No Price Tags? No Problem: Query Strategies for Unpriced Information

Shivam Nadimpalli [*] Mingda Qiao [†] Ronitt Rubinfeld [‡]

November 11, 2025

Abstract

The classic *priced query model* introduced by Charikar et al. (STOC 2000) captures the task of computing a known function on an unknown input, where each input variable can only be revealed by paying an associated cost. The goal is to design a query strategy that determines the function's value while minimizing the total cost incurred. All prior work in this model, however, assumes complete advance knowledge of these query costs—an assumption that breaks down dramatically in many realistic settings.

We introduce a variant of the priced query model designed explicitly to handle *unknown* variable costs. We prove a separation from the traditional priced query model, showing that uncertainty in variable costs imposes an unavoidable overhead for every query strategy. Despite this, we are able to give query strategies that essentially match our lower bound and are competitive with the best-known cost-aware strategy for an arbitrary Boolean function. Our results build on a recent connection between priced query strategies and the analysis of Boolean functions, and also draw on techniques from online algorithms.

[*]MIT. Email: shivamn@mit.edu.

[†]University of Massachusetts Amherst. Email: mingda.qiao.cs@gmail.com.

[‡]MIT. Email: ronitt@csail.mit.edu.

1 Introduction

Information is rarely free, and its value is often revealed only after the cost to acquire it has been paid. A scientist may devote weeks to running an experiment, only to find the results inconclusive; a data analyst may purchase costly annotations that add little insight; a policymaker may commission new surveys, learning only afterward which questions truly mattered. Even in familiar settings—medical diagnostics or crowd-sourced data collection—each "query" carries a cost and a payoff that remain unknown until a response has been obtained. The same tension arises in collective decision-making: for instance, the U.S. Electoral College can be viewed as a "majority of majorities" function, where confident prediction demands focusing effort on the few states whose outcomes remain uncertain. Additionally, costly observations may sometimes yield no new information, whereas a single inexpensive one can sometimes decisively influence the outcome.

The same fundamental difficulty runs through all these scenarios: an overall outcome depends on many unknown variables—not all of which may need to be learned—yet revealing each variable carries a cost that is itself unknown in advance. The challenge is to allocate effort strategically: which variables to reveal, and in what order, when each becomes known only once its cost threshold has been met. To capture this problem, we introduce a theoretical model of information acquisition under unknown query costs. Our framework extends the classical *priced query* model of Charikar, Fagin, Guruswami, Kleinberg and Raghavan [CFG+00] to an "online" setting, where the cost required to reveal a variable is hidden until the cumulative investment surpasses its threshold. We begin by briefly recalling the classical model.

Background: Query Strategies for Priced Information. The *priced query* model introduced in [CFG+00] provides a clean abstraction for decision-making when query costs are known in advance. In this model, the goal is to evaluate a Boolean function $f : \{0,1\}^n \to \{0,1\}$ on an unknown input $x \in \{0,1\}^n$, where each input bit can only be revealed by paying a specified cost. The challenge is to design query strategies that minimize the total cost incurred by the algorithm.

Over the past two decades, there has been substantial work designing priced query strategies for various function classes; an incomplete list includes [CL05a, CL05b, CL08, CGLM11, CL11, CM11, DHK14, AHKÜ17, GGHK18, Hel18, BLT21, HKLW22, HLS24, GGN24] among many others. The work of Charikar et al. [CFG+00] also inspired priced variants of other basic algorithmic tasks such as learning [KKM05], optimization [Sin18], and search or sorting [GK01, KK03]. The problem has also been extensively studied within the operations research community under the label of *sequential testing* [Ünl04].

This Work: Query Strategies for *Unpriced* Information. The classical model, however, assumes that the algorithm knows all query costs in advance—an assumption that fails dramatically in many natural settings. In practice, the cost of uncovering information is often uncertain: we discover how expensive a variable is only through the process of revealing it. This brings us to the focus of the present work:

Can we design effective query strategies for *unpriced* information?

In other words, we consider the task of designing query strategies in the setting when the variable costs are *entirely unknown* to the algorithm. We study this problem in the framework of competitive analysis, where our goal is to design query strategies whose total cost is comparable to that of the optimal algorithm that knows the variable costs in advance. Throughout, we refer to this setting as the "online" model, to distinguish it from the "offline" (known-cost) framework of [CFG+00].

We note that our problem bears resemblance to several models in the online algorithms literature, such as threshold queries and Pandora's box. These online frameworks incorporate uncertainty and sequential revelation, yet they typically do not aim to evaluate a fixed Boolean function; the emphasis there is often on regret or reward objectives rather than computing a prescribed function of the inputs. Conversely, the classical priced-query model assumes complete knowledge of all costs in advance. (We refer the reader to Section 1.2 for a detailed discussion of related work.) Our work bridges these two areas, both conceptually and technically.

Algorithmically, we draw on ideas from the query-strategies literature—particularly influence-based techniques from the analysis of Boolean functions, building on a recent connection highlighted by Blanc, Lange, and Tan [BLT21]—as well as tools from online algorithms, including martingale arguments and testing/consistency checks. In addition to our algorithmic contributions, we establish lower bounds for both specific strategies and general online query algorithms. We believe that the ideas and techniques developed here will find broader applications across these domains.

1.1 Our Results

For the sake of readability, we restrict ourselves to our main algorithmic results and lower bounds in the discussion below. We summarize all our results in Table 1, and give a detailed technical overview in Section 2.

1.1.1 Problem Setup

We start with an informal overview of our setup and refer the reader to Section 3.2 for a precise formulation. Throughout, we assume that the goal is to compute a Boolean function $f : \{0,1\}^n \to \{0,1\}$ on an unknown input $x \in \{0,1\}^n$. We also denote the fixed (but unknown) vector of costs as $c \in \mathbb{R}^n_{\geq 0}$. At each time step, the algorithm may invest in a single variable; its cumulative investment is maintained as a vector $\theta \in \mathbb{R}^n$. The variable x_i is revealed to the algorithm when $\theta_i \geq c_i$.

We consider both *zero-error* algorithms (i.e. algorithms that correctly compute $f(x)$ on *every* input $x \in \{0,1\}^n$) as well as ε*-error* algorithms (i.e. algorithms which are allowed to err in computing $f(\boldsymbol{x})$ for ε-fraction of $\boldsymbol{x} \sim \{0,1\}^n$). We benchmark our algorithms in terms of both the optimal *average cost* and the *worst-case cost* among all offline algorithms. For example, $\mathrm{opt}_0^{\mathrm{avg}} := \mathrm{opt}_0^{\mathrm{avg}}(f, c)$ will denote the optimal expected cost among all zero-error offline algorithms computing $f(\boldsymbol{x})$ on a uniformly random $\boldsymbol{x} \sim \{0,1\}^n$; we correspondingly define $\mathrm{opt}_\varepsilon^{\mathrm{avg}}$, $\mathrm{opt}_0^{\mathrm{w}}$, and $\mathrm{opt}_\varepsilon^{\mathrm{w}}$ with the latter two being worst-case costs. See Definition 15 for formal definitions of these performance measures, and note for now that

$$\mathrm{opt}_\varepsilon^{\mathrm{avg}} \leq \mathrm{opt}_0^{\mathrm{avg}} \leq \mathrm{opt}_0^{\mathrm{w}} \qquad \text{and} \qquad \mathrm{opt}_\varepsilon^{\mathrm{avg}} \leq \mathrm{opt}_\varepsilon^{\mathrm{w}} \leq \mathrm{opt}_0^{\mathrm{w}}\,. \tag{1}$$

Thus, benchmarking against $\mathrm{opt}_\varepsilon^{\mathrm{avg}}$ is the strongest guarantee we can hope for.

