context variables for each sample are realized as per the natural distribution induced by $\mathcal{M}$, but X is chosen randomly. In each run, the agent is presented with a randomly selected setting from the representative set. Results are averaged over 50 independent runs; error bars display ± 2 standard errors.

Figure 3 provides the results of this Experiment. It plots the value of regret (normalized to [0, 1] since different settings have different ranges for regret) as budget B increases. We see that our algorithm performs better than all baselines. Our algorithm also retains its relatively lower regret at all values of B, providing empirical evidence of overall better regret performance.

4.3 Experiment 2 (randomized parameters)

To ensure that the results are not biased due to our choice of the representative set in Experiment 1, this experiment studies the performance of our algorithm when we *directly randomize the parameters* of the CPDs in each run, subject to realistic constraints. Specifically, in each run, we (1) randomly pick an $i \in \{1, ..., \lfloor |\mathsf{val}(C_1)|/2 \rfloor\}$, (2) distribute 20% of the probability mass randomly over the smallest i values of C_1, and (3) distribute the remaining 80% of the mass over the remaining values of C_1. The smallest i values of C_1 have higher value (i.e., the agent obtains higher rewards when the optimal action is chosen) than the other C_1 values. Intuitively, this captures the commonly observed 80–20 pattern (for example, 20% of the customers often contribute to around 80% of the revenue); but we randomize the other aspects. For details of all the parameterizations, please refer to the Supplementary material. Averaging over runs provides an estimate of the performance of the algorithms on expectation. The number of samples in the initial dataset $\mathcal{D}_L$ is kept at $0.25 \cdot |\mathsf{val}(\mathcal{C}^A)| \cdot |\mathsf{val}(X)|$. The results are averaged over 50 independent runs; error bars display ± 2 standard errors.

Figure 4 shows that our algorithm performs better than all baselines in this experiment. Our algorithm also demonstrates overall better regret performance by achieving the lowest regret for every choice of B.

4.4 Experiment 3 (calibrated using real-world data)

While Experiments 1 and 2 study purely synthetic settings, this experiment seeks to study the performance of our algorithm in *realistic scenarios*. We use the same causal graph used in the real world-inspired experiment in Section 4.2 of Subramanian and Ravindran (2022) and calibrate the CPDs using the data provided there. For parameterizations, refer to the Supplementary material.

The objective is to learn a policy that can assist salespeople by learning to decide how many outgoing calls to make in an ongoing deal, given just the type of deal and size of customer, so as to maximize a reward metric. The variables are related to each other causally as per the causal graph [Figure 3a in Subramanian and Ravindran (2022)]. The number of samples in the initial dataset $\mathcal{D}_L$ is kept at $0.125 \cdot |\mathsf{val}(\mathcal{C}^A)| \cdot |\mathsf{val}(X)|$. The results are averaged over 50 independent runs; error bars display ± 2 standard errors. Figure 5 shows the results of the experiment. Our algorithm performs better than all other algorithms in this real-world inspired setting as well. Further, it retains its better performance at every value of B.

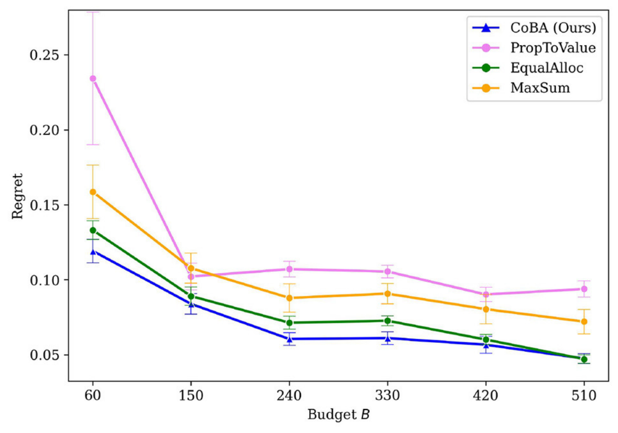

FIGURE 5
Results of the real-world data inspired experiment (Experiment 3, Section 4.4). Our algorithm achieves better mean regrets than baselines.

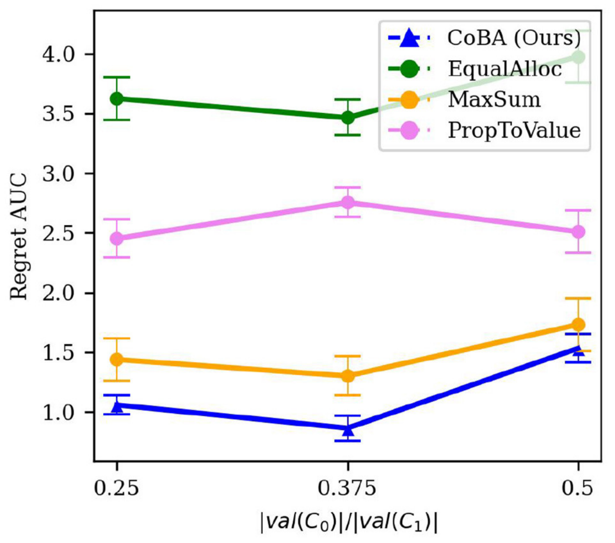

FIGURE 6
Results of the experiment studying robustness to narrowness of $\mathcal{M}$ (Experiment 4, Section 4.5.1). It shows the relation between $|\text{val}(C_0)|/|\text{val}(C_1)|$ and Regret AUC. Our algorithm's performance remains similar (within each other's the confidence interval) for lower values of $|\text{val}(C_0)|/|\text{val}(C_1)|$, but significantly worsens when $|\text{val}(C_0)|/|\text{val}(C_1)| = 0.5$. However, our algorithm continues to perform better than all baselines for all values of $|\text{val}(C_0)|/|\text{val}(C_1)|$.

