Toward Optimal Regret in Robust Pricing: Decoupling Corruption and Time

Kalana Kalupahana[*†]
Politecnico di Milano

Francesco Emanuele Stradi[*†]
Politecnico di Milano

Matteo Castiglioni[†]
Politecnico di Milano

Alberto Marchesi[†]
Politecnico di Milano

Abstract

We design the first regret guarantees for robust dynamic pricing that decouple the dependence on the corruption C and the time horizon T. In dynamic pricing, a seller with unlimited supply of a good interacts with a stream of buyers over T rounds, with the goal of maximizing revenue. At each round t, the seller posts a price p_t, and the buyer purchases the good only if their unknown valuation $v^\star$ exceeds this price. The seller observes only the binary feedback $\mathbb{1}\{p_t \leq v^\star\}$, indicating whether a sale occurred. In the *robust* pricing setting, a malicious adversary is allowed to corrupt this feedback in at most C rounds. Even if the learner knows the corruption C, the best known regret bound is $\mathcal{O}(C \log\log T)$ by Gupta et al. [2025]. This leaves as an open problem to "decouple" the dependence on C and T. In this work, we resolve this open problem. In particular, we develop a robust variant of binary search that achieves regret $\mathcal{O}(C + \log T)$ when the corruption C is known and $\mathcal{O}(C + \log^2 T)$ when the corruption is unknown.

1 Introduction

Dynamic pricing is a central problem in sequential decision-making under uncertainty, first studied by Kleinberg and Leighton [2003]. In its simplest form, a seller (the learner) with unlimited supply of a good interacts with a stream of buyers over T rounds. At each round t, the seller posts a price $p_t \in [0, 1]$, and the buyer purchases the good only if their *unknown* valuation $v^\star \in [0, 1]$ exceeds this price, *i.e.*, when $p_t \leq v^\star$. Thus, the seller's revenue at round t is given by:

$$r_t = p_t \cdot \mathbb{1}\{p_t \leq v^\star\}.$$

The goal of the seller is to maximize the total revenue over T rounds, or equivalently, to minimize the *regret* R_T with respect to always posting the optimal fixed price in hindsight, namely $v^\star$. Formally:

$$R_T = T \cdot v^\star - \textstyle\sum_{t=1}^{T} r_t.$$

One of the main challenges in dynamic pricing is that, at the end of each round t, the seller observes only *binary feedback*, indicating whether a purchase occurred, *i.e.*, the value of $\mathbb{1}\{p_t \leq v^\star\}$. In their seminal work, Kleinberg and Leighton [2003] show that such problems can be addressed very efficiently by designing a clever binary-search-style algorithm that achieves an optimal regret rate of $\Theta(\log\log T)$. Since then, the dynamic pricing problem and several of its variants have been extensively studied (see Appendix A for additional discussion on related work).

[*]Equal contribution.
[†]name.surname@polimi.it

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In this paper, we study the *robust* version of the dynamic pricing problem described above—originally introduced by Krishnamurthy et al. [2023]—in which the feedback received by the seller is subject to *adversarial corruptions*. In particular, in at most C rounds, the feedback observed by the seller may be arbitrary. The goal is thus to derive regret guarantees that degrade gracefully as a function of C.

Adversarial corruptions arise naturally in many real-world scenarios, where the observed sale/no-sale signal may fail to reflect the true comparison between the posted price and the buyer's valuation due to noise, strategic behavior, or manipulation. Remarkably, dealing with corruption is especially delicate in pricing. Since each round reveals only one bit of information, even a small number of corrupted observations may mislead the learning process. Put differently, the same structure that makes the uncorrupted problem highly learnable also renders it intrinsically fragile: once some of the comparisons are unreliable, naively extending binary search ideas is no longer sufficient.

The state-of-the-art approach for this setting is the recent algorithm of Gupta et al. [2025], which achieves regret $\mathcal{O}(C \log\log T)$—their result also extends to the d-dimensional contextual setting; see Appendix A for a more detailed discussion. Interestingly, they also show that regret guarantees of order $\mathcal{O}(C + \log\log T)$ are unattainable, implying that obtaining a linear dependence on C requires a departure from the classical $\mathcal{O}(\log\log T)$ rate typical of uncorrupted dynamic pricing. Moreover, they achieve this linear dependence on C in a restricted setting with *cautious buyers*, in which corruptions are *one-sided*: a buyer may refuse to purchase even when the posted price is below their valuation, but can never purchase when the price exceeds it. In this model and assuming to know the corruption C, they obtain a regret guarantee of order $\mathcal{O}(C + \log T)$. They leave open whether the same regret bound can be achieved under arbitrary but known corruption. We go beyond that and answer in the affirmative the following question:

Can regret $\mathcal{O}(C) + \text{poly}(\log T)$ be achieved in dynamic pricing under arbitrary and unknown adversarial corruption?

1.1 Our Results and Techniques

Motivated by the problem left open by Gupta et al. [2025], we focus on regret guarantees of order $\mathcal{O}(C) + \text{poly}(\log T)$. This rate is provably optimal in the large-corruption regime $C \geq \text{poly}(\log T)$, as suggested by the following simple observation.

Observation 1.1. *Consider the robust dynamic pricing problem with feedback corrupted in at most C rounds. Then, any deterministic algorithm suffers regret $R_T \geq \Omega(C)$, even if C is known.*

The intuition is the following. Consider two instances with well-separated valuations, say $v_L^\star = 1/3$ and $v_H^\star = 2/3$. For the first C rounds, the adversary can use its corruption budget to make the feedback observed under the second instance identical to the uncorrupted feedback that would have been observed under the first one. Since the algorithm is deterministic, the same prices are posted in both instances during these rounds. However, for any posted price p_t, at least one of the two instances incurs constant regret. Hence, the sum of the regrets over the two instances is $\Omega(C)$, which implies that one of them has regret $\Omega(C)$.

In this work, we provide regret guarantees for both known and unknown corruption C. At the core of both results lies a unified meta-algorithm, which is inspired by a classical binary search and then robustified through two additional components: *safety checks* and *backtracking*. This approach is primarily motivated by the impossibility result of Gupta et al. [2025], which shows that the $\mathcal{O}(\log\log T)$ rate achieved by clever approaches in the uncorrupted setting (see, *e.g.*, [Kleinberg and Leighton, 2003]) is unattainable in the presence of adversarial corruption. This suggests that a classical binary-search-style approach—yielding a $\log T$[3] dependence—is the appropriate algorithmic approach for pricing under corruption.

The algorithm begins with a binary search phase designed to shrink the candidate interval. This phase is naturally agnostic to the corruption level C and is shared across both settings. We employ a potential function argument to prove that the number of "wrong steps"—in which the algorithm narrows the search to an interval not containing $v^\star$—is bounded linearly in the corruption C.

Once the algorithm reaches a leaf interval of size $1/T$, it enters a "commitment phase" in which it tries to determine whether it is committing to the right interval. If it is the case, the algorithm can

[3] In this work, we will refer as $\log T$ to $\log_2 T$.

simply post the lower extreme, incurring a negligible per-round regret of at most $1/T$. This step is tailored to the known or unknown corruption setting.

In the known corruption setting, the algorithm keeps posting the left and right extremes of the interval. If the feedback remains consistent with $v^\star$ belonging to the interval for $C+1$ times, we can safely assume that $v^\star$ belongs to the interval and thus starts committing to the left extreme. Any inconsistent feedback triggers a backtracking procedure, returning the search to the parent node, and restarting the binary search procedure from the parent node. We upperbound the number of rounds required to commit to the optimal interval and show that this strategy yields a regret bound of $\mathcal{O}(C+\log T)$, resolving the open problem posed by Gupta et al. [2025].

In the unknown-corruption setting, designing the commitment phase is more challenging. The algorithm must balance a fundamental tension: to quickly leave an incorrect leaf, it should query the right endpoint, but if the leaf is correct, these queries only lead to unnecessary regret. To address this, we design a randomized commitment strategy in which the algorithm queries the right endpoint with a probability that decreases over time. The intuition is that, as a leaf survives more safety checks, either it is increasingly plausible that the leaf is correct, or a large number of corruptions have been used to make it appear correct. In both cases, the algorithm can reduce exploration: in the first case, because committing is likely safe, and in the second, because the extra regret can be charged to the corruption budget. By carefully scheduling the decaying probability, we balance the regret from staying in a wrong leaf against the regret from exploring in a correct one, obtaining a total regret bound of $\mathcal{O}(C+\log^2 T)$.