Much of our focus will be on the average-case setting; we will thus refer to the *competitive ratio* against the benchmark $\mathrm{opt}_\diamond^\star$ as the ratio between the expected cost of algorithm on a uniformly random $\boldsymbol{x} \sim \{0,1\}^n$ and the benchmark $\mathrm{opt}_\diamond^\star$ where $\star, \diamond$ will always be clear from context.

1.1.2 Separating the Offline and Online Settings

We first show that the online setting is strictly harder than the offline setting of [CFG+00]. In particular, the following result gives a lower bound on the competitive ratio of average-case online algorithms against the benchmark $\mathrm{opt}_0^{\mathrm{avg}}$, even when the online algorithm is allowed to err on a constant fraction of inputs.

Function $f : \{0,1\}^n \to \{0,1\}$	Error	Benchmark	Competitive Ratio
AND (Proposition 39)	0	$\mathrm{opt}_0^{\mathrm{avg}}$	$\Omega(n)$
Tribes (Theorem 1)	$\varepsilon \in [0, 1/4)$	$\mathrm{opt}_0^{\mathrm{avg}}$	$\Omega(\log n)$
General (Theorem 7)	ε	$\mathrm{opt}_0^{\mathrm{w}}$	$O\left(\frac{\mathbf{I}[f] \log n}{\varepsilon}\right)$
Symmetric (Theorem 5)	ε	$\mathrm{opt}_0^{\mathrm{avg}}$	$O\left(\log \frac{1}{\varepsilon}\right)$
General (Theorem 4)	$O(\varepsilon)$	$\mathrm{opt}_\varepsilon^{\mathrm{avg}}$	$O\left(\frac{\mathbf{I}[f] \log n}{\varepsilon^3} \cdot \log\left(\frac{\log \mathrm{opt}_\varepsilon^{\mathrm{avg}}}{\varepsilon}\right)\right)$
D.T. with avg. depth d (Theorem 6)	ε	$\mathrm{opt}_0^{\mathrm{avg}}$	$O\left(\frac{d}{\varepsilon}\right)$

Table 1: An overview of our bounds on the competitive ratio in the online setting. The upper bounds hold for specific online algorithms that satisfy the corresponding error bounds, and D.T. is shorthand for decision tree. See also Table 2 for bounds on the total influence $\mathbf{I}[f]$ (cf. Definition 2) for some function classes. The lower bounds hold against *all* online algorithms with the corresponding error guarantees.

Theorem 1 (Informal version of Theorem 41)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be the* Tribes *function on n variables (cf. Definition 40). Then every* 0.24999*-error online algorithm computing f has competitive ratio $\Omega(\log n)$ against $\mathrm{opt}_0^{\mathrm{avg}}$.*

The Tribes function is a well-known extremal function in Boolean function analysis and social choice theory [BOL85, KKL88, O'D14]. Note that that a lower bound on the competitive ratio against $\mathrm{opt}_0^{\mathrm{avg}}$ is a stronger guarantee than comparing against $\mathrm{opt}_\varepsilon^{\mathrm{avg}}$ (cf. Equation (1)). En route to proving Theorem 1, we give an $\Omega(n)$ lower bound on the competitive ratio against $\mathrm{opt}_0^{\mathrm{avg}}$ for zero-error online algorithms computing the AND function (Proposition 39).

1.1.3 Online Query Algorithms

Despite Theorem 1, we give online algorithms that match (up to a logarithmic factor) the performance of the best known offline strategy for a *generic* Boolean function $f : \{0,1\}^n \to \{0,1\}$ obtained by Blanc, Lange, and Tan [BLT21]. Prior to [BLT21], much of the work on priced query strategies focused on designing algorithms tailored to structured classes of Boolean functions such as disjunctions, DNF formulas, game trees, halfspaces, and so on. In contrast, Blanc, Lange, and Tan [BLT21] give an offline algorithm whose performance can be expressed in terms of a basic complexity measure of the function f, namely its *total influence* $\mathbf{I}[f]$:

Definition 2. Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, we define the *influence of the i^{th} variable on f* as

$$\mathbf{Inf}_i[f] := \Pr_{\boldsymbol{x} \sim \{0,1\}^n}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i})\right]$$

where $x^{\oplus i} = (x_1, \ldots, x_{i-1}, 1 - x_i, x_{i+1}, \ldots, x_n)$. Furthermore, we define the *total influence of f* as

$$\mathbf{I}[f] := \sum_{i=1}^{n} \mathbf{Inf}_i[f].$$

Family of functions $f : \{0,1\}^n \to \{0,1\}$	$\mathbf{I}[f]$	Reference
Monotone functions	$O(\sqrt{n})$	[BT96]
Intersections of s halfspaces	$O(\sqrt{n \log s})$	[Kan14]
Size-s DNFs	$O(\log s)$	[Bop97]
Depth-d size-s AC^0 circuits	$O(\log s)^{d-1}$	[LMN93, Bop97]

Table 2: Upper bounds on the total influence of various function classes.

We note that both the coordinate influences and total influences are central notions across a number of fields including combinatorics, complexity theory, social choice, and statistical physics; see [Kal04, Juk12, O'D14, GS14] for more information. See Table 2 for bounds on the total influence of structured function classes including DNF formulas, circuits, intersections of halfspaces, and monotone functions.

From [BLT21] to Unknown Costs. We start by recalling [BLT21]'s guarantee in the offline setting:

Theorem 3 (Theorem 1 of [BLT21])**.** *Let $f : \{0,1\}^n \to \{0,1\}$, $c \in \mathbb{N}^n$ be a* known *cost vector, and $\varepsilon \in (0, 0.5)$. There is an efficient $O(\varepsilon)$-error query strategy for f with expected cost at most*

$$\mathrm{opt}_{\varepsilon}^{\mathrm{avg}} \cdot \frac{\mathbf{I}[f]}{\varepsilon^2}.$$

The algorithm of Blanc, Lange, and Tan [BLT21] establishing Theorem 3 follows a simple but powerful strategy: repeatedly query the variable x_i that maximizes the ratio of its influence $\mathbf{Inf}_i[f]$ to its (known) cost c_i, updating the function with its restrictions as coordinates are revealed. A natural first attempt in the *unknown-cost* setting is to adapt this rule by maintaining a cumulative investment vector θ proportional to the vector of coordinate influences.