4.5 Experiments 4 through 7 (robustness to problem setting)

Experiments 4 through 7 study *sensitivity of the results to key aspects* that define our settings. To aid this analysis, instead of regret, we consider a more aggregate measure which we call AUC. For any run, AUC is computed for a given algorithm by summing over B the regrets for that algorithm; this provides an approximation of the area under the curve (hence the name). We then study the sensitivity of AUC to various aspects of the setting or environment.

4.5.1 Experiment 4 ("narrowness" of $\mathcal{M}$)

We use the term "narrowness" informally. Since our algorithm CoBA exploits the information leakage in the causal graph, we expect it to achieve better performance when there is more leakage. To see this, suppose we do a forward sampling (Koller and Friedman, 2009) of $\mathcal{M}$; then, intuitively, more leakage occurs when more samples require sampling overlapping CPDs. For this experiment, we proxy this by varying $|\text{val}(C_0)|$ while keeping $|\text{val}(C_1)|$ fixed. The rest of the setting is the same as in Experiment 2. A lower $|\text{val}(C_0)|$ means that the causal model is more "squeezed" and there is likely more information leakage. The results are averaged over 50 independent runs; error bars display ± 2 standard errors.

Figure 6 shows the results of this experiment. We see that our algorithm's performance remains similar (within each other's the confidence interval) for $|\text{val}(C_0)|/|\text{val}(C_1)| \in \{0.25, 0.375\}$, but significantly worsens when $|\text{val}(C_0)|/|\text{val}(C_1)| = 0.5$. However, our algorithm continues to perform better than all baselines for all values of $|\text{val}(C_0)|/|\text{val}(C_1)|$.

4.5.2 Experiment 5 (size of initial dataset)

The number of samples in the initial dataset $\mathcal{D}_L$ would impact the algorithm's resulting policy, for any given B. Specifically, we would expect that as the cardinality of $\mathcal{D}_L$ increases, regret reduces. For this experiment, we consider a uniformly exploring logging policy, and vary $|\mathcal{D}_L|$ by setting it to be $k \cdot |\text{val}(\mathcal{C}^A)| \cdot |\text{val}(X)|$, where $k \in \{0, 0.25, 0.5\}$. The rest of the setting is the same as in Experiment 2. The results are averaged over 50 independent runs; error bars display ± 2 standard errors.

The results are shown in Figure 7. We would expect the performance of all algorithms improve with increase in k since that would give the agent better starting beliefs; this, indeed, is what we observe. Importantly, our algorithm performs better than all baselines in all these settings. Figure 7 broken down by B is provided in the Supplementary material.

4.5.3 Experiment 6 (choice of β)

Though we allow the cost function to be arbitrary, this experiment studies our algorithm's performance under two natural choices of $\beta(.)$ to test its robustness: (1) a constant cost function; that is, $\beta(x, \mathbf{c}^A, N_{x,\mathbf{c}^A}) \propto N_{x,\mathbf{c}^A}$, and (2) cost function that is inversely proportional to the likelihood of observing the context naturally (i.e., rarer samples are costlier); that is, $\beta(x, \mathbf{c}^A, N_{x,\mathbf{c}^A}) \propto \frac{N_{x,\mathbf{c}^A}}{\mathbb{P}[\mathbf{c}^A]}$. The rest of the setting is the same as in Experiment 2. The results are averaged over 50 independent runs; error bars display ± 2 standard errors.

Figure 8 shows the results. As expected, the choice of cost function does affect performance of all algorithms. However, our algorithm performs better than all algorithms for both cost function choices.

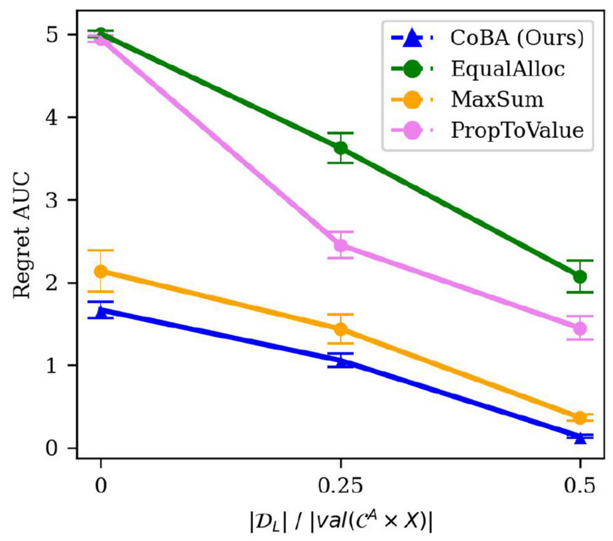

FIGURE 7
Results of the experiment studying robustness to size of the initial dataset (Experiment 5, Section 4.5.2). It shows the relation between $|\mathcal{D}_L|/|\text{val}(\mathcal{C}^A \times X)|$ and Regret AUC. As expected, the performance of all algorithms improve with increase in $|\mathcal{D}_L|$. Importantly, our algorithm performs better than all baselines in all these settings.