1.2 Relation to Robust Binary Search

Robust pricing is deeply connected with binary search problems under corruption [Rivest et al., 1980, Dagan et al., 2021]. Specifically, in standard, uncorrupted dynamic pricing, binary search is not optimal: the seminal result of Kleinberg and Leighton [2003] shows that the threshold structure of the problem can be exploited much more efficiently, leading to the optimal $\Theta(\log\log T)$ regret rate. However, as discussed above, this doubly logarithmic dependence is no longer compatible with a linear dependence on the corruption budget C under adversarial corruptions. Once the goal becomes to obtain guarantees of the form $\mathcal{O}(C)+\mathrm{poly}(\log T)$, the problem naturally reduces to identify an approximately optimal solution as quickly as possible, upper-bounding the total regret by the length of the learning phase. Indeed, suppose that the learner can identify an interval of length at most $1/T$ containing $v^\star$. Then, by repeatedly posting the left endpoint of this interval, the learner guarantees a sale and suffers at most $1/T$ per-round regret. Identifying such small interval is the objective of robust binary search: each price query can be viewed as a comparison with $v^\star$, and a corrupted observation corresponds to a comparison answer pointing to the wrong side.

This perspective connects our setting to the seminal work of Rivest et al. [1980], who study binary search with erroneous answers. In their model, at most C comparison answers may be corrupted, and the value of C is known to the algorithm. Their main result gives a worst-case query complexity of $\log T + C\log\log T + \mathcal{O}(C\log C)$. Although this is the form in which their result is presented, a more refined analysis of their actual upper bound (which we provide in Appendix B) can lead to a worst-case query complexity of $\mathcal{O}(C+\log T)$. Combined with the commit-to-the-left-endpoint argument above, this yields an alternative proof of the $\mathcal{O}(C+\log T)$ regret guarantee for robust pricing in the known-corruption case.

The limitation of this approach is that it crucially relies on knowing the corruption budget in advance, and it cannot be easily extended to the case C is *not* known. This motivates the approach developed in the present paper. We design a robust search procedure that preserves the same high-level goal—finding a $1/T$-accurate interval and then exploiting its left endpoint—but is built such that it can be extended to the unknown-corruption setting.

2 Preliminaries

We consider a *dynamic pricing problem* over a horizon of T rounds, denoted by $[T]=\{1,\dots,T\}$, during which a seller with unlimited supply of a good interacts with a stream of buyers who share

a common but unknown valuation $v^\star \in [0, 1)$ for one unit of the good.[4] At each round $t \in [T]$, the seller posts a price $p_t \in [0, 1]$ based on past observations. The buyer then decides whether to purchase the item according to a standard threshold rule: a sale occurs if and only if the posted price does not exceed the valuation. Accordingly, the seller observes binary feedback:

$$\sigma_t := \mathbb{1}\{p_t \leq v^\star\},$$

where $\sigma_t = 1$ indicates a successful sale and $\sigma_t = 0$ indicates no sale. The revenue collected by the seller at each round $t \in [T]$ is therefore $r_t := p_t \cdot \sigma_t$, and the seller's goal is to maximize the cumulative revenue over the entire time horizon.

The seller's performance is evaluated in terms of their *(cumulative) regret*, which measures how much revenue the seller loses compared to a clairvoyant benchmark that knows the true valuation $v^\star$ and posts it at every round. Since posting $v^\star$ always results in a sale and yields revenue exactly $v^\star$ per round, the cumulative benchmark revenue is $T \cdot v^\star$. Thus, the seller's regret is defined as:

$$R_T := T \cdot v^\star - \sum_{t=1}^{T} r_t.$$

2.1 Robust Dynamic Pricing

In this paper, we study a *robust* extension of the dynamic pricing model introduced above, in which the feedback observed by the seller may be corrupted by an adversary. Following the standard formulation for adversarial corruptions in pricing [Krishnamurthy et al., 2023, Gupta et al., 2025], we assume that the total number of corrupted rounds is bounded by a budget C. We consider two settings of increasing difficulty: the *known-corruption setting*, in which the seller knows the budget C, and the *unknown-corruption setting*, in which C is not known to the seller.

To distinguish between the truthful feedback generated according to $v^\star$ and the feedback actually observed by the seller, we let:

$$y_t := \mathbb{1}\{p_t \leq v^\star\}$$

denote the uncorrupted sale indicator, while $\sigma_t \in \{0, 1\}$ denotes the feedback actually observed by the seller. In the robust setting, σ_t may differ from y_t subject to the following budget constraint:

$$|\{t \in [T] : \sigma_t \neq y_t\}| \leq C.$$

On corrupted rounds, a truthful sale ($y_t = 1$) may be reported as a no-sale ($\sigma_t = 0$), and a truthful no-sale ($y_t = 0$) may be reported as a sale ($\sigma_t = 1$). On all uncorrupted rounds, the feedback is consistent with the threshold rule induced by $v^\star$. As is standard in the literature [Krishnamurthy et al., 2023], we assume that the adversary can decide whether to corrupt a round t based on the history and the seller's distribution over prices at time t, but not on the realized price.[5]

Our goal is to design dynamic pricing strategies for the seller that achieve regret guarantees depending on the corruption budget C.

3 Meta Algorithm

In this section, we present a meta-algorithm for dynamic pricing under adversarial corruptions. In subsequent sections, we instantiate it with different components depending on whether the corruption budget is known or unknown to the seller.

Our approach is motivated by the impossibility result of Gupta et al. [2025], which shows that the $\mathcal{O}(\log \log T)$ regret achieved by clever approaches in the uncorrupted setting (see [Kleinberg and Leighton, 2003]) is unattainable in the presence of adversarial corruptions. Consequently, we build our dynamic pricing strategy upon a classic binary search, "robustified" through two additional components: *(i) safety checks* and *(ii) backtracking*.

[4] We assume that $v^\star \in [0, 1)$ to avoid boundary cases in our analysis. All our results can be readily extended to the case $v^\star \in [0, 1]$ using standard techniques.

[5] We allow the adversary to decide *how* to corrupt the feedback depending on the actual price selected by the seller.

In Algorithm 1, we provide the pseudocode of our meta-algorithm. The algorithm begins with the unit interval $I_0 := [0, 1]$ and iteratively shrinks it to a "leaf" interval I_D of length at most $1/T$, which is reached at depth $D := \log T$.[6] The algorithm operates under the (possibly incorrect) assumption that the true valuation $v^\star$ lies within the current interval. It proceeds by shrinking the interval according to a standard binary search, augmented with additional safety checks. If the feedback becomes inconsistent with past observations, the algorithm triggers a backtracking procedure.

Given the current interval $I = [L, R)$, the algorithm first performs a safety check by querying the endpoints L and R. Let σ_L and σ_R denote the observed feedback. If $\sigma_L = 0$ (no sale at the lower extreme) or $\sigma_R = 1$ (sale at the upper extreme), the observations are inconsistent with the hypothesis $v^\star \in [L, R]$.[7] In this case, the algorithm backtracks to the previous iteration of the binary search, *i.e.*, the "parent" interval. If the safety check passes, the algorithm queries the midpoint $M := \frac{L+R}{2}$. Based on the feedback σ_M obtained by posting M, it updates the interval according to the standard binary search rule.

The process described above continues until the algorithm reaches a leaf interval in $\mathcal{L} := \{\ell_1, \ldots, \ell_{2^D}\}$ at depth D. Notice that all the leaf intervals have length at most $1/T$. We denote the unique leaf interval containing $v^\star$ as $\ell^\star$.

Algorithm 1 Robust Dynamic Pricing Meta-Algorithm

$I \leftarrow I_0 = [0, 1]$
$i \leftarrow 0$
while TRUE **do**
  $I \leftarrow I_i = [L, R)$
  if $I \notin \mathcal{L}$ **then**
    if SAFETYCHECK(I) = FAIL **then**
      $i \leftarrow i - 1$
    else
      Post $M = \frac{L+R}{2}$ and observe σ_M
      $i \leftarrow i + 1$
      if $\sigma_M = 1$ **then**
        $I_i \leftarrow [M, R]$
      else
        $I_i \leftarrow [L, M]$
      end if
    end if
  else
    if COMMIT(I)= FAIL **then**
      $i \leftarrow i - 1$
    end if
  end if
end while

Algorithm 2 SAFETYCHECK

Require: Current interval $I = [L, R)$
Post L and observe σ_L
Post R and observe σ_R
if $\sigma_L = 0$ or $\sigma_R = 1$ **then**
  return FAIL
else
  return PASS
end if

Once a leaf interval is reached, the interval is sufficiently small that no further shrinking is required. The algorithm therefore enters a *commitment phase*. For now, we consider an arbitrary committing strategy; its precise implementation depends on the specific setting—namely, whether the corruption budget C is known or unknown. At each round, this procedure may either continue committing or return a FAIL signal, in which case the algorithm backtracks to the parent interval. Crucially, aside from the commitment phase, our algorithm does not require knowledge of C. As we will see in the following sections, this knowledge can instead be leveraged to design simpler and more effective commitment strategies.