This serves as a warm-up to our main algorithm: we show that this natural online adaptation is competitive with respect to the weakest benchmark, $\mathrm{opt}_0^{\mathrm{w}}$, up to a multiplicative factor of the total influence; see Theorem 18 for a formal statement. Even in this preliminary setting, our analysis already departs from [BLT21]: we require a martingale argument to control the evolution of partial influences as costs are gradually revealed.

However, recall from Equation (1) that $\mathrm{opt}_0^{\mathrm{w}}$ is the weakest benchmark we can hope to be competitive against. If we wish to compete against the stronger quantities $\mathrm{opt}_{\varepsilon}^{\mathrm{avg}}$ or even $\mathrm{opt}_0^{\mathrm{avg}}$, this specific "influence-proportional" query strategy *provably fails*. In particular, we construct instances where its competitive ratio is $\mathbf{I}[f] \cdot \widetilde{\Omega}(\sqrt{n})$, polynomially worse than the offline guarantee achieved by [BLT21] (cf. Theorem 3). We refer the reader to Proposition 21 for the formal statement and proof, and to Figure 1 for an illustration of the hard instance, which is based on a modification of the classical "address function" from Boolean function analysis [O'D14].

Our Main Algorithmic Result. Our main result builds on the online adaptation of [BLT21]'s algorithm, while taking into account the specific structure of the hard instance discussed above. In essence, the $\mathrm{poly}(n)$ degradation relative to Theorem 3 arises from the algorithm's *local* stopping condition: the algorithm described above terminates once the function becomes ε-close to constant.

While this stopping condition suffices for benchmarking against $\mathrm{opt}_0^{\mathrm{w}}$, we show that it fails to be competitive against the benchmark $\mathrm{opt}_\varepsilon^{\mathrm{avg}}$, as local termination can lead to significant over-investment when the input comes from a small but "difficult" subset of the hypercube, whereas the optimal ε-error strategy could safely ignore or "give up" on this region.

This motivates our main algorithmic contribution: we continue to use influence as the guiding statistic but replace the local stopping rule with a more *global* termination condition that aggregates progress across coordinates. The modification may appear minor, but its analysis is substantially more intricate: in addition to the martingale argument from the warm-up algorithm, it requires additional "testing" and "consistency" steps. We give a detailed technical overview of these ideas in Section 2 and state our main result below.

Theorem 4 (Informal version of Theorem 16 and Proposition 24)**.** *Let $f : \{0,1\}^n \to \{0,1\}$, $\varepsilon \in (0, 0.5)$, and $c \in \mathbb{R}^n_{\geq 0}$ be an* unknown *cost vector. There is an efficient algorithm* Online-Query *(cf. Algorithm 4) which computes f to error $O(\varepsilon)$ and an expected cost of*

$$\mathrm{opt}_\varepsilon^{\mathrm{avg}} \cdot O\left(\frac{\mathbf{I}[f] \log n}{\varepsilon^3} \cdot \log\left(\frac{\log \mathrm{opt}_\varepsilon^{\mathrm{avg}}}{\varepsilon}\right)\right).$$

Theorem 4 thus extends the guarantee of [BLT21] to the online setting, matching its performance up to logarithmic factors. We note that in both Theorem 3 and Theorem 4, efficiency is measured with respect to the given representation of the function $f : \{0,1\}^n \to \{0,1\}$.

Beyond the General Case. Beyond Theorem 4, which holds for arbitrary Boolean functions, we also design online query strategies for two basic and well-studied families of functions in the query-strategies literature: *symmetric functions*, and functions computed by *low-depth decision trees*. We start with our guarantee for the former:

Theorem 5 (Informal version of Theorem 27)**.** *For any symmetric function $f : \{0,1\}^n \to \{0,1\}$,* Warmup-IPRR$(f, \varepsilon)$ *(Algorithm 2) is an efficient ε-error algorithm with an expected cost of*

$$\mathrm{opt}_0^{\mathrm{avg}} \cdot O\left(\log \frac{1}{\varepsilon}\right).$$

We note that many natural Boolean functions including the AND, OR, and Majority functions are symmetric. Indeed, previous works [GGHK18, Hel18, HKLW22, HLS24] have specifically designed query strategies for symmetric functions. We also note that the $O(\log(1/\varepsilon))$ competitive ratio cannot be improved in general: when $\varepsilon = 2^{-(n+1)} < 2^{-n}$, the algorithm Warmup-IPRR(f, ε) is a zero-error algorithm and computes any symmetric function with an expected cost of $\mathrm{opt}_0^{\mathrm{avg}} \cdot O(n)$. (The algorithm Warmup-IPRR is deterministic, so its error probability on $\boldsymbol{x} \sim \{0,1\}^n$ must be a multiple of 2^{-n}.) This matches our lower bound for the AND function (Proposition 39).

We also design query strategies for functions which can be represented as a decision tree with small average depth. We consider the setting where the decision tree is given to the query algorithm, as well as the setting where the small average depth decision tree is unknown.

Theorem 6 (Informal version of Theorem 36 and Remark 37)**.** *There is an ε-error algorithm that, given a decision tree representation of f with average depth d, has expected cost*

$$\mathrm{opt}_0^{\mathrm{avg}} \cdot \frac{d}{\varepsilon}.$$

Moreover, only assuming that f has a decision tree representation of average depth d, there is an ε-error algorithm that, without knowing d or the decision tree in advance, runs in time $\mathrm{poly}(n) \cdot (d/\varepsilon)^{O(d/\varepsilon)}$ *and gives an expected cost of*

$$\mathrm{opt}_0^{\mathrm{avg}} \cdot O\left(\frac{d}{\varepsilon}\right).$$

In general, the bound given by Theorem 6 is incomparable to the $\mathrm{opt}_0^{\mathrm{w}} \cdot O((\mathbf{I}[f] \log n)/\varepsilon)$ bound guaranteed by Theorem 7. Note that Theorem 6 is stronger in that it competes with the average-case benchmark $\mathrm{opt}_0^{\mathrm{avg}}$, yet the d/ε competitive ratio can be either higher or lower than $(\mathbf{I}[f] \log n)/\varepsilon$. Indeed, note that while the bound $d \geq \mathbf{I}[f]$ holds in general, both $d = \mathbf{I}[f]$ (e.g., parity functions) and $d \gg \mathbf{I}[f]$ (e.g., Tribes) can be true.

1.2 Related Work

We briefly survey related work in both the query-strategies and online-algorithms literature. In addition, our notion that a variable is revealed once the cumulative investment in it exceeds an unknown cost threshold bears a superficial resemblance to "threshold queries" studied in differential privacy and related areas [BSU17, CLN+24], though the connection appears to be largely informal.