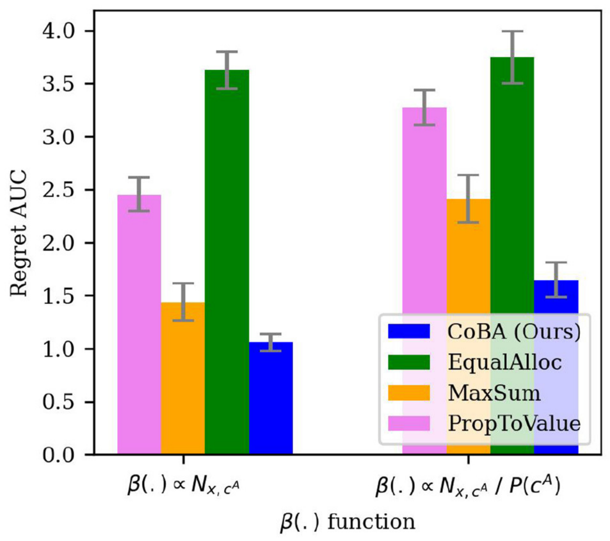

FIGURE 8
Results of the experiment studying robustness of Regret AUC of our algorithm to the choice of β (Experiment 6, Section 4.5.3). Though, as expected, the choice of cost function affects performance of all algorithms, our algorithm performs better than all algorithms for both cost function choices.

4.5.4 Experiment 7 (misspecification of $\mathcal{G}$)

In real-world applications, the true underlying causal graph may not always be known. In this experiment, we study the impact of mis-specification of $\mathcal{G}$ on the performance of our algorithm. Note that the formalism described in Section 2.1 does not necessitate that the agent knows the true underlying causal graph, but rather only that it knows a causal graph such that $\mathbb{P}$ factorizes according to it. This means that the graph $\mathcal{G}$ that the agent knows might include additional arrows not present in the true underlying graph. Intuitively, using such an imperfect graph would result in worsened performance by our algorithm since there are less overlapping information pathways to exploit.

In Experiment 7a, we study this effect empirically by comparing results of Experiment 2 to the same experiment but with the causal graph having an extra edge: $C_1 \rightarrow Y$. The results are averaged over 25 independent runs; error bars display ± 2 standard errors. Figure 9A shows the results of the experiment. As expected, performance of our algorithm degrades when there is imperfect knowledge of the true underlying graph. However, our algorithm continues to perform better than all baselines, while also maintaining a similar difference in regret AUC compared to the baselines. Figure 9A broken down by B is provided in the Supplementary material.

In Experiment 7b, we perform a similar analysis to the real-world inspired experiment presented in Section 4.4. Specifically, we compare the case where the true causal graph is known with the cases where there is a misspecification of the causal relationships between the context variables. To capture this, we add one edge ($C_1 \rightarrow C_0$) to $\mathcal{G}$ and then one more edge ($C_2 \rightarrow C_1$); we compare the performance under these two settings to the case where the true causal graph is known. The results are averaged over 25 independent runs; error bars display ± 2 standard errors. The results in Figure 9B shows the results. In this case, the deterioration in performance due to misspecification of $\mathcal{G}$ is quite small for all algorithms. However, our algorithm continues to perform better than all baselines; it performs the best when the true graph is known.

4.6 Regret behavior for large values of *B*

Figures 3, 4 provided regret behavior for small values of B. We are primarily interested in such small-budget behavior since that occurs more commonly in practice; for example, budgets exclusively for experimentation in software teams in often quite low.

However, it is also interesting to look at regret behavior as B becomes large. Specifically, we increase B large enough that all algorithms converge to optimal (or very close to optimal). We do this for Experiments 1 and 2. Figures 10A, B provide the results. Note that the Figures 3, 4 just zoom into these plots for small B (i.e., B between 15 and 30).

PropToValue is the slowest to converge to the optimal policy in both instances, though it demonstrates better low-budget behavior than EqualAlloc. MaxSum has the best low budget behavior among the baselines because it maximizes the total number of samples within that low budget; it, as B gets larger, EqualAlloc catches up (and even outperforms it) as it explores the context-action space better. In both experiments, however, our algorithm converges to an optimal policy faster than all baselines.