In Figure 1, we provide a graphical representation of the "robustified" binary search performed by our algorithm.

3.1 Regret Decomposition and Potential Argument

In this section, we provide a regret decomposition for Algorithm 1. Then, by combining this decomposition with the analysis of the commitment phases in Section 4 for known C, and in Section 5 for unknown C, we obtain our final regret bounds.

[6] In principle, it should be $D := \lceil \log T \rceil$. Throughout the paper, for ease of presentation, we often omit the $\lceil \cdot \rceil$ operator.

[7] To handle boundary cases in which corruption is trivially evident—specifically, when $L = 0$ and $\sigma_L = 0$, or $R = 1$ and $\sigma_R = 1$—we assume that the checks at $L = 0$ and $R = 1$ pass by default.

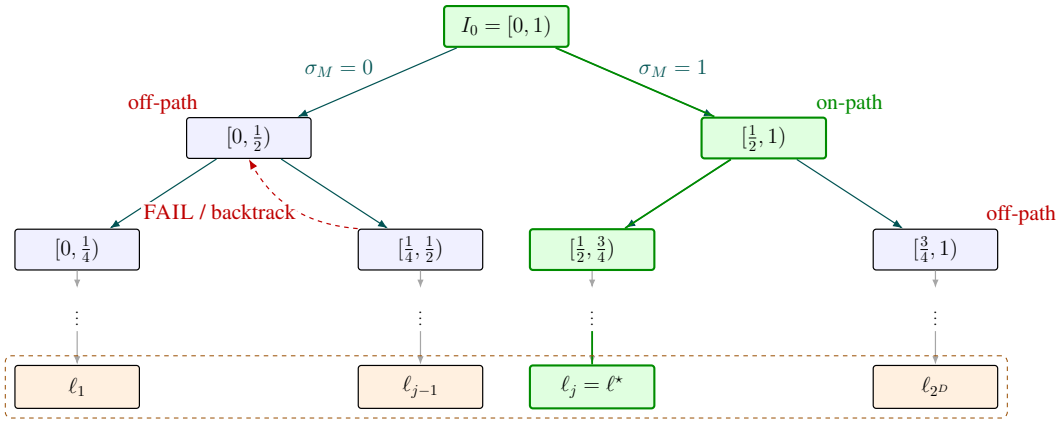

Figure 1: Tree of intervals of our robust dynamic pricing algorithm. The intervals induced by repeated halving $I_0 = [0, 1)$ can be represented as a binary tree of depth $D = \lceil \log_2 T \rceil$. The green nodes are the on-path intervals leading to the correct leaf $\ell^\star$, while the remaining intervals are off-path. The intervals at depth D are the leaves ℓ_j of the tree.

To analyze the cumulative regret, we partition the horizon T into *search steps*. Each search step corresponds either to *(i)* a single iteration of the binary search, consisting of a safety check and a possible midpoint query, or to *(ii)* a sequence of consecutive rounds in the commitment phase.

We introduce some additional notation. For any $\ell \in \mathcal{L}$, let $Q(\ell)$ denote the set of rounds in which the algorithm runs the commitment subroutine on ℓ. The regret incurred over these rounds is denoted by:

$$R(\ell) := \sum_{t \in Q(\ell)} (v^\star - p_t \mathbb{1}\{p_t \le v^\star\}) .$$

Finally, let N_{T} and N_{F} be the number of times the algorithm fails the commitment check on the correct leaf and on an incorrect leaf, respectively. In this section, we aim to show that the total regret can be bounded as:

$$R_T \le \mathcal{O}(C + \log T + N_{\mathrm{T}}) + \sum_{\ell \in \mathcal{L}} R(\ell).$$

We employ a potential argument to track the progress of the search. For any interval I in the binary tree, let $\mathrm{dist}(I, \ell^\star)$ denote the length of the shortest path between I and $\ell^\star$. At search step q, let I^q be the current interval, and define the potential:

$$\Phi_q := \mathrm{dist}(I^q, \ell^\star).$$

Initially, the algorithm starts at the root I_0, whose distance from any leaf is exactly D; hence, $\Phi_0 = D$. We say that an interval is *on-path* if it lies on the shortest path from the root to $\ell^\star$, and *off-path* otherwise. Moreover, we call a search step *honest* if all feedback observed during the step is uncorrupted, and *corrupted* if at least one round in the step is corrupted.

Now, we analyze how different search steps affect the potential Φ_q. In the following two lemmas, we start by considering non-leaf intervals.

Lemma 3.1. *Every honest search step q at a non-leaf interval $I^q \notin \mathcal{L}$ decreases the potential by exactly one, i.e., $\Phi_{q+1} = \Phi_q - 1$.*

Lemma 3.2. *Every corrupted search step q at a non-leaf interval $I^q \notin \mathcal{L}$ increases the potential by at most one.*

Finally, we consider how commitment search steps affect the potential.

Lemma 3.3. *Every commit search step q at a leaf interval $\ell^q \in \mathcal{L}$:*

- *Reduces the potential by 1 if $\ell^q \neq \ell^\star$*
- *Increases the potential by 1 if $\ell^q = \ell^\star$.*

Given the previous results, we are now able to provide the promised regret decomposition for our meta-algorithm. This is done in the following theorem.

Theorem 3.4. *Assume that the feedback is corrupted in at most C rounds. Then, Algorithm 1 limits the number of commitment steps on incorrect leaves to $N_{\mathrm{F}} \le \log T + C + N_{\mathrm{T}}$. Furthermore, its regret is at most:*

$$R_T \le \sum_{\ell \in \mathcal{L}} R(\ell) + 3\log T + 6C + 3N_{\mathrm{T}}.$$

Intuitively, Theorem 3.4 shows that the regret of the meta-algorithm is essentially the regret of the commitment subroutines, plus a small overhead due to searching. The key point is that failed commitments on incorrect leaves cannot occur too often: each such failure decreases the distance to the correct leaf, while this distance can increase only because of corrupted search steps or failed commitments on the correct leaf. Since the initial distance is only $D = \log T$, this yields an overhead of order $\log T + C + N_{\mathrm{T}}$. Hence, after this decomposition, it remains to control the leaf-level regret terms $R(\ell)$ and the number N_{T} of failed commitment checks on the correct leaf.

4 Warm-Up: The Known Corruption Case

In this section, we design a commitment strategy for the setting where the learner knows the corruption budget C. The procedure CommitKnown (see Algorithm 3 for the pseudocode) employs a leaf-specific counter s_ℓ for each leaf ℓ, which tracks how many times the interval has undergone a safety check at its extremes. We can think of the variables s_ℓ as global counters: they are initialized to 0 at the beginning of the horizon and are shared across all calls to the commitment procedure.

If the counter s_ℓ has not yet reached the threshold $C+1$, the algorithm performs endpoint queries as in the SafetyCheck procedure. If this check returns FAIL, the commitment subroutine terminates and signals the meta-algorithm to backtrack to the parent node. Otherwise, the counter s_ℓ is incremented, and the algorithm repeats the safety check.

Algorithm 3 CommitKnown

Require: Corruption C, current leaf interval $\ell = [L, R) \in \mathcal{L}$, counters s_ℓ for each $\ell \in \mathcal{L}$

```
while TRUE do
    if s_ℓ ≤ C then
        Post L and observe σ_L
        Post R and observe σ_R
        if σ_L = 0 or σ_R = 1 then
            return FAIL
        else
            s_ℓ ← s_ℓ + 1
        end if
    else
        Post L
    end if
end while
```

Once the counter s_ℓ reaches the threshold $C+1$, the algorithm enters the "true commitment" phase, posting the price L in every round. The intuition is straightforward: for any incorrect leaf ℓ, an honest safety check would return FAIL, and therefore the adversary can prevent this from happening only by corrupting the endpoint feedback. Since this can occur at most C times, a leaf that passes $C+1$ safety checks must be the correct leaf $\ell^\star$. At this point, the algorithm can safely stop performing safety checks and repeatedly post L, which guarantees per-round regret at most $1/T$ since $v^\star - 1/T < L < v^\star$.

To prove the regret upper bound, we start from the regret decomposition result in Theorem 3.4, which partitions the regret into three components: *(i)* an $\tilde{\mathcal{O}}(\log T + C)$ regret term that is independent of the specific commitment procedure, *(ii)* the number of backtracks on the correct leaf N_{T}, and *(iii)* the regret on leaf intervals $\sum_{\ell \in \mathcal{L}} R(\ell)$.

Bounding N_{T} is straightforward. Indeed, the algorithm backtracks from a commit step on N_{T} only if the check on one of the two extremes is corrupted. Hence, the following observation holds.