Query Strategies for Priced Information. The work of Charikar et al. [CFG+00] measured the performance of a query strategy on an instance-by-instance basis. Specifically, for each unknown input $x \in \{0,1\}^n$, they compare the cost incurred by an algorithm $\mathcal{A}$ to the minimal cost of a certificate for $f(x)$.[1] Their goal, then, is to design algorithms that minimize this ratio for all $x \in \{0,1\}^n$. Their approach thus gives a worst-case instance-by-instance guarantee; in contrast, we compare the *expected cost* of $\mathcal{A}$ on a uniformly random input $\boldsymbol{x} \sim \{0,1\}^n$ to the optimal expected cost. Charikar et al. [CFG+00] give query strategies for functions computable by AND-OR trees, with subsequent works considering other function classes (e.g. game trees, monotone functions, and threshold functions) as well as related algorithmic tasks such as search and sorting with prices. A partial and incomplete list includes [GK01, KK03, CL05a, CL05b, CL08, CL11, CM11, CGLM11], among many others.

The work of Kaplan, Kushilevitz, and Mansour [KKM05] considers query strategies within learning models with attribute costs. Their problem formulation can be viewed as an average-case version of that of [CFG+00], where an algorithm's performance is measured by comparing the expected cost of a strategy to the expected cost of the cheapest certificate for $f(\boldsymbol{x})$. Notably, [KKM05] consider arbitrary distributions on $\{0,1\}^n$ but restrict their attention to studying functions represented by disjunctions, CDNF formulas, and read-once DNF formulas.

Our problem formulation is inspired by that of the *Stochastic Boolean Function Evaluation* problem considered by Deshpande, Hellerstein, and Keletenik [DHK14], with a long line of subsequent work [BDHK18, AHKÜ17, Hel18, BLT21, GHKL22, HKLW22, HLS24, GGN24]. As in our setup, the work of [DHK14] measures the performance of an algorithm according to the expected cost on a random input $\boldsymbol{x}$ drawn from a (known) product distribution on $\{0,1\}^n$. However, much of the focus has been on designing algorithms tailored to structured function classes such as CDNF formulas, decision trees, and linear threshold functions.

The recent work of Blanc, Lange, and Tan [BLT21] considers the setup of [DHK14], but differs from previous results in two notable ways: first, they allow the algorithm to err on a small fraction

[1] A *certificate* for $f(x)$ corresponds to a subset of coordinates $S \subseteq [n]$ whose values, when fixed according to x, completely determine the value of the function as $f(x)$.

of inputs; and second, they design algorithms that apply to *all* Boolean functions, with performance guarantees in terms of the total influence of the function. Our work aligns with theirs in both of these aspects.

A key distinction of our work is that all previous algorithms for querying priced information—including those listed above—assume that the costs of revealing input variables are known to the algorithm beforehand.

Online Algorithms Our work also connects to the broader literature on online algorithms. In particular, the problem we consider shares conceptual similarities with both classical and contemporary models such as Pandora's Box [Wei78] and the Markovian price-of-information [GJSS19]. A growing body of recent work [Sin18, CGT+20, QV23, DNS23] further develops related ideas in online decision-making under cost constraints, and the literature in this area is by now too extensive to survey comprehensively. However, as mentioned earlier, the objectives in these models are typically *reward-based*—aimed at maximizing expected value—rather than computing a specific, known function of an unknown input.

1.3 Discussion

Much remains to be understood even within the online priced query model introduced in this paper. A substantial body of work has designed query strategies tailored to specific classes of Boolean functions in the offline setting. This includes monotone functions [CL05b], AND-OR and game trees [CFG+00, CL05b], subclasses of DNFs [KKM05, DHK14, AHKÜ17], threshold functions and their generalizations [DHK14, GGHK18, GGN24], and symmetric functions [GGHK18, Hel18, HLS24]. In particular, additional structural information about the target function can often be effectively leveraged to design better query strategies in the offline setting. It is natural to ask whether similar improvements can be obtained in the online setting.

We further highlight two promising directions for future investigation:

- **Partial Information Settings**: The model considered in this paper assumes that the algorithm begins with no knowledge of the variable costs. However, a natural extension is to consider *partial information* settings, where the algorithm has access to some side information about the costs—perhaps through a distributional prior, historical data, or external advice. Exploring how such partial knowledge affects the design and performance of online query strategies could likely yield connections to Bayesian optimization, bandit algorithms, and learning with advice.

- **Dynamic Costs:** In many real-world scenarios the variable costs may evolve dynamically, driven by e.g. external market forces or time. In such settings, the algorithm must reason not only about *which* queries to make, but also about *when* to make them. We believe that modeling such settings and designing effective query strategies with shifting prices pose interesting challenges, and are likely to reveal new connections to online learning and decision-making under uncertainty.

1.4 Organization

We give a technical overview of our results in Section 2. After recalling preliminaries in Section 3, we prove Theorems 4 and 5 in Section 4. We then give our decision tree-based query strategies establishing Theorem 6 in Section 5. Finally, we prove separations between the offline and online setting (Theorem 1) in Section 6.

2 Technical Overview

Our starting point is the greedy algorithm of Blanc, Lange, and Tan [BLT21] for the offline setting, where the variable costs are known to the algorithm. Their key idea is to use the *influence* of a variable (Definition 2) as a proxy for its "importance" in determining the value of function: the algorithm keeps querying the variable with the highest influence-to-cost ratio, until a certain stopping condition is met.

An Online Algorithm: Influence-Proportional Round Robin. When the costs are unknown, we use a natural strategy: *invest in the variables in proportion to their influences.* At each step, we examine the restricted function f_π induced by the input bits that have been revealed so far. We identify the variable x_i that maximizes the ratio

$$\frac{\mathbf{Inf}_i[f_\pi]}{\theta_i},$$

where $\mathbf{Inf}_i[f_\pi]$ is the influence of x_i with respect to f_π, and θ_i denotes the current investment in x_i. Then, we increment θ_i and, if x_i gets revealed as a result, we update f_π with the knowledge of x_i. In other words, we keep investing in the most under-invested variable relative to its influence, maintaining the invariant that the investment in each variable is roughly proportional to its influence. We term this query strategy "Influence-Proportional Round Robin" (IPRR).

The only remaining design choice is a stopping criterion. We start with a simple one: Let ε be the target error probability. The algorithm terminates whenever f_π is ε-close to a constant function, at which point it outputs the rounding of $\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_\pi(\boldsymbol{x})\right]$ to $\{0,1\}$. We call the resulting algorithm Warmup-IPRR.

Input: Succinct representation of $f:\{0,1\}^n\to\{0,1\}$, error parameter $\varepsilon\in(0,0.5]$
Output: A bit $b\in\{0,1\}$

Warmup-IPRR(f,ε):

1. Initialize $\theta\leftarrow 0^n$ and $\pi\leftarrow\emptyset$.
2. While $\mathrm{bias}(f_\pi)>\varepsilon$:
   (a) Let $i^*\in[n]$ be the index such that $i^*=\arg\max_i \frac{\mathbf{Inf}_i[f_\pi]}{\theta_i}$.
   (b) Increment θ_{i^*}. If x_{i^*} is revealed to be $b_{i^*}\in\{0,1\}$, update $\pi\leftarrow\pi\cup\{x_{i^*}\mapsto b_{i^*}\}$.
3. Output $\mathbf{1}\left\{\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_\pi(\boldsymbol{x})\right]\geq 1/2\right\}$.