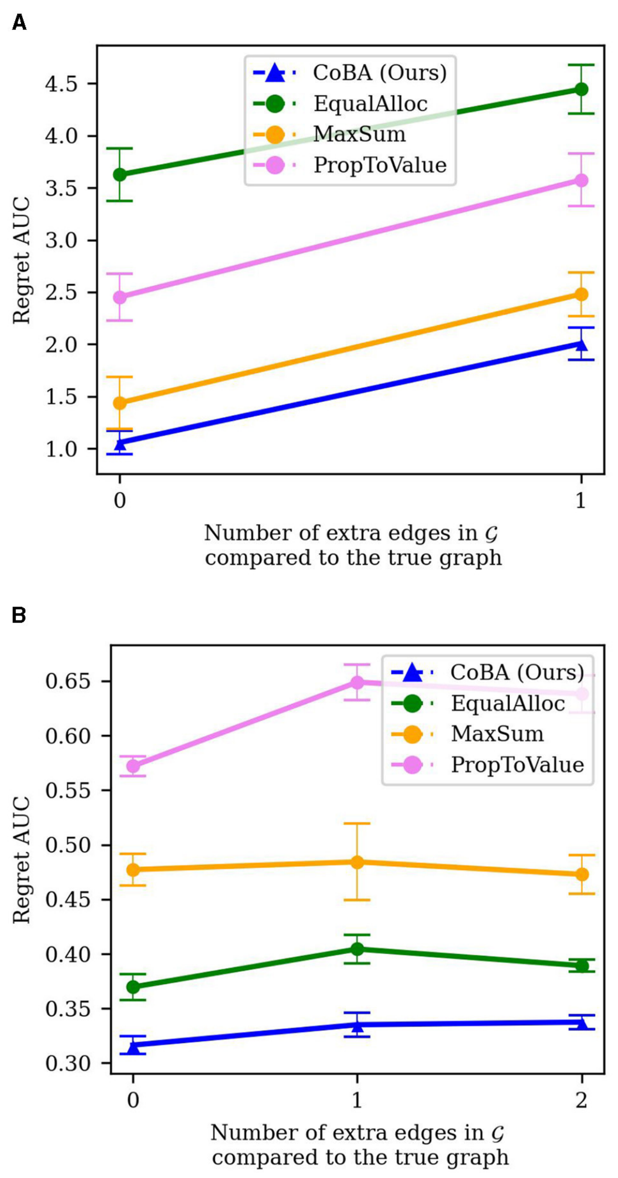

FIGURE 9
(A) Experiment 7a results (misspecification of $\mathcal{G}$ in Experiment 2). Though, as expected, imperfect knowledge of the true underlying graph results in reduced performance of our algorithm, it retains its relative better performance. **(B)** Experiment 7b results (misspecification of $\mathcal{G}$ in Experiment 3). All algorithms experience mild degradation in performance; our algorithm retains its relative performance compared to baselines.

4.7 Intuition for better performance of our algorithm

As discussed in Section 2.2, our algorithm balances the trade off between allocating more samples to context-action pairs that are higher value according to its beliefs and allocating more samples for exploration, while taking into account information leakage due to the causal graph. To understand this in more detail, we consider Setting 1 of Experiment 1, and zoom into the case where $B = 20$. We do 50 independent runs and plot the frequency of choosing

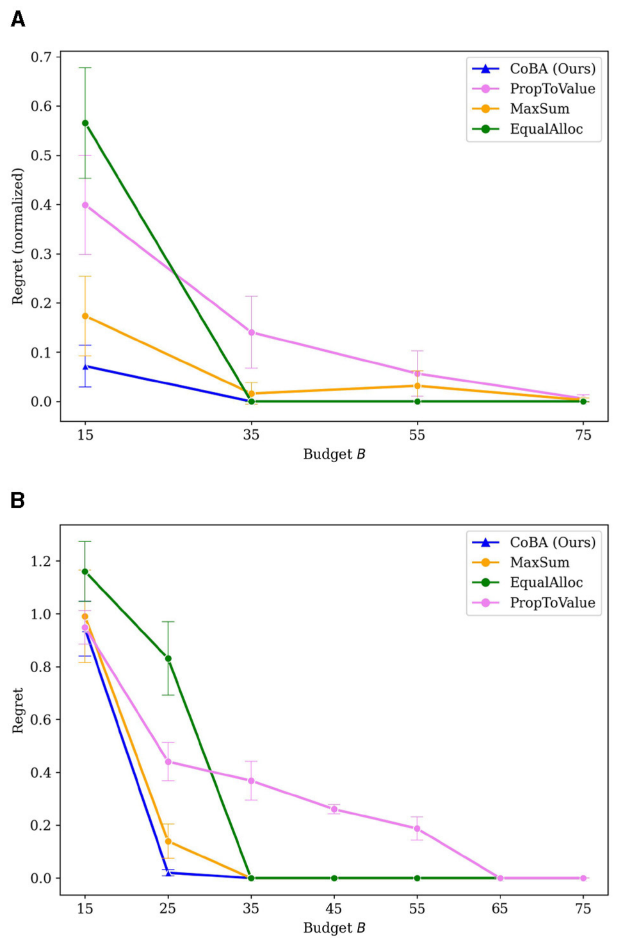

FIGURE 10
(A) Results of Experiment 1 (provided in Section 4.2) extended for large values of B. **(B)** Results of Experiment 2 (provided in Section 4.3) extended for large values of B.

samples containing different value of C_1. We show this for our algorithm and all baselines (except EqualAlloc since it is obvious how it allocates).