Observation 4.1. *Algorithm 1 with* Commit *procedure* CommitKnown *guarantees:*

$$N_{\mathrm{T}} \le C.$$

Next, we bound the regret incurred while the algorithm is in the commitment phase, regardless of whether it is on the correct leaf or on an incorrect one. Recall that Theorem 3.4 guarantees that N_{F} is small. The result is provided in the following lemma.

Lemma 4.2. *Algorithm 1 with* COMMIT *procedure* COMMITKNOWN *guarantees:*

$$\sum_{\ell \in \mathcal{L}} R(\ell) \leq 2N_{\mathrm{F}} + 6C + 3.$$

Intuitively, the proof separates the commitment regret on incorrect leaves from the one on the correct leaf. On an incorrect leaf, every successful safety check must be supported by at least one corruption; hence, apart from the final check that may trigger a backtrack, the number of commitment iterations on incorrect leaves is charged to the corruption budget. On the correct leaf, safety checks can fail only because of corrupted feedback, so after at most C such failures the algorithm reaches the true commitment phase and repeatedly posts a price with regret at most $1/T$ per round.

We conclude the section by providing the final regret guarantee when C is known to the learner. This is done in the following theorem, which is obtained by combining the previous results.

Theorem 4.3. *Assume that the feedback is corrupted in at most C rounds and C is known to the learner. Algorithm 1 with* COMMIT *procedure* COMMITKNOWN *guarantees:*

$$R_T \leq \mathcal{O}\left(C + \log T\right).$$

5 Handling Unknown Corruption

The setting with unknown corruption is arguably more challenging. To the best of our knowledge, for all the algorithms for this setting, the regret includes the product between the corruption C and a T-dependent component. In particular, the best result is the $\mathcal{O}\left((C + \log T) \log \log T\right)$ upper bound of Gupta et al. [2025].

In this setting, we need to balance the tension between two components. On the one hand, we want to exit quickly from a commit step on a wrong leaf,—in particular when the prices associated to that leaf are smaller than $v^\star$— which requires to play the right extreme. On the other hand, if we are in the optimal leaf, we want to play the right extreme as little as possible to avoid a large regret. To do that, we design a randomized commit strategy which works as follows.

We provide the pseudocode of our commitment procedure for the unknown-C setting in Algorithm 4. Specifically, COMMITUNKNOWN commits to different search blocks until an inconsistency is detected, in which case it returns FAIL. Each search block consists of two rounds. In the first round, the algorithm posts the left price L and observes the corresponding feedback. If $\sigma_L = 0$, then an inconsistency is detected and the algorithm returns FAIL. In the second round, the algorithm *explores* the right price with probability $q_{s_\ell} := \min\left\{\frac{4\ln(T/\delta)}{s_\ell}, 1\right\}$. Otherwise, the algorithm selects L for the additional round. Again, COMMITUNKNOWN returns FAIL whenever an inconsistency is detected. If posting R triggers a failure, we decrease s_ℓ by $4/q_{s_\ell}$, clipped at zero.

Finally, we remark that the counters s_ℓ are initialized to zero and shared across different calls to Algorithm 4, without any other reinitialization.

Algorithm 4 COMMITUNKNOWN

Require: Current leaf interval $\ell = [L, R) \in \mathcal{L}$, counters s_ℓ for each $\ell \in \mathcal{L}$, $\delta \in (0, 1)$

while TRUE **do**
  Post L and observe σ_L
  if $\sigma_L = 0$ **then return** FAIL
  end if
  $s_\ell \leftarrow s_\ell + 1$
  Sample $B \sim \mathrm{Bern}\left(\min\left\{\frac{4\ln(T/\delta)}{s_\ell}, 1\right\}\right)$
  if $B = 0$ **then**
    Post L and observe σ_L
    if $\sigma_L = 0$ **then return** FAIL
    end if
  else
    Post R and observe σ_R
    if $\sigma_R = 1$ **then**
      $s_\ell \leftarrow \max\{0, s_\ell - 4/q_{s_\ell}\}$
      return FAIL
    end if
  end if
end while

As for the known corruption case, the commit phase exits the optimal leaf only if a round is corrupted.

Observation 5.1. *Under commit procedure* COMMITUNKNOWN*, we have:*

$$N_{\mathrm{T}} \leq C.$$

Thus, we are left to bound the regret associated with the leaves, $\sum_{\ell \in \mathcal{L}} R(\ell)$, in order to obtain the final bound on the total regret R_T. Intuitively, this is done by splitting $\mathcal{L}$ into three components: the set $\mathcal{L}^+$ of intervals above $\ell^\star$, the correct leaf $\ell^\star$, and the set $\mathcal{L}^-$ of intervals below $\ell^\star$, and then bounding the corresponding regret terms separately.

We start bounding the regret on the optimal leaf.

Lemma 5.2. *Let $\delta \in (0,1)$. Algorithm 1 with* COMMIT *procedure* COMMITUNKNOWN *guarantees:*

$$R(\ell^\star) \leq 1 + 44C + 20 \ln T \, \ln(T/\delta),$$

with probability at least $1 - \delta/3$.

Intuitively, the regret on the correct leaf $\ell^\star = [L, R)$ is essentially due to the rounds in which the algorithm posts the right endpoint R. Indeed, by construction of the tree, the left endpoint satisfies $|L - v^\star| \leq 1/T$, so posting L over the whole horizon contributes at most constant regret. It remains to control the number N_R of times R is posted. Between counter decreases, the probability of selecting R decays as $\mathcal{O}(1/s_\ell)$. The sum of these probabilities is bounded by $\mathcal{O}(\log T \log(T/\delta))$, plus a constant for each counter decrease. Since each decrease on $\ell^\star$ requires a corrupted round, the additional contribution is $\mathcal{O}(C)$. Finally, employing the Bernstein's inequality gives, with high probability, the desired bound $R(\ell^\star) \leq \mathcal{O}\left(C + \log T \log(T/\delta)\right)$.

Then, we focus on bounding the regret associated with $\mathcal{L}^+$.

Lemma 5.3. *Algorithm 1 with* COMMIT *procedure* COMMITUNKNOWN *guarantees:*

$$\sum_{\ell \in \mathcal{L}^+} R(\ell) \leq 2N_{\mathrm{F}} + 2C.$$

For the leaves above $\ell^\star$, an uncorrupted first round immediately triggers a failure, since $L_\ell > v^\star$. Hence, there are at most N_{F} blocks with an uncorrupted first round. Every remaining block has a corrupted first round, so there are at most C such blocks. Since each block consists of at most two rounds, this gives $\sum_{\ell \in \mathcal{L}^+} R(\ell) \leq 2N_{\mathrm{F}} + 2C$.

We finally bound the regret associated with the set $\mathcal{L}^-$.

Lemma 5.4. *Let $\delta \in (0,1)$. Algorithm 1 with* COMMIT *procedure* COMMITUNKNOWN *guarantees:*

$$\sum_{\ell \in \mathcal{L}^-} R(\ell) \leq 64C + 129N_{\mathrm{F}} + 65,$$

with probability at least $1 - \delta/3$.

Bounding the regret for the leaves in $\mathcal{L}^-$ is the most delicate case. Indeed, posting the left endpoint L_ℓ does not reveal that the leaf is incorrect. Since $L_\ell \leq v^\star$, the learner observes a sale and may remain on the leaf. Posting the right endpoint R_ℓ in a non-corrupted block instead triggers a failure and decreases the counter. Therefore, if the learner spends many blocks on such a leaf while only few of them are corrupted or trigger a failure, the accumulated counter decreases must be unusually small. A martingale concentration argument bounds this event uniformly over all leaves and all possible values of $C_\ell + F_\ell$, where F_ℓ counts failed commitments on ℓ. This shows that the number of completed blocks spent on each $\ell \in \mathcal{L}^-$ is, with high probability, of order $C_\ell + F_\ell + 1$, and summing over leaves yields $\sum_{\ell \in \mathcal{L}^-} R(\ell) \leq \mathcal{O}\left(C + N_{\mathrm{F}} + 1\right)$.

Combining the previous results, we are ready to provide the final regret bound when C is unknown.

Theorem 5.5. *Let $\delta \in (0,1)$. Assume that the feedback is corrupted in at most C rounds. Algorithm 1 with* COMMIT *procedure* COMMITUNKNOWN *guarantees:*

$$R_T \leq \mathcal{O}\left(C + \log T \log(T/\delta)\right),$$

with probability at least $1 - \delta$.

Theorem 5.5 provides the main result of the paper, showing that Algorithm 1, when instantiated with the procedure COMMITUNKNOWN, attains a linear dependence on the corruption C even when C is *not* known to the learner, thus effectively decoupling corruption and time.