Algorithm 1: The Warmup-IPRR Algorithm (Informal version of Algorithm 2)

We establish the following guarantee on the performance of Warmup-IPRR:

Theorem 7 (Informal version of Theorem 18)**.** *The algorithm* Warmup-IPRR(f,ε) *(cf. Algorithm 2) is an ε-error algorithm for computing $f:\{0,1\}^n\to\{0,1\}$ with an expected cost of*

$$\mathrm{opt}_0^{\mathrm{w}}\cdot O\left(\frac{\mathbf{I}[f]\cdot\log n}{\varepsilon}\right).$$

The analysis of WARMUP-IPRR in Theorem 7 is quite different from that of the greedy algorithm of [BLT21]. Roughly speaking, the proof of [BLT21] uses the total influence of the restricted function as a measure of progress in computing the function value. They show that, whenever the greedy algorithm queries a variable x_i, the amount of progress is at least $\varepsilon/\mathrm{opt}_0^{\mathrm{w}}$ times the cost of x_i. Since the algorithm makes a progress of $\leq \mathbf{I}[f]$ in total, the total cost paid by the algorithm is upper bounded by $(\mathrm{opt}_0^{\mathrm{w}}/\varepsilon) \cdot \mathbf{I}[f] = \mathrm{opt}_0^{\mathrm{w}} \cdot (\mathbf{I}[f]/\varepsilon)$.

However, in the online setting, the analogous claim no longer holds for the WARMUP-IPRR algorithm. Before observing any input bits, WARMUP-IPRR might have already invested substantially in all the n variables, making the total investment *much* higher than the progress gained from revealing a single bit. This suggests that the "progress-to-cost" ratio cannot be lower bounded as in the offline setting.

Instead, our analysis controls the expected investment in each of the n variables separately. We show that, when WARMUP-IPRR terminates, the investment in each variable x_i is at most $\mathrm{opt}_0^{\mathrm{w}}/\varepsilon$ times the maximum of $\mathbf{Inf}_i[f_\pi]$, where π ranges over all restrictions that WARMUP-IPRR encounters. In general, this maximum influence *can* be much higher than $\mathbf{Inf}_i[f]$. Fortunately, the trajectory of $\mathbf{Inf}_i[f_\pi]$ throughout the execution forms a martingale bounded between $[0,1]$, which allows us to upper bound the expectation of $\max_\pi \mathbf{Inf}_i[f_\pi]$ by $\mathbf{Inf}_i[f]$ up to a logarithmic factor. Summing over all $i \in [n]$ gives the upper bound of $\mathrm{opt}_0^{\mathrm{w}} \cdot O(\mathbf{I}[f]\log(n)/\varepsilon)$.

An Improved Bound for Symmetric Functions. We prove Theorem 5 by comparing how the optimal zero-error offline algorithm and the online algorithm WARMUP-IPRR work on a symmetric function. Knowing the costs of the variables, the optimal offline algorithm queries the variables in increasing order of costs. Furthermore, since the algorithm must have a zero error, it stops only if function f reduces to a constant given the observed inputs.

For our online algorithm, since every variable is equally influential with respect to a symmetric function (or a restriction thereof), WARMUP-IPRR maintains the same investment level across all inputs that have not been revealed. Consequently, the variables also get revealed in increasing order of costs. There are two differences: (1) WARMUP-IPRR stops as soon as the restricted function become ε-close to a constant; (2) the algorithm pays an additional cost for each variable that is not revealed at the end. Theorem 5 follows from analyzing the stopping times of these two different yet similar processes.

Competing Against Average-Case Benchmarks. To derive our algorithm that competes against $\mathrm{opt}_\varepsilon^{\mathrm{avg}}$ in Theorem 4, we take a detour and see why the WARMUP-IPRR algorithm fails to even compete with $\mathrm{opt}_0^{\mathrm{avg}}$ without incurring an $\widetilde{\Omega}(\sqrt{n})$ competitive ratio:

Proposition 8 (Informal version of Proposition 21)**.** *There is a function $f : \{0,1\}^n \to \{0,1\}$ such that, for any accuracy parameter $\varepsilon \in (0, 1/4)$,* WARMUP-IPRR(f, ε) *gives an expected cost of*

$$\mathrm{opt}_0^{\mathrm{avg}} \cdot \mathbf{I}[f] \cdot \widetilde{\Omega}(\sqrt{n}).$$

The hard instance in Proposition 8 is constructed as follows. First, we find a function $g(x)$ such that $\mathbf{I}[g] = \Theta(\sqrt{n})$, while WARMUP-IPRR pays an expected cost of $\mathrm{opt}_0^{\mathrm{avg}}(g) \cdot \widetilde{\Omega}(\sqrt{n})$ on g.[2] Then, we "dilute" $g(x)$ by constructing an alternative function $f(x, y)$ such that, over the randomness in y: (1) with probability $p := 1/\sqrt{n}$, $f(x, y)$ is set to $g(x)$; (2) with probability $1 - p$, $f(x, y) = h(y)$ for some simple function h. Furthermore, each additional input bit in y has a cost of zero. This dilution ensures that: (1) $\mathrm{opt}_0^{\mathrm{avg}}(f) = \mathrm{opt}_0^{\mathrm{avg}}(g)/\sqrt{n}$, since the offline algorithm for $f(x, y)$ needs to

[2]Note that g alone does not give the hard instance in Proposition 8.

evaluate $g(x)$ only with probability $p = 1/\sqrt{n}$; (2) $\mathbf{I}[f] \le p \cdot \mathbf{I}[g] + O(1) = O(1)$; (3) The cost of WARMUP-IPRR on f is a p-fraction of its cost on g, namely,

$$p \cdot \left[\mathrm{opt}_0^{\mathrm{avg}}(g) \cdot \widetilde{\Omega}(\sqrt{n})\right] = \mathrm{opt}_0^{\mathrm{avg}}(g) \cdot \widetilde{\Omega}(1) = \mathrm{opt}_0^{\mathrm{avg}}(f) \cdot \mathbf{I}[f] \cdot \widetilde{\Omega}(\sqrt{n}).$$

The hard instance above exploits the fact that WARMUP-IPRR uses a *local* stopping criterion: the algorithm stops only if the current restriction is ε-close to a constant function. Suppose that $\varepsilon = \Omega(1)$. In the rare case that $f(x, y)$ is determined by $g(x)$ (which happens with probability $p = 1/\sqrt{n} \ll \varepsilon$), WARMUP-IPRR would still pay the cost for evaluating $g(x)$. On the other hand, a more "global" algorithm might realize that $f(x, y)$ is ε-close to $h(y)$, so it might as well compute the simpler function h instead.