Figure 11 shows the results of this experiment. MaxSum allocates lesser number of samples than our algorithm to the two context values ($C_1 \in \{0, 1\}$) that are high value. PropToValue over-allocates to these two context values, resulting in poor exploration of other contexts. Our algorithm, in contrast, allocate relatively more to the high-value contexts, while also maintaining good exploration of other contexts.

5 Fairness results

Fairness is becoming an increasingly important angle to discuss when designing machine learning algorithms. A common way to approach fairness is to ensure some subset of variables (assumed given to the algorithm), called "sensitive variables," is not discriminated against. Specific formal definitions of this discrimination give rise to different notions of fairness in literature

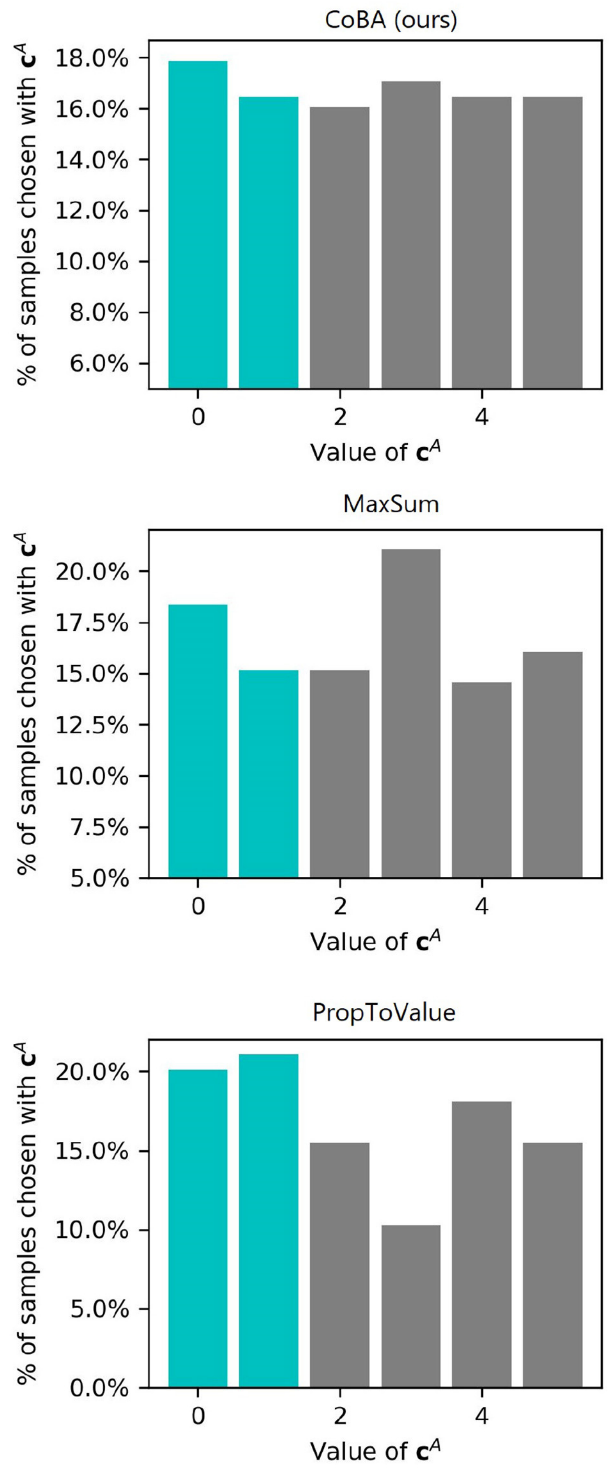

FIGURE 11
Frequency of choosing or encountering each value of $\mathcal{C}^A$. Highlighted in teal color are the "high-value" contexts (i.e., contexts for which learning the right actions provides higher expected rewards).

(Grgić-Hlača et al., 2016; Dwork et al., 2012; Kusner et al., 2017; Zuo et al., 2022; Castelnovo et al., 2022).

5.1 Counterfactual fairness

Counterfactual fairness is a commonly used notion of individual fairness. Intuitively, a *counterfactually fair* mapping from contexts to actions ensures that the actions mapped to an individual (given by a specific choice of values for the context variables) are the same in a counterfactual world where a subset $W \subseteq \mathcal{C}$ of sensitive contexts is changed.

For example, if a loan decision policy is counterfactually fair with respect to gender, then a a male applicant would have had the same probability of being granted the loan *had he been* a different gender. Counterfactual fairness requires the knowledge of the causal graph to be able to establish, making it difficult to use in practice; however, when causal graphs are available, it is a powerful notion of individual-level fairness.

In our case, counterfactual fairness can be achieved by setting $\mathcal{C}^B$ to contain all the sensitive attributes; that is, by letting $\mathcal{C}^B \supseteq W$.