6 Discussion and Open Problems

In this work, we show that, in robust pricing, it is possible to decouple the dependence on the corruption level C from the time horizon T, resolving the open problem posed by Gupta et al. [2025]. By adopting a robustified binary-search approach augmented with tailored commitment phases, we obtain a regret bound of $\mathcal{O}(C + \log T)$ in the known-corruption setting. Furthermore, by introducing a novel randomized commitment strategy that carefully balances exploration and exploitation on candidate leaves, we achieve a regret of order $\mathcal{O}(C + \log^2 T)$ even when the corruption budget is unknown. Our results leave two main directions open. First, it remains to determine whether the $\mathcal{O}(C + \log^2 T)$ bound in the unknown corruption setting is tight, either by proving a matching lower bound or by developing improved algorithms. Second, an important direction is to extend our techniques to the contextual setting considered by Krishnamurthy et al. [2023], Gupta et al. [2025].

Acknowledgments

This publication was funded with the contribution of Ministero dell'Università e della ricerca pursuant to D.D. n. 18010 of 12 November 2025 – BANDO FIS 3. Project FIS-2024-05736 (Starting Grant), title: "Towards a trustworthy strategic use of data in machine learning pipelines" (STRATDATA). CUP: D53C25002380001.

References

Kareem Amin, Afshin Rostamizadeh, and Umar Syed. Learning prices for repeated auctions with strategic buyers. In C. J. C. Burges, Léon Bottou, Max Welling, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1169–1177. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/hash/996a7fa078cc36c46d02f9af3bef918b-Abstract.html.

Omar Besbes and Assaf Zeevi. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations Research*, 57(6):1407–1420, 2009. doi: 10.1287/opre.1080.0640.

Nicolò Cesa-Bianchi, Claudio Gentile, and Yishay Mansour. Regret minimization for reserve prices in second-price auctions. *IEEE Transactions on Information Theory*, 61(1):549–564, 2015. doi: 10.1109/TIT.2014.2365772.

Maxime C. Cohen, Ilan Lobel, and Renato Paes Leme. Feature-based dynamic pricing. *Management Science*, 66(11):4921–4943, 2020. doi: 10.1287/mnsc.2019.3485.

Yuval Dagan, Yuval Filmus, Daniel Kane, and Shay Moran. The entropy of lies: Playing twenty questions with a liar. In *12TH INNOVATIONS IN THEORETICAL COMPUTER SCIENCE CONFERENCE, ITCS 2021*, volume 185, 2021.

Alexey Drutsa. Horizon-independent optimal pricing in repeated auctions with truthful and strategic buyers. In Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich, editors, *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 33–42. ACM, 2017. doi: 10.1145/3038912.3052700. URL https://doi.org/10.1145/3038912.3052700.

Anupam Gupta, Guru Guruganesh, Renato Paes Leme, and Jon Schneider. Robust contextual pricing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=eHm1EdLypn.

Adel Javanmard and Hamid Nazerzadeh. Dynamic pricing in high-dimensions. *Journal of Machine Learning Research*, 20(9):1–49, 2019. URL https://jmlr.org/papers/v20/17-357.html.

Robert Kleinberg and Frank Thomson Leighton. The value of knowing a demand curve: bounds on regret for online posted-price auctions. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 594–605. IEEE, 2003.

Akshay Krishnamurthy, Thodoris Lykouris, Chara Podimata, and Robert Schapire. Contextual search in the presence of adversarial corruptions. *Operations Research*, 71(4):1120–1135, 2023.

Renato Paes Leme and Jon Schneider. Contextual search via intrinsic volumes. *SIAM Journal on Computing*, 51(4):1096–1125, 2022. doi: 10.1137/20M1385718.

Renato Paes Leme, Chara Podimata, and Jon Schneider. Corruption-robust contextual search through density updates. In Po-Ling Loh and Maxim Raginsky, editors, *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 3504–3505. PMLR, 02–05 Jul 2022. URL https://proceedings.mlr.press/v178/leme22a.html.

Renato Paes Leme, Chara Podimata, and Jon Schneider. Density-based algorithms for corruption-robust contextual search and convex optimization. *arXiv preprint arXiv:2206.07528*, 2026. doi: 10.48550/arXiv.2206.07528. URL https://doi.org/10.48550/arXiv.2206.07528.

Allen Liu, Renato Paes Leme, and Jon Schneider. Optimal contextual pricing and extensions. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1059–1078. SIAM, 2021. doi: 10.1137/1.9781611976465.66.

Ilan Lobel, Renato Paes Leme, and Adrian Vladu. Multidimensional binary search for contextual decision-making. *Operations Research*, 66(5):1346–1361, 2018. doi: 10.1287/opre.2018.1722.

R. L. Rivest, A. R. Meyer, D. J. Kleitman, K. Winklmann, and J. Spencer. Coping with errors in binary search procedures. *Journal of Computer and System Sciences*, 20(3):396–404, 1980. doi: 10.1016/0022-0000(80)90014-8.

Stanislaw M. Ulam. *Adventures of a Mathematician*. Scribner, New York, 1976. ISBN 0684143917.

Contents

1 Introduction **1**

1.1 Our Results and Techniques 2

1.2 Relation to Robust Binary Search 3

2 Preliminaries **3**

2.1 Robust Dynamic Pricing 4

3 Meta Algorithm **4**

3.1 Regret Decomposition and Potential Argument 5

4 Warm-Up: The Known Corruption Case **7**

5 Handling Unknown Corruption **8**

6 Discussion and Open Problems **10**

A Additional Related Works **12**

B Robust Binary Search **13**

C Omitted Proofs of Section 3 **14**

D Omitted Proofs of Section 4 **15**

E Omitted Proofs of Section 5 **16**

A Additional Related Works

Learning to price was initiated by the seminal work of Kleinberg and Leighton [2003], which provided matching upper and lower bounds for the regret achievable in the classical dynamic pricing problem. In particular, they showed that, in the uncorrupted one-dimensional setting, the optimal regret scales as $\Theta(\log\log T)$. This doubly-logarithmic dependence on the horizon is one of the most distinctive features of dynamic pricing, and reflects the fact that binary-search-type strategies are exceptionally well aligned with the threshold structure induced by the unknown buyer valuation. Dynamic pricing have been then extended to stochastic or partially specified demand models [Besbes and Zeevi, 2009, Javanmard and Nazerzadeh, 2019] and to richer settings such as pricing in repeated-auction and under strategic behavior [Amin et al., 2013, Cesa-Bianchi et al., 2015, Drutsa, 2017].

A further line of work studied higher-dimensional variants of the same learning problem under binary threshold feedback, often under the broader name of *contextual pricing* or *contextual search*. In these settings, the buyer's valuation depends on side information describing the item, but the learner still observes only one-bit sale/no-sale feedback and faces the same asymmetric pricing loss. A sequence of works progressively sharpened the achievable regret bounds in this richer model, including Cohen et al. [2020], Lobel et al. [2018], Leme and Schneider [2022], and Liu et al. [2021], culminating in the $\mathcal{O}(d\log\log T)$ guarantee of Liu et al. [2021], where d is the dimension of the context.

A closely related research line concerns learning under corrupted feedback. This line of work studies settings in which the underlying structure of the problem remains well defined, but a bounded number of observations may be unreliable or even chosen adversarially. Classical antecedents of this idea go back to Ulam's game [Ulam, 1976]. More recently, corruption-robust guarantees have been investigated for higher-dimensional search and pricing problems under binary feedback. In

particular, Krishnamurthy et al. [2023] pioneered the study of corrupted contextual pricing through the lens of corrupted multidimensional binary search, obtaining guarantees of order $\mathcal{O}(d^3(C + \log T)\log T \log(d/\varepsilon))$ in that more general setting. Subsequently, Leme et al. [2022] introduced a density-based approach to corruption-robust contextual search, obtaining regret $\mathcal{O}(C + d\log(1/\varepsilon))$ for the ε-ball loss and $\mathcal{O}(C + d\log T)$ for a symmetric loss. Furthermore, the extended version of this work [Leme et al., 2026] clearly notes that extending density-based methods to the dynamic pricing setting appears to require significant novel approaches. The most closely related work is Gupta et al. [2025], which sharpens guarantees for robust contextual pricing, obtaining regret $\mathcal{O}(Cd\log\log T)$ under known corruption, ruling out $\mathcal{O}(C+d\log\log T)$, and achieving $\mathcal{O}(C+\log T)$ in the one-dimensional cautious-buyer model with one-sided corruptions.