Indeed, our algorithm for competing against $\mathrm{opt}_\varepsilon^{\mathrm{avg}}$, denoted by IPRR, simply replaces the stopping condition of WARMUP-IPRR: The algorithm takes an additional parameter B as input, and stops by outputting the value $\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_\pi(\boldsymbol{x})\right]$ rounded to $\{0, 1\}$ whenever f_π—the current restriction of the function—satisfies the condition[3]

$$\frac{\mathbf{Inf}_i[f_\pi]}{c_i} \le \frac{\varepsilon}{B}, \ \forall i \in [n].$$

Via an analysis similar to that of Theorem 7, we show that IPRR(B) gives an $O(\varepsilon)$ error as long as $B \ge \mathrm{opt}_\varepsilon^{\mathrm{avg}}/\varepsilon$, and the resulting expected cost is $B \cdot O((\mathbf{I}[f] \log n)/\varepsilon)$. In particular, the first part of Theorem 4 follows from setting $B = \mathrm{opt}_\varepsilon^{\mathrm{avg}}/\varepsilon$, so that the expected cost is $\mathrm{opt}_\varepsilon^{\mathrm{avg}} \cdot O((\mathbf{I}[f] \log n)/\varepsilon^2)$. When $\mathrm{opt}_\varepsilon^{\mathrm{avg}}$ is unknown, a natural strategy is to guess the value $B = 2^1, 2^2, 2^3, \ldots$ through doubling. For each guess of B, we check whether IPRR(B) is $O(\varepsilon)$-error by sampling $\widetilde{O}(1/\varepsilon)$ inputs uniformly at random from $\{0, 1\}^n$, and estimate the empirical error of IPRR(B). If the error is indeed $O(\varepsilon)$, we run IPRR(B) on the actual unknown input x. This ensures that we stop with high probability as soon as the guess of B reaches $\mathrm{opt}_\varepsilon^{\mathrm{avg}}/\varepsilon$. This testing procedure introduces an additional $\widetilde{O}(1/\varepsilon)$ factor in the competitive ratio.

Functions with Small-Depth Decision Trees. We prove Theorem 6 using algorithms that are significantly different from the IPRR strategy. Rather than simultaneously investing in multiple variables in proportion to their influences, we focus on finding an "efficient" decision tree representation of the function, so that "following the tree" (i.e., querying variables as suggested by the decision tree) *always* leads to a low competitive ratio, regardless of the variable costs.

Formally, we show that, if the given decision tree representation T is *everywhere τ-influential* (i.e., every node in the tree queries a variable with influence $\ge \tau$), simply following tree T gives a $(1/\tau)$-competitive algorithm. More generally, if T has average depth d and is not guaranteed to be everywhere-influential, we apply the pruning lemma of [BLQT22] to transform T into an everywhere (ε/d)-influential tree T' while introducing an error $\le \varepsilon$. Then, following the pruned tree T' gives a (d/ε)-competitive algorithm. Finally, when T is not given, the learning algorithm of [BLQT22] allows us to compute an everywhere $\Omega(\varepsilon/d)$-influential decision tree that approximates f up to an error of ε. This algorithm only requires query access to f, and runs in time $(d/\varepsilon)^{O(d/\varepsilon)}$. The knowledge of d can be further removed by a standard doubling trick.

Competitive Ratio Lower Bounds. We first prove the following lower bound against zero-error algorithms computing the AND function:

[3]This can be verified by checking whether θ_i exceeds $(B/\varepsilon) \cdot \mathbf{Inf}_i[f_\pi]$ for every unrevealed variable x_i.

Proposition 9 (Informal version of Proposition 39)**.** *Let* $f : \{0,1\}^n \to \{0,1\}$ *be the* AND *function on* n *variables. Then every zero-error online algorithm computing* f *has expected cost* $\mathrm{opt}_0^{\mathrm{avg}} \cdot \Omega(n)$.

We prove Proposition 9 by choosing the costs of the n variables in the AND function as a uniformly random permutation of $[n] = \{1, 2, \ldots, n\}$. The optimal offline algorithm queries the variables in ascending order of costs and incurs an $O(1)$ cost in expectation. The online algorithm, however, does not know which variables are the cheapest. Intuitively, the best strategy is to invest in the n variables in a round robin fashion, which incurs an $\Omega(n)$ cost even before seeing a single input bit.

For Theorem 1, recall that an n-variable Tribes function is the OR of $n/w = \Theta(n/\log n)$ disjoint tribes, each of which is the AND of $w = \Theta(\log n)$ variables. We choose the costs of the variables in each tribe as a random permutation of $[w]$, so that the Tribes instance consists of n/w independent copies of the AND instance in Proposition 9. To lower bound the cost of the online algorithm, we show that any low-error algorithm for Tribes must observe at least one variable from an $\Omega(1)$-fraction of the tribes. Then, by planting a w-variable AND instance as one of the n/w tribes, we obtain an algorithm for the AND function that reveals at least one of the variables with probability $\Omega(1)$. By a strengthening of Proposition 9, such an algorithm for AND must have an $\Omega(w)$ competitive ratio, which then translates into an $\Omega(\log n)$ competitive ratio for the Tribes instance.

3 Preliminaries

We use boldfaced letters (e.g. $\boldsymbol{x} \sim \{0,1\}^n$) to denote random variables. Unless explicitly stated otherwise, all probabilities and expectations will be with respect to the uniform distribution. All logarithms will be with respect to base-2, unless explicitly stated otherwise. We write $[n] := \{1, \ldots, n\}$ and $\{e_i\}_{i=1}^n$ for the collection of standard basis vectors in $\mathbb{R}^n$, i.e. $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$.

3.1 Boolean Functions

Our notation and terminology follow [O'D14]. Given two Boolean functions $f, g : \{0,1\}^n \to \{0,1\}$, we define

$$\mathrm{dist}(f, g) := \mathop{\mathbf{Pr}}_{\boldsymbol{x} \sim \{0,1\}^n} [f(\boldsymbol{x}) \neq g(\boldsymbol{x})] \,. \tag{2}$$

Given $f : \{0,1\}^n \to \{0,1\}$, we will write

$$\mathrm{bias}(f) := \min\{\mathrm{dist}(f, 0), \mathrm{dist}(f, 1)\} \,.$$

We recall the notion a variable's *influence* on a Boolean function (see Chapter 2 of [O'D14] for further background and information) defined earlier in Section 1:

Definition 2. Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, we define the *influence of the* i^{th} *variable on* f as

$$\mathbf{Inf}_i[f] := \mathop{\mathbf{Pr}}_{\boldsymbol{x} \sim \{0,1\}^n} \left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i}) \right]$$

where $x^{\oplus i} = (x_1, \ldots, x_{i-1}, 1 - x_i, x_{i+1}, \ldots, x_n)$. Furthermore, we define the *total influence of* f as

$$\mathbf{I}[f] := \sum_{i=1}^{n} \mathbf{Inf}_i[f].$$

Given a query algorithm $\mathcal{A}$, it will be convenient to write

$$\mathrm{error}_f(\mathcal{A}) := \mathrm{dist}(f, \mathcal{A})$$

where we identify $\mathcal{A} : \{0,1\}^n \to \{0,1\}$ with the decision tree induced by its computation. We will sometimes say that "$\mathcal{A}$ is an ε-error query algorithm for f" if $\mathrm{error}_f(\mathcal{A}) \leq \varepsilon$. For every index $i \in [n]$, we write

$$\delta_i(\mathcal{A}) := \mathop{\mathbf{Pr}}_{\boldsymbol{x} \sim \{0,1\}^n} [\mathcal{A} \text{ queries } \boldsymbol{x}_i] .$$