5.1.1 Proof of counterfactual fairness

To prove counterfactual fairness, first note that the learned policy is a map $\hat{\phi} : \mathsf{val}(\mathcal{C}^A) \to \mathsf{val}(X)$; during inference, for any given $\mathbf{c}^A$, the value of X is intervened to be set to $\hat{\phi}(\mathbf{c}^A)$. Following the notation in Pearl (2009a), we let $\hat{\phi}_{\mathcal{C}^B \leftarrow \mathbf{c}^{B'}}(\mathbf{c}^A)$ denote $\hat{\phi}(\mathbf{c}^A)$ in the counterfactual world where the variables in $\mathcal{C}^B$ are set equal to $\mathbf{c}^{B'}$. To achieve counterfactual fairness [whose definition we draw from the definitions in Kusner et al. (2017) and Zuo et al. (2022)], it is sufficient that, for all $\mathbf{c}^A, \mathbf{c}^B, \mathbf{c}^{B'}, x$,

$$\mathbb{P}\left[\hat{\phi}_{\mathcal{C}^B \leftarrow \mathbf{c}^B}(\mathbf{c}^A) = x | \mathbf{c}^A, \mathbf{c}^B\right] = \mathbb{P}\left[\hat{\phi}_{\mathcal{C}^B \leftarrow \mathbf{c}^{B'}}(\mathbf{c}^A) = x | \mathbf{c}^A, \mathbf{c}^B\right]$$

Now, under the assumptions in Section 2.1.4, the conditional independences in $\mathcal{G}$ imply that we have

$$\mathbb{P}\left[\hat{\phi}(\mathbf{c}^A) = x | \mathbf{c}^A, \mathbf{c}^B\right] = \mathbb{P}\left[\hat{\phi}(\mathbf{c}^A) = x\right], \forall \mathbf{c}^B$$

This gives us that

$$\mathbb{P}\left[\hat{\phi}_{\mathcal{C}^B \leftarrow \mathbf{c}^{B'}}(\mathbf{c}^A) = x | \mathbf{c}^A, \mathbf{c}^B\right] = \mathbb{P}\left[\hat{\phi}(\mathbf{c}^A) = x\right], \forall \mathbf{c}^A, \mathbf{c}^B, \mathbf{c}^{B'}, x$$

which satisfies the counterfactual fairness condition.

5.2 Demographic parity

A common criterion for group-level fairness is Demographic Parity (Kusner et al., 2017). Demographic Parity (DP) requires that the distribution over actions remains the same irrespective of the value of the sensitive variables.

Demographic parity is a popular notion of fairness as it aligns well with most people's understanding of fairness and does not require strong assumptions such as the knowledge of the underlying causal graph to compute. As an example, if a loan decision policy has demographic parity with respect to gender, then the probability of granting a loan given a random male is the same as that for a random individual from any other gender group.

In our case, note that

$$\mathbb{P}\left[\hat{\phi} = x | \mathbf{c}^B\right] = \sum_{\mathbf{c}^A} \mathbb{P}[\mathbf{c}^A, \hat{\phi}(\mathbf{c}) = x | \mathbf{c}^B]$$

Formally, DP requires that

$$\mathbb{P}\left[\hat{\phi} = x|\mathbf{c}^B\right] = \mathbb{P}\left[\hat{\phi} = x|\mathbf{c}^{B\prime}\right],\ \forall \mathbf{c}^{B\prime}$$

Our algorithm, however, does *not* guarantee DP even if $\mathcal{C}^B \supseteq W$:

$$\mathbb{P}\left[\hat{\phi} = x|\mathbf{c}^B\right] = \sum_{\mathbf{c}^A} \mathbb{P}[\mathbf{c}^A|\mathbf{c}^B] \cdot \mathbb{P}[\hat{\phi}(\mathbf{c}^A) = x]$$

which may not equal $\mathbb{P}\left[\phi = x|\mathbf{c}^{B\prime}\right]$ since $\mathbb{P}[\mathbf{c}^A|\mathbf{c}^B]$ may not equal $\mathbb{P}[\mathbf{c}^A|\mathbf{c}^{B\prime}]$. And since $\mathcal{C}^B \supseteq W$, we cannot guarantee demographic parity.

However, we discuss one way through which DP can be achieved, but with a reduction in agent's performance. Specifically, we can achieve DP by ensuring that the agent acts according to a fixed policy irrespective of the value of $\mathbf{c}^A$. Intuitively, we construct a fixed policy that maximizes rewards given the agent's learned beliefs.

Specifically, let $\hat{\psi}(x, \mathbf{c}^A) \triangleq \hat{\mathbb{E}}[Y|do(x), \mathbf{c}^A]$. Assume the fixed policy is probabilistic. Therefore, we're interested in a policy q which is a distribution over $|\mathsf{val}(X)|$. Denoting $q^{(x)} \triangleq q(x)$, we solve the following optimization problem:

$$< ..., q^{(x)}, ... >=$$
$$argmax_{<...,q^{(x)\prime},...>} \sum_x \left[q^{(x)\prime} \left[\sum_{\mathbf{c}^A} \hat{\mathbb{P}}[\mathbf{c}^A] \cdot \hat{\psi}(x, \mathbf{c}^A) \right] \right]$$

subject to

$$\sum_x q^{(x)\prime} = 1$$

It is easy to see that one global optimum to this involves assigning a probability of 1 to an action x that results in the largest value of $\sum_{\mathbf{c}^A} \hat{\mathbb{P}}[\mathbf{c}^A] \cdot \hat{\psi}(x, \mathbf{c}^A)$. That is, choose x such that

$$x = argmax_{x'} \sum_{\mathbf{c}^A} \hat{\mathbb{P}}[\mathbf{c}^A] \cdot \hat{\psi}(x', \mathbf{c}^A)$$

and let $q(x) = 1$, and $q(x') = 0, \forall x' \neq x$. Note that this fixed policy would perform worser on expectation than the context-specific policy learned by the agent in the main part of the paper. This, however, is a cost that can be paid to achieve DP.