B Robust Binary Search

In this section, we show that the robust binary-search technique of [Rivest et al., 1980] can be used to attain a $\mathcal{O}(C + \log T)$ regret bound for the known-corruption version of our pricing problem. Specifically, Rivest et al. [1980] study the problem of identifying an unknown element $x \in \{1, \dots, n\}$ by using comparison queries, when up to C answers may be erroneous and C is known. Their theorem is usually stated in the sharper form:

$$\log n + C\log\log n + \mathcal{O}(C\log C).$$

For our purposes, however, the exact upper bound in their Theorem 2 is more convenient. In particular, their result implies that the number of comparisons needed in the worst case is at most:

$$Q(n, C) < \min\left\{Q' : 2^{Q'-C} > n\sum_{i=0}^{C}\binom{Q'-C}{i}\right\}. \tag{1}$$

We now show that this expression is of order $\mathcal{O}(C + \log n)$.

Lemma B.1. *For every $n \geq 2$ and every $C \geq 0$, the robust binary-search procedure of Rivest et al. [1980] identifies an unknown element $x \in \{1, \dots, n\}$ under at most C erroneous comparison answers using $\mathcal{O}(\log n + C)$ comparisons in the worst case.*

Proof. We use the exact upper bound from Rivest et al. [1980] provided in Equation (1). Thus, to prove the lemma, it is enough to exhibit a value $Q' = \mathcal{O}(C + \log n)$ such that:

$$2^{Q'-C} > n\sum_{i=0}^{C}\binom{Q'-C}{i}.$$

Choose $Q' = C + \lceil 16(\lceil\log n\rceil + C)\rceil$. In the following, we show that this value satisfies the required condition.

If $C = 0$, then $\sum_{i=0}^{C}\binom{Q'-C}{i} = 1$, and the condition reduces to $2^{Q'} > n$, which holds by the choice of Q'. Hence assume $C \geq 1$.

By construction, it holds:

$$\frac{C}{Q'-C} \leq \frac{C}{16(\lceil\log n\rceil + C)} \leq \frac{1}{16}.$$

Let $H(p) = -p\log p - (1-p)\log(1-p)$ be the binary entropy function. Since $C/(Q'-C) \leq 1/16 \leq 1/2$, the standard entropy bound for binomial sums gives:

$$\sum_{i=0}^{C}\binom{Q'-C}{i} \leq 2^{(Q'-C)H(C/(Q'-C))}.$$

The function H is increasing on $[0, 1/2]$, so $H(C/(Q'-C)) \leq H(1/16) < 1/2$. Therefore,

$$\sum_{i=0}^{C}\binom{Q'-C}{i} \leq 2^{(Q'-C)/2}, \quad n\sum_{i=0}^{C}\binom{Q'-C}{i} \leq 2^{\lceil\log n\rceil}2^{(Q'-C)/2}$$

Moreover, again by the choice of Q',

$$\frac{Q'-C}{2} = \frac{\lceil 16(\lceil \log n \rceil + C) \rceil}{2} > \lceil \log n \rceil.$$

Hence $2^{\lceil \log n \rceil} 2^{(Q'-C)/2} < 2^{Q'-C}$. Combining the last two inequalities gives:

$$n \sum_{i=0}^{C} \binom{Q'-C}{i} < 2^{Q'-C}.$$

Thus, the condition in the upper bound of Rivest et al. [1980] is satisfied. Consequently,

$$Q(n, C) \leq C + \lceil 16\big(\lceil \log n \rceil + C\big) \rceil = \mathcal{O}(\log n + C).$$

This concludes the proof. □

We now translate this robust-search guarantee into a regret guarantee.

Theorem B.2. *Consider the robust dynamic pricing problem with valuation $v^\star \in [0,1]$, horizon T, and at most C corrupted feedback observations. If C is known to the learner, then there exists a pricing algorithm with regret:*

$$R_T \leq \mathcal{O}(C + \log T).$$

Proof. The result follows by partitioning $[0,1]$ into T intervals of length $1/T$ and running the algorithm of Rivest et al. [1980] over this discretization. Once the optimal interval has been identified, the algorithm commits to the left endpoint of that interval, incurring regret at most $1/T$ on each subsequent round. Since during the search each round incurs regret at most 1, the final bound is:

$$R_T \leq Q(n, C) + \frac{1}{T} \cdot T \leq \mathcal{O}(C + \log(T)),$$

by Lemma B.1 with $n = T$. This concludes the proof. □

C Omitted Proofs of Section 3

Lemma 3.1. *Every honest search step q at a non-leaf interval $I^q \notin \mathcal{L}$ decreases the potential by exactly one, i.e., $\Phi_{q+1} = \Phi_q - 1$.*

Proof. We distinguish two cases depending on whether I^q is (i) on-path or off-path:

(i) On-path. Since the step is honest, SAFETYCHECK(I^q) returns PASS and the midpoint query is uncorrupted. Hence, the algorithm moves to the unique on-path child of I^q, which is one step closer to $\ell^\star$. Thus, $\Phi_{q+1} = \Phi_q - 1$.

(ii) Off-path. Being $I^q = [L, R)$ off-path, either $R \leq v^\star$ or $L > v^\star$. In either case, SAFETYCHECK(I^q) returns FAIL, and the algorithm backtracks to the parent interval of I^q. Since $\ell^\star$ is not in the subtree rooted at I^q, the parent is one step closer to $\ell^\star$. Therefore, $\Phi_{q+1} = \Phi_q - 1$.

This concludes the proof. □

Lemma 3.2. *Every corrupted search step q at a non-leaf interval $I^q \notin \mathcal{L}$ increases the potential by at most one.*

Proof. In a single search step, the algorithm either moves to a child interval or backtracks to its parent interval. Consequently, the distance to $\ell^\star$ changes by one in absolute value. Therefore, $\Phi_{q+1} \leq \Phi_q + 1$. □

Lemma 3.3. *Every commit search step q at a leaf interval $\ell^q \in \mathcal{L}$:*

- *Reduces the potential by 1 if $\ell^q \neq \ell^\star$*
- *Increases the potential by 1 if $\ell^q = \ell^\star$.*

Proof. A commit search step terminates when the commit procedure returns FAIL and the algorithm backtracks to the parent interval. If $\ell^q \neq \ell^*$, the leaf is not the target, and its parent lies on the path to ℓ^*, decreasing the distance from $\ell^\star$ and the potential by 1. If $\ell^q = \ell^*$, the algorithm moves from the target interval to its parent (which has distance 1 from $\ell^\star$). Hence, the potential is increased by 1. □

Theorem 3.4. *Assume that the feedback is corrupted in at most C rounds. Then, Algorithm 1 limits the number of commitment steps on incorrect leaves to $N_{\mathrm{F}} \leq \log T + C + N_{\mathrm{T}}$. Furthermore, its regret is at most:*

$$R_T \leq \sum_{\ell \in \mathcal{L}} R(\ell) + 3 \log T + 6C + 3N_{\mathrm{T}}.$$

Proof. Let H and K denote the number of honest and corrupted search steps occurring at non-leaf intervals, respectively. Since each corrupted search step contains at least one corrupted round, we have $K \leq C$. Let $q^\star$ be the final search step of the algorithm. We can express the final potential using a telescoping sum of the potential changes:

$$\Phi_{q^\star} = \Phi_0 + \sum_{q \in [q^\star]} \Phi_q - \Phi_{q-1}.$$

Applying Lemma 3.1, Lemma 3.2, and Lemma 3.3 to bound the potential change in different types of search steps and observing that $\Phi_0 = D$ we get:

$$\Phi_{q^\star} = \Phi_0 + \sum_{q \in [q^\star]} \Phi_q - \Phi_{q-1} \leq D + K - H + N_{\mathrm{T}} - N_{\mathrm{F}}$$

Since the potential is by definition non negative, $\Phi_{q^\star} \geq 0$, and hence:

$$N_{\mathrm{F}} + H \leq D + K + N_{\mathrm{T}}.$$

By observing that $K \leq C$ and $D = \log T$, we get:

$$N_{\mathrm{F}} + H \leq \log T + C + N_{\mathrm{T}}. \tag{2}$$

This immediately yields the first result:

$$N_{\mathrm{F}} \leq D + K + N_{\mathrm{T}}.$$

To bound the total regret, note that search step at a non-leaf interval consists of at most three rounds (two for the safety check and one for the midpoint), with each round incurring a regret of at most 1. We decompose the total regret into the regret from the regret on commitment steps and search step at non-leaf interval:

$$R_T \leq \sum_{\ell \in \mathcal{L}} R(\ell) + 3(H+K) \leq \sum_{\ell \in \mathcal{L}} R(\ell) + 3(\log T + C + N_{\mathrm{T}} + C) = \sum_{\ell \in \mathcal{L}} R(\ell) + 3\log(T) + 6C + 3N_{\mathrm{T}},$$

where in the second inequality we used Equation (2) and $K \leq C$. This concludes the proof. □

D Omitted Proofs of Section 4

Lemma 4.2. *Algorithm 1 with* COMMIT *procedure* COMMITKNOWN *guarantees:*

$$\sum_{\ell \in \mathcal{L}} R(\ell) \leq 2N_{\mathrm{F}} + 6C + 3.$$

Proof. The regret of the commitment phase is the sum of regret on incorrect leaves and on the correct leaf $\ell^\star$.