Note that the expected number of queries to compute f via query algorithm $\mathcal{A}$ can be written in terms of the $\delta_i(\mathcal{A})$'s as follows:

$$\sum_{i=1}^{n} \delta_i(\mathcal{A}).$$

Our algorithms in Section 4 will rely on the well-known "OSSS inequality" from the analysis of Boolean functions [OSSS05] (see also Chapter 8 of [O'D14]). The following variant of the OSSS inequality is due to Jain and Zhang [JZ11]:

Theorem 10 (OSSS inequality, [JZ11] version)**.** *For all functions $f : \{0,1\}^n \to \{0,1\}$ and query algorithms $\mathcal{A}$, we have*

$$\mathrm{bias}(f) - \mathrm{error}_f(\mathcal{A}) \leq \sum_{i=1}^{n} \delta_i(\mathcal{A}) \cdot \mathbf{Inf}_i[f].$$

Note that Theorem 10 is a refinement of the classical edge-isoperimetric or Poincaré inequality (see Chapter 2 of [O'D14]) over the Boolean hypercube that takes the query complexity (or alternatively, decision-tree structure) of f into account. This inequality underlies the analysis of both the offline strategy of [BLT21] as well as our online algorithms in Section 4.

3.2 Problem Setup

We begin by recalling the setup of Blanc, Lange, and Tan [BLT21], which is itself a variant of the problem formulation considered by Deshpande, Hellerstein, and Kletenik [DHK14]. In this setting, the algorithm is given a function $f : \{0,1\}^n \to \{0,1\}$ and a cost vector $c \in \mathbb{R}^n_{\geq 0}$. Its objective is to design a query algorithm to compute f on an unknown (but fixed) input $x \in \{0,1\}^n$, where it incurs a cost of c_i to reveal the i^{th} bit of x. We will sometimes refer to this setup, where the cost vector c is known to the algorithms, as the "offline" setting.

In our setting, the key difference is that c_i is *unknown* to the algorithm. To distinguish this setup from the previous one, we will refer to it as the "online" setting.

Definition 11 (Online Priced Query Model)**.** The algorithm is given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, while the input $x \in \{0,1\}^n$ and the cost vector $c \in \mathbb{R}^n_{\geq 0}$ are unknown. The algorithm maintains a *cumulative investment vector* $\theta \in \mathbb{R}^n_{\geq 0}$, where θ_i represents the algorithm's total investment towards revealing the variable x_i. At each step, the algorithm selects a coordinate $i \in [n]$ to invest in, incrementing θ_i by β (which can be viewed as the algorithm's minimal budget or resolution). A bit x_i is revealed once the investment θ_i reaches or exceeds the corresponding cost c_i. The *total cost* incurred by the algorithm is defined as the value of $\|\theta\|_1$ when it halts.

Remark 12. Throughout, we assume that our algorithms are given an explicit representation of the function $f : \{0,1\}^n \to \{0,1\}$. They can be modified in a natural fashion to work with black-box or query access to f. For example, the WARMUP-IPRR and IPRR algorithms in Section 4 require knowledge of coordinate influences, which can be estimated via random sampling; the failure probability can then be controlled by a union bound.

Remark 13. Our upper bounds will exhibit a dependence on a "unit investment" or "resolution" parameter β, which represents the algorithm's unit investment step size. However, note that β is a free parameter that can be chosen to be arbitrarily small.

The following notation will be helpful:

Definition 14. Let $\mathcal{A}$ be an ε-error (offline or online) algorithm for computing $f : \{0,1\}^n \to \{0,1\}$, and let $c \in \mathbb{R}^n_{\geq 0}$ be an associated cost vector. We will write $\text{cost}_c^f(\mathcal{A}, x)$ for the total cost incurred by $\mathcal{A}$ on input $x \in \{0,1\}^n$. We define

$$\text{avg-cost}_c^f(\mathcal{A}) := \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n} \left[\text{cost}_c(\mathcal{A}, \boldsymbol{x})\right] \qquad \text{and} \qquad \text{worst-cost}_c^f(\mathcal{A}) := \max_{x \in \{0,1\}^n} \text{cost}_c(\mathcal{A}, x)$$

to be the *expected cost* and *worst-case cost* of $\mathcal{A}$ respectively. When the function f is clear from context, we will omit dependence on it (writing, for e.g., $\text{avg-cost}_c(\mathcal{A})$ instead of $\text{avg-cost}_c^f(S)$) for notational simplicity.

We will benchmark the performance of our algorithms against the cost of optimal query algorithms, which may be either offline or online:

Definition 15. Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, a cost vector $c \in \mathbb{R}^n_{\geq 0}$, and an error parameter $\varepsilon \in [0, 0.5]$, we define

$$\text{opt}_\varepsilon^{\text{avg}}(f, c) := \inf \text{avg-cost}_c^f(\mathcal{A}) \qquad \text{and} \qquad \text{opt}_\varepsilon^{\text{w}}(f, c) := \inf \text{worst-cost}_c^f(\mathcal{A}),$$

where both infima are taken over all ε-error query algorithms $\mathcal{A}$ (either offline or online) for f. We will frequently write $\text{opt}_\varepsilon^{\text{avg}} = \text{opt}_\varepsilon^{\text{avg}}(f, c)$ and $\text{opt}_\varepsilon^{\text{w}} = \text{opt}_\varepsilon^{\text{w}}(f, c)$ for simplicity whenever f and c are clear from context.

It is readily verified that

$$\text{opt}_\varepsilon^{\text{avg}} \leq \text{opt}_0^{\text{avg}} \leq \text{opt}_0^{\text{w}} \qquad \text{and} \qquad \text{opt}_\varepsilon^{\text{avg}} \leq \text{opt}_\varepsilon^{\text{w}} \leq \text{opt}_0^{\text{w}}. \tag{3}$$

Thus, benchmarking against $\text{opt}_\varepsilon^{\text{avg}}$ is the strongest guarantee we can hope for.

4 The Influence-Proportional Round Robin Algorithm

The main result of this section is an $O(\varepsilon)$-error online algorithm that is competitive with the optimal ε-error *offline* algorithm, up to an additional factor of $\mathbf{I}[f]$ and some logarithmic terms:

Theorem 16. *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and $c \in \mathbb{R}_{\geq 0}$ be an unknown cost vector. For $\varepsilon \in (0, 0.5]$ and $\beta > 0$, there exists an online algorithm with unit investment β,* Online-Query *(Algorithm 4) which:*

- *Computes f to $O(\varepsilon)$-error; and*
- *Assuming $\text{opt}_\varepsilon^{\text{avg}} = \Omega(1)$, satisfies*

$$\text{avg-cost}_c(\textsc{Online-Query}) \leq \beta n + \text{opt}_\varepsilon^{\text{avg}} \cdot O\left(\frac{1}{\varepsilon^3} \log \frac{\log \text{opt}_\varepsilon^{\text{avg}}}{\varepsilon}\right) \cdot \sum_{i=1}^n \mathbf{Inf}_i[f] \left(1 + \ln \frac{1}{\mathbf{Inf}_i[f]}\right). \tag{4}$$

Remark 17. Since the function $x \mapsto x(1 + \ln x^{-1})$ is concave, it follows from Equation (4) and Jensen's inequality that

$$\text{avg-cost}_c(\textsc{Online-Query}) \leq \beta n + \text{opt}_\varepsilon^{\text{avg}} \cdot O\left(\frac{\mathbf{I}[f] \log n}{\varepsilon^3} \cdot \log \frac{\log \text{opt}_\varepsilon^{\text{avg}}}{\varepsilon}\right).$$

4.1 Warmup

Before proving Theorem 16, we first establish a weaker result which illustrates the key ideas behind the proof of Theorem 16. In particular, some parts of the proof of Theorem 16 will rely on calculations from the proof of Theorem 18 below.

Theorem 18 (Formal version of Theorem 7)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and $c \in \mathbb{R}^n_{\geq 0}$ be a cost vector. For $\varepsilon \in (0, 0.5]$ and $\beta > 0$, there exists an ε-error online algorithm* Warmup-IPRR *(Algorithm 2) with unit investment β such that*

$$\text{avg-cost}_c(\textsc{Warmup-IPRR}) \leq \beta n + \frac{\text{opt}_0^{\text{w}}}{\varepsilon} \cdot \sum_{i=1}^{n} \mathbf{Inf}_i[f] \left(1 + \ln \frac{1}{\mathbf{Inf}_i[f]}\right). \tag{5}$$

Theorem 18 gives an ε-error algorithm whose expected cost is upper bounded by the (offline) *zero-error* algorithm with minimal worst-case cost—a guarantee that is the weakest among those we consider (cf. Equation (3)). (In contrast, Theorem 16 is competitive against the optimal ε-error algorithm with minimal average-case cost, which is the strongest guarantee we consider.)

Input: Succinct representation of $f : \{0,1\}^n \to \{0,1\}$, error parameter $\varepsilon \in (0, 0.5]$
Output: A bit $b \in \{0,1\}$

Warmup-IPRR(f, ε):

1. Initialize $\theta \leftarrow 0^n$ and $\pi \leftarrow \emptyset$.
2. While $\text{bias}(f_\pi) > \varepsilon$:
   (a) Let $i^* \in [n]$ be the index such that
   $$i^* = \arg\max_i \frac{\mathbf{Inf}_i[f_\pi]}{\theta_i}.$$
   (b) Spend cost β towards x_{i^*} and update $\theta \leftarrow \theta + \beta e_{i^*}$.
   (c) If x_{i^*} is revealed to be $b_{i^*} \in \{0,1\}$, then update $\pi \leftarrow \pi \cup \{x_{i^*} \mapsto b_{i^*}\}$.
3. Output $\mathbf{1}\left\{ \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n} \left[f_\pi(\boldsymbol{x})\right] \geq 1/2 \right\}$.

Algorithm 2: The Warmup-IPRR Algorithm

Our proof of Theorem 18 will rely on the following consequence of the well-known OSSS inequality (Theorem 10) obtained by Blanc, Lange, and Tan [BLT21]:

Lemma 19 (Lemma 3.1 of [BLT21])**.** *For all Boolean functions $f : \{0,1\}^n \to \{0,1\}$, cost vectors $c \in \mathbb{R}^n_{\geq 0}$, and query algorithms $\mathcal{A}$, we have*

$$\max_{i \in [n]} \frac{\mathbf{Inf}_i[f]}{c_i} \geq \frac{\text{bias}(f) - \text{error}_f(\mathcal{A})}{\text{avg-cost}^f_c(\mathcal{A})}.$$

We will also require Doob's martingale inequality; recall that a discrete-time stochastic process $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_T)$ is a *martingale* if for all $t \in [T]$, we have

$$\mathbf{E}\left[|\boldsymbol{X}_t|\right] < \infty \qquad \text{and} \qquad \mathbf{E}\left[\boldsymbol{X}_{t+1} \mid \boldsymbol{X}_1, \ldots, \boldsymbol{X}_t\right] = \boldsymbol{X}_t\,.$$

Lemma 20 (Doob's inequality, Section 14.6 of [Wil91])**.** *Given a martingale $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_T)$, for all $C > 0$ we have*

$$\mathbf{Pr}\left[\max_{t\in[T]} \boldsymbol{X}_t \geq C\right] \leq \frac{\mathbf{E}\left[\max\left\{\boldsymbol{X}_T, 0\right\}\right]}{C}\,.$$

We are now ready to prove Theorem 18:

Proof of Theorem 18. Consider the execution of the algorithm WARMUP-IPRR on a fixed input $x \in \{0,1\}^n$ with unknown cost vector c. Note that x induces a sequence of restrictions π corresponding to the variables revealed by WARMUP-IPRR; call this collection of restrictions $\Pi(x)$. For each index $i \in [n]$, let $\pi^{(i)}$ denote the restriction π *right before* θ_i (which is the investment in variable x_i) gets incremented for the last time. In the rest of the proof, vector θ denotes the investments in the n variables *at this moment* (before θ_i gets incremented for the last time). In contrast, the *final* investment in variable x_i will be denoted by θ_i^*, which is equal to $\theta_i + \beta$.

Let $\mathcal{A}^\circ$ denote the zero-error offline algorithm for f with minimal worst-case cost, i.e.,[4]

$$\mathrm{error}_f(\mathcal{A}^\circ) = 0 \qquad \text{and} \qquad \text{worst-cost}_c^f(\mathcal{A}^\circ) = \mathrm{opt}_0^{\mathrm{w}}(f, c)\,.$$

It follows from Step 2(a) of Algorithm 2 that

$$\frac{\mathbf{Inf}_i[f_{\pi^{(i)}}]}{\theta_i^* - \beta} = \frac{\mathbf{Inf}_i[f_{\pi^{(i)}}]}{\theta_i} = \max_j \frac{\mathbf{Inf}_j[f_{\pi^{(i)}}]}{\theta_j} \geq \max \frac{\mathbf{Inf}_j[f_{\pi^{(i)}}]}{c_j}\,.$$

Let $\mathcal{A}^\circ_{\pi^{(i)}}$ denote the algorithm obtained from $\mathcal{A}^\circ$ by enforcing the restriction $\pi^{(i)}$. In other words, $\mathcal{A}^\circ_{\pi^{(i)}}$ follows $\mathcal{A}^\circ$ and, whenever a variable x_j is queried by $\mathcal{A}^\circ$ while $x_j \mapsto b_j$ is among restriction $\pi^{(i)}$, $\mathcal{A}^\circ_{\pi^{(i)}}$ proceeds by forwarding $x_j = b_j$ to $\mathcal{A}^\circ$. By construction, we have

$$\mathrm{error}_{f_{\pi^{(i)}}}(\mathcal{A}^\circ_{\pi^{(i)}}) = 