As an example of how this might be useful in practice, note that a recruitment agency might be more concerned (perhaps due to legal requirements) about ensuring that it does not discriminate against particular races, genders, etc., even if that leads to suboptimal allocations of jobs. The above approach ensures that such an entity can achieve demographic parity, even though it costs in terms of performance.

As a toy example, suppose there are just two contexts $\{0, 1\}$ and two actions $\{0, 1\}$ such that for context 0, the rewards for actions 0 and 1 are 10 and 0, respectively; and for context 1, the corresponding rewards are 0 and 9. Both contexts are equally likely. In this case, the above approach will choose action 0 always and achieve DP with respect to the context variable, but will receive 0 reward for context 1 which is far from optimal. We do not provide a full-fledged characterization of the performance-fairness tradeoffs and reserve that for future work.

6 Discussion and conclusion

Though exploitation of causal side information in multi-armed bandits has been relatively well-studied, its integration in contextual bandit settings remains much less investigated. This work presented only the second work in this area. Specifically, this paper proposed a new contextual bandit problem formalism where the agent, which has access to qualitative causal side information, can also actively obtain a table of experimental data in one shot, but at a cost and within a budget.

Further, most contextual bandit problems have been studied in the case when contexts are received from the environment. However, there are several real world settings such as marketing campaigns where targeted experiments are possible. This work is one of the very few works to study this setting.

We proposed a novel algorithm based on a new measure similar to entropy, and showed extensive empirical analysis of our algorithm's performance. We also showed theoretical results on soundness and regret. As demonstrated, smartly exploiting information leakage from the causal graph can yield significantly improved performance. Further, it is also possible to achieve certain notions of individual and group fairness, though it might come at a cost of reduced performance.

This work opens up various directions of future research. Fairness of machine learning models is one of the fast-growing areas of research due to the increased interest in responsible development of AI; it is worthwhile to design causal contextual bandit algorithms that meet population-level fairness criteria with minimal impact on performance. It is also useful to investigate more general approaches to fairness such as optimizing under general fairness constraints. Another useful direction is to allow the presence of unobserved confounders in the causal model; while the lack of unmeasured confounders is frequently assumed in causal bandits literature, removing that assumption can provide a qualitative improvement to the setting and make it more widely applicable. It is also interesting to investigate infinite action or context spaces (for example, domains that are continuous) as they often show up in real world scenarios. Another direction is to study contextual bandit algorithms that combine causal discovery with regret optimization can further expand the scope of application by entirely removing the need for a causal graph to be provided. Finally, it might be fruitful to investigate ways to combine one-shot algorithms with sequential algorithms to provide a more comprehensive approach, and to also look at evolving causal graphs.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found at: Zenodo, https://zenodo.org/, doi: 10.5281/zenodo.5540348.

Author contributions

CS: Writing – original draft, Conceptualization, Data curation, Investigation, Methodology, Software, Writing – review & editing.

BR: Writing – review & editing, Conceptualization, Funding acquisition, Methodology, Supervision, Validation.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Acknowledgments

This paper is based on two chapters from the thesis (Subramanian, 2024).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

The author(s) declared that they were an editorial board member of Frontiers, at the time of submission. This had no impact on the peer review process and the final decision.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frai.2024.1346700/full#supplementary-material

References

Agrawal, S., and Goyal, N. (2012). "Analysis of Thompson sampling for the multi-armed Bandit problem," in *Proceedings of the 25th Annual Conference on Learning Theory, Vol. 23 of PMLR* (Edinburgh), 39.1–39.26.

Ameko, M. K., Beltzer, M. L., Cai, L., Boukhechba, M., Teachman, B. A., and Barnes, L. E. (2020). "Offline contextual multi-armed bandits for mobile health interventions: a case study on emotion regulation," in *Proceedings of the 14th ACM Conference on Recommender Systems* (New York, NY), 249–258.

Bouneffouf, D., Rish, I., and Aggarwal, C. (2020). "Survey on applications of multi-armed and contextual bandits," in *2020 IEEE Congress on Evolutionary Computation (CEC)* (Glasgow), 1–8.

Castelnovo, A., Crupi, R., Greco, G., Regoli, D., Penco, I. G., and Cosentini, A. C. (2022). A clarification of the nuances in the fairness metrics landscape. *Sci. Rep.* 12:4209. doi: 10.1038/s41598-022-07939-1

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., et al. (2021). Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learn.* 110, 2419–2468. doi: 10.1007/s10994-021-05961-4

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. (2012). "Fairness through awareness," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (New York, NY), 214–226.

Grgić-Hlača, N., Zafar, M. B., Gummadi, K. P., and Weller, A. (2016). "The case for process fairness in learning: feature selection for fair decision making," in *Symposium on Machine Learning and the Law at the 29th Conference on Neural Information Processing Systems*. Barcelona.