Consider a fixed incorrect leaf $\ell = [L, R) \in \mathcal{L} \setminus \{\ell^*\}$. At any iteration of the commit loop at leaf ℓ with $s_\ell \leq C$, either the safety check fails and backtracks, or it passes and increments s_ℓ. If it passes, then at least one of the checks on the two extremes is corrupted. Hence, it must be the case that at the end of time horizon holds $s_\ell \leq C$. This implies that on a commit search step on an incorrect leaf, it is always the case that $s_\ell \leq C$. This implies that for a commit search step, all the iterations of the

loop except (possibly) the last one are corrupted, namely, if the step last for a iteration of the loop then the corruption is at least $a-1$. Since each iteration includes exactly two rounds we get:

$$\sum_{\ell \in \mathcal{L} \setminus \ell^\star} R(\ell) \leq 2C + 2N_{\mathrm{F}}.$$

Consider now the correct leaf $\ell^\star = [L, R)$. As we noticed in Observation 4.1, the iterations of the commit procedure on $\ell^\star$ that fail and return are at most C. Hence, after at most $2C+1$ iteration, we get that $s_\ell > C$ and the algorithm commit to L. Since each iteration of the safety check requires 2 rounds, the regret before committing to L is at most $4C+2$. Finally, once the algorithm commit to L gets per-round regret at most $1/T$ since $v^\star - 1/T \leq L \leq v^\star$.

Overall, we get:

$$\sum_{\ell \in \mathcal{L}} R(\ell) \leq 2C + 2N_{\mathrm{F}} + 4C + 2 + T\frac{1}{T} \leq 2N_{\mathrm{F}} + 6C + 3.$$

This concludes the proof. □

Theorem 4.3. *Assume that the feedback is corrupted in at most C rounds and C is known to the learner. Algorithm 1 with* COMMIT *procedure* COMMITKNOWN *guarantees:*

$$R_T \leq \mathcal{O}\left(C + \log T\right).$$

Proof. Combining our results we get:

$$\begin{aligned} R_T &\leq \sum_{\ell \in \mathcal{L}} R(\ell) + 3\log T + 6C + 3N_{\mathrm{T}} \\ &\leq 2N_{\mathrm{F}} + 6C + 3 + 3\log T + 3N_{\mathrm{T}} + 6C \\ &\leq 2(\log T + C + N_{\mathrm{T}}) + 6C + 3 + 3\log T + 3N_{\mathrm{T}} + 6C \\ &\leq 5\log T + 19C + 3, \end{aligned}$$

where in the first inequality we use Theorem 3.4, in the second inequality we use Lemma 4.2, in the third again Theorem 3.4, and in the fourth Observation 4.1. This concludes the proof. □

E Omitted Proofs of Section 5

Lemma 5.2. *Let $\delta \in (0,1)$. Algorithm 1 with* COMMIT *procedure* COMMITUNKNOWN *guarantees:*

$$R(\ell^\star) \leq 1 + 44C + 20\ln T\ \ln(T/\delta),$$

with probability at least $1-\delta/3$.

Proof. To bound the regret associated to $\ell^\star = [L, R)$, we first notice that, by structure of the tree built by Algorithm 1, it holds $|L - v^\star| \leq \frac{1}{T}$, thus the regret over the entire learning dynamic incurred when the left point is chosen is bounded by 1. Thus, in order to bound $R(\ell^\star)$, we need to bound, in high probability, the number of times R is selected. Now, let N_R be the number of times the right prices have been posted.

To account for the counter decreases, we consider the quantity:

$$4\ln(T/\delta)\ln\left(1 + \frac{s_\ell}{4\ln(T/\delta)}\right).$$

For $T \geq 3$, a direct computation shows that each increment of the counter increases this quantity by at least $q_{s_\ell}/2$, where s_ℓ is the value after the increment, whereas each decrease reduces it by at most 11. On the correct leaf, this can happen only if a round is corrupted, so there are at most C decreases. Since the counter starts at zero and never exceeds T, we can bound the sum of the conditional probabilities of selecting R as follows:

$$\sum q_{s_\ell} \leq 8\ln(T/\delta)\ln\left(1 + \frac{T}{4\ln(T/\delta)}\right) + 22C \leq 8\ln T\ln(T/\delta) + 22C =: \overline{Q},$$

where the sum is over the second rounds played on $\ell^\star$.

Now, we want to relate this bound to the true value. Since the draws are conditionally Bernoulli and the sum of their conditional selection probabilities is bounded by $\overline{Q}$, the conditional version of Bernstein's inequality gives that, for every $x > 0$:

$$\mathbb{P}\left(N_R \geq \overline{Q} + \sqrt{2\overline{Q}x} + \frac{x}{3}\right) \leq e^{-x}.$$

Applying this with $x := \ln(3/\delta)$, we obtain:

$$\mathbb{P}\left(N_R \geq \overline{Q} + \sqrt{2\overline{Q}\ln(3/\delta)} + \frac{1}{3}\ln(3/\delta)\right) \leq \frac{\delta}{3}.$$

Using $\sqrt{2\overline{Q}\ln(3/\delta)} \leq \overline{Q} + \frac{1}{2}\ln(3/\delta)$, we get that, with probability at least $1 - \delta/3$:

$$N_R \leq 2\overline{Q} + \frac{5}{6}\ln(3/\delta).$$

Since $T \geq 3$ implies $\ln(3/\delta) \leq \ln(T/\delta)$, it follows that, with probability at least $1 - \delta/3$, $N_R \leq 44C + 20\ln T\ \ln(T/\delta)$. This leads to a final regret bound of order:

$$R(\ell^\star) \leq 1 + 44C + 20\ln T\ \ln(T/\delta),$$

which holds with probability at least $1 - \delta/3$. □

Lemma 5.3. *Algorithm 1 with* COMMIT *procedure* COMMITUNKNOWN *guarantees:*

$$\sum_{\ell \in \mathcal{L}^+} R(\ell) \leq 2N_{\mathrm{F}} + 2C.$$

Proof. When $\ell \in \mathcal{L}^+$, the bound follows from the following observations. The regret on these leaves is bounded by the total number of rounds spent on them, and these rounds can be grouped into blocks of at most two, since the commit procedure may return after the first round. We now decompose these blocks into two types. The first type corresponds to blocks whose first round is uncorrupted. For a leaf $\ell \in \mathcal{L}^+$, the first round of every block posts L_ℓ. Since $L_\ell > v^\star$, if that round is not corrupted then the learner observes a no-sale and immediately leaves the leaf. Hence the total number of blocks of the first type is at most N_{F}, where N_{F} denotes the total number of failed commitments on non-optimal leaves. The second type corresponds to blocks whose first round is corrupted. It follows that the total number of blocks of the second type, summed over all leaves in $\mathcal{L}^+$, is at most C, including a possible final unfinished block. Since each block consists of at most two rounds which implies a regret at most 2, we conclude:

$$\sum_{\ell \in \mathcal{L}^+} R(\ell) \leq 2N_{\mathrm{F}} + 2C.$$

This concludes the proof. □

Lemma 5.4. *Let $\delta \in (0, 1)$. Algorithm 1 with* COMMIT *procedure* COMMITUNKNOWN *guarantees:*

$$\sum_{\ell \in \mathcal{L}^-} R(\ell) \leq 64C + 129N_{\mathrm{F}} + 65,$$

with probability at least $1 - \delta/3$.

Proof. Bounding the regret for the leaves in $\mathcal{L}^-$ requires a different approach compared to the previous cases. Unlike for Lemma 5.3, we cannot directly relate the number of times L is played to the corruption because, in the absence of corruption, posting L does not trigger any backtracking. Furthermore, unlike for Lemma 5.2, we do not need to show that the probability of playing R is small; instead, we must show that this probability is large enough to detect as soon as possible whether the leaf has experienced corruption.

To do so, for every $\ell \in \mathcal{L}^-$ we denote by N_ℓ the total number of completed two-round search blocks spent on ℓ, by C_ℓ the total number of corrupted search blocks experienced on ℓ, and by F_ℓ the number of times the commitment procedure returns FAIL on ℓ. Notice that these are random variables and,

in general, they are not independent. We consider $T \geq 3$, since the statement is immediate for $T \leq 2$. For every integer $h \geq 0$, let $m_h := 32(h+1)$, and define the event $\mathcal{E}_{\ell,h} := \{N_\ell > m_h,\ C_\ell + F_\ell \leq h\}$.

We first show that $\mathbb{P}(\mathcal{E}_{\ell,h})$ is small. Fix $\ell \in \mathcal{L}^-$ and fix $h \geq 0$. On the event $\mathcal{E}_{\ell,h}$, the learner plays the first m_h completed blocks on ℓ, possibly across different visits. Each non-corrupted query of R_ℓ triggers a failure and decreases the counter, whereas each completed block increases the counter by one before the second round.

We now formalize this argument. Let $\mathcal{F}_n$ denote the history up to the moment in which the learner randomizes in the second round of the n-th completed block on ℓ, including the adversary's corruption decision, and let B_n be the Bernoulli random variable indicating whether the learner posts R_ℓ in that round. Since the counter can only decrease between consecutive increments, its value before this draw is at most n. Thus, conditional on $\mathcal{F}_n$, we have:

$$\mathbb{P}(B_n = 1 \mid \mathcal{F}_n) = q_n, \qquad q_n \geq \min\left\{\frac{4\ln(T/\delta)}{n}, 1\right\}.$$

Let S denote the set of indices whose second round is corrupted.

We now sum the counter updates over the first n completed blocks. Each non-corrupted query of R_ℓ decreases the counter by $4/q_n$, unless the update is clipped at zero. Since $4\ln(T/\delta) > 4$, clipping can occur only when $q_n = 1$, and reduces the decrement by at most 4 per failure. The counter starts at zero, receives n increments and stays nonnegative. Therefore, we obtain:

$$\sum_{\substack{i \leq n \\ i \notin S}} \frac{B_i}{q_i} \leq \frac{n}{4} + F_\ell.$$

Since at most C_ℓ of these second rounds are corrupted, at least $n - C_\ell$ of them are uncorrupted. On $\mathcal{E}_{\ell,h}$, taking $n = m_h = 32(h+1)$, we also have $C_\ell + F_\ell \leq h < n/32$. Thus, the previous bound gives:

$$\sum_{\substack{i \leq n \\ i \notin S}} \left(1 - \frac{B_i}{q_i}\right) \geq \frac{3}{4}n - C_\ell - F_\ell \geq \frac{23}{32}n.$$

To bound $\mathbb{P}(\mathcal{E}_{\ell,h})$, we therefore bound the probability that the sum above is at least $23n/32$.

Since $\mathbb{E}[B_i \mid \mathcal{F}_i] = q_i$, each term $1 - B_i/q_i$ has conditional mean zero and is at most 1. Its conditional variance satisfies:

$$\mathrm{Var}\left(1 - \frac{B_i}{q_i} \,\middle|\, \mathcal{F}_i\right) = \frac{1}{q_i} - 1 \leq \max\left\{\frac{i}{4\ln(T/\delta)} - 1, 0\right\},$$

where the inequality follows from the lower bound on q_i. In particular, the variance is zero when $i \leq 4\ln(T/\delta)$, because then $q_i = 1$.

To apply Freedman's inequality, we keep these increments only in the first m_h second rounds on ℓ, using zero on corrupted rounds and in all other rounds. Whether a round is corrupted or belongs to ℓ is determined before its Bernoulli draw. Thus, the increments still have conditional mean zero, even when the visits to ℓ depend on previous feedback.

If $m_h \leq 4\ln(T/\delta)$, the sum is identically zero and cannot reach $23m_h/32$, so $\mathcal{E}_{\ell,h}$ is empty. Otherwise, summing the variance bounds gives:

$$2\sum_{i=1}^{m_h} \max\left\{\frac{i}{4\ln(T/\delta)} - 1, 0\right\} + \frac{23m_h}{48} \leq \frac{m_h^2}{4\ln(T/\delta)}.$$

The left-hand side bounds the denominator in Freedman's inequality: twice the sum of the conditional variances, plus $2/3$ of the deviation $23m_h/32$. Applying the Freedman's inequality, we obtain:

$$\mathbb{P}(\mathcal{E}_{\ell,h}) \leq \exp\left(-\frac{(23m_h/32)^2}{m_h^2/[4\ln(T/\delta)]}\right) \leq \exp\big(-2\ln(T/\delta)\big) = \left(\frac{\delta}{T}\right)^2,$$

where we used $(23/32)^2 \geq 1/2$ in the last inequality.

We can now take a union bound over all $\ell \in \mathcal{L}^-$ and all integers $h \geq 0$. Since $N_\ell \leq T/2$, only the values of h satisfying $m_h \leq T/2$ can give nonempty events. There are at most $T/64$ such values and at most $2T$ leaves. Thus, we obtain:

$$\mathbb{P}\left(\bigcup_{\ell \in \mathcal{L}^-} \bigcup_{h \geq 0} \mathcal{E}_{\ell,h}\right) \leq \frac{T^2}{32}\left(\frac{\delta}{T}\right)^2 = \frac{\delta^2}{32} \leq \frac{\delta}{3}.$$

Therefore, with probability at least $1 - \delta/3$, none of the events $\mathcal{E}_{\ell,h}$ occurs. In what follows, we work on this good event.

Fix now a visited leaf $\ell \in \mathcal{L}^-$. Since none of the events $\mathcal{E}_{\ell,h}$ occurs, we may choose $h = C_\ell + F_\ell$. If it were true that $N_\ell > m_{C_\ell + F_\ell}$, then this would imply that $\mathcal{E}_{\ell, C_\ell + F_\ell}$ occurs, contradicting the good event. Therefore, we can conclude that $N_\ell \leq m_{C_\ell + F_\ell}$, which implies:

$$N_\ell \leq 32(C_\ell + F_\ell + 1),$$

with probability at least $1 - \delta/3$.

We can now bound the regret associated with the set $\mathcal{L}^-$. Again, we remark that each completed search block consists of two rounds, and each round contributes regret at most 1. Blocks that fail at the first round contribute at most F_ℓ, and there can be at most one final unfinished block over the entire horizon. Since at most $N_{\mathrm{F}} + 1$ leaves in $\mathcal{L}^-$ are visited, summing over all leaves gives:

$$\begin{aligned}\sum_{\ell \in \mathcal{L}^-} R(\ell) &\leq 2 \sum_{\ell \in \mathcal{L}^-} N_\ell + N_{\mathrm{F}} + 1 \\ &\leq 64C + 65N_{\mathrm{F}} + 64(N_{\mathrm{F}} + 1) + 1 \\ &\leq 64C + 129N_{\mathrm{F}} + 65.\end{aligned}$$

This concludes the proof. □

Lemma E.1. *Let $\delta \in (0, 1)$. Algorithm 1 with* COMMIT *procedure* COMMITUNKNOWN *guarantees:*

$$\sum_{\ell \in \mathcal{L}} R(\ell) \leq 66 + 20 \ln T \, \ln(T/\delta) + 131 N_{\mathrm{F}} + 110C,$$

with probability at least $1 - \delta$.

Proof. The statement follows from combining Lemmas 5.2, 5.3, 5.4, with a final union bound. □

Theorem 5.5. *Let $\delta \in (0, 1)$. Assume that the feedback is corrupted in at most C rounds. Algorithm 1 with* COMMIT *procedure* COMMITUNKNOWN *guarantees:*

$$R_T \leq \mathcal{O}\left(C + \log T \log(T/\delta)\right),$$

with probability at least $1 - \delta$.

Proof. Combining the previous results we get:

$$\begin{aligned}R_T &\leq \sum_{\ell \in \mathcal{L}} R(\ell) + 3 \log T + 6C + 3N_{\mathrm{T}} \\ &\leq 66 + 20 \ln T \, \ln(T/\delta) + 131 N_{\mathrm{F}} + 110C + 3 \log T + 6C + 3N_{\mathrm{T}} \\ &\leq 66 + 20 \ln T \, \ln(T/\delta) + 131(\log T + C + N_{\mathrm{T}}) + 110C + 3 \log T + 6C + 3N_{\mathrm{T}} \\ &\leq 66 + 20 \ln T \, \ln(T/\delta) + 131 \log T + 131C + 131C + 110C + 3 \log T + 6C + 3C \\ &\leq 66 + 20 \ln T \, \ln(T/\delta) + 134 \log T + 381C,\end{aligned}$$

where in the first inequality we use Theorem 3.4, in the second inequality we use Lemma E.1, which holds with probability at least $1 - \delta$, in the third again Theorem 3.4, and in the fourth Observation 5.1. This concludes the proof. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state all the main contributions made by the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: All the assumptions are clearly stated in Section 2.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All the theoretical results clearly state their assumptions, while all their proofs are provided in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [N/A]

   Justification: The paper does *not* include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [N/A]

Justification: The paper does *not* include experiments.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [N/A]

   Justification: The paper does *not* include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: The paper does *not* include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [N/A]

   Justification: The paper does *not* include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed, since the work is mainly theoretical.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The paper does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.