Guo, R., Cheng, L., Li, J., Hahn, P. R., and Liu, H. (2020). A survey of learning causality with data: problems and methods. *ACM Comput. Surv.* 53, 1–37. doi: 10.1145/3397269

Han, Y., Zhou, Z., Zhou, Z., Blanchet, J., Glynn, P. W., and Ye, Y. (2020). Sequential batch learning in finite-action linear contextual bandits. *arXiv [preprint]*. doi: 10.48550/arXiv.2004.06321

Joachims, T., Swaminathan, A., and Rijke, M. d. R. (2018). "Deep learning with logged bandit feedback," in *Proceedings of the Sixth International Conference on Learning Representations*. Vancouver, BC.

Koller, D., and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: The MIT Press.

Kusner, M. J., Loftus, J., Russell, C., and Silva, R. (2017). "Counterfactual fairness," in *Advances in Neural Information Processing Systems, Vol. 30*, eds. I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (New York, NY: CA), 4069–4079.

Lattimore, F., Lattimore, T., and Reid, M. D. (2016). "Causal bandits: learning good interventions via causal inference," in *Advances in Neural Information Processing Systems 29, Vol. 29*, eds. D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Barcelona), 1189–1197.

Lattimore, T., and Szepesvári, C. (2020). *Bandit Algorithms*. Cambridge: Cambridge University Press.

Liu, B., Wei, Y., Zhang, Y., Yan, Z., and Yang, Q. (2018). "Transferable contextual bandit for cross-domain recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32* (New York, NY: LA), 3619–3626.

Lu, Y., Meisami, A., Tewari, A., and Yan, W. (2020). "Regret analysis of bandit problems with causal background knowledge," in *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), Vol. 124 of PMLR* (New York, NY: Virtual Event), 141–150.

Pearl, J. (2009a). Causal inference in statistics: an overview. *Stat. Surv.* 3, 96–146. doi: 10.1214/09-SS057

Pearl, J. (2009b). *Causality, 2nd Edn*. Cambridge: Cambridge University Press.

Pearl, J. (2019). On the Interpretation of do(x). *J. Causal Infer.* 7:2002. doi: 10.1515/jci-2019-2002

Ren, Z., Zhou, Z., and Kalagnanam, J. R. (2022). Batched learning in generalized linear contextual bandits with general decision sets. *IEEE Contr. Syst. Lett.* 6, 37–42. doi: 10.1109/LCSYS.2020.3047601

Russo, D., Roy, B. V., Kazerouni, A., and Osband, I. (2017). A tutorial on thompson sampling. *arXiv [preprint]*. doi: 10.48550/arXiv.1707.02038

Sawant, N., Namballa, C. B., Sadagopan, N., and Nassif, H. (2018). Contextual multi-armed bandits for causal marketing. *arXiv [preprint]*. doi: 10.48550/arXiv.1810.01859

Sen, R., Shanmugam, K., Dimakis, A. G., and Shakkottai, S. (2017). "Identifying best interventions through online importance sampling," in *Proceedings of the 34th International Conference on Machine Learning, Vol. 70 of PMLR* (New York, NY), 3057–3066.

Settles, B. (2012). *Active Learning, 1st Edn*. Cham: Springer.

Storn, R., and Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optimizat.* 11, 341–359. doi: 10.1023/A:1008202821328

Subramanian, C. (2024). *Causal Contextual Bandits* (Ph. D. thesis). Indian Institute of Technology Madras, Chennai, India.

Subramanian, C., and Ravindran, B. (2022). "Causal contextual bandits with targeted interventions," in *Proceedings of the Tenth International Conference on Learning Representations (ICLR 2022)*. Appleton, WI: Virtual Event.

Swaminathan, A., and Joachims, T. (2015a). Batch learning from logged bandit feedback through counterfactual risk minimization. *J. Machine Learn. Res.* 16, 1731–1755. doi: 10.5555/2789272.2886805

Swaminathan, A., and Joachims, T. (2015b). "Counterfactual risk minimization: learning from logged bandit feedback," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning, Vol. 37* (Lille), 814–823.

Wu, H., Srikant, R., Liu, X., and Jiang, C. (2015). "Algorithms with logarithmic or sublinear regret for constrained contextual bandits," in *Advances in Neural Information Processing Systems 28* (Montreal, QC), 433–441.

Yabe, A., Hatano, D., Sumita, H., Ito, S., Kakimura, N., Fukunaga, T., et al. (2018). "Causal bandits with propagating inference," in *Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of PMLR* (Stockholm), 5512–5520.

Zhang, J., and Bareinboim, E. (2017). "Transfer learning in multi-armed bandits: a causal approach," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence* (Melbourne, VIC), 1340–1346.

Zhang, Z., Ji, X., and Zhou, Y. (2022). Almost optimal batch-regret tradeoff for batch linear contextual bandits. *arXiv [preprint]*. doi: 10.48550/arXiv.2110.08057

Zuo, A., Wei, S., Liu, T., Han, B., Zhang, K., and Gong, M. (2022). "Counterfactual fairness with partially known causal graph," in *Advances in Neural Information Processing Systems, Vol. 35*, eds. S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (New York, MY: LA), 1238–1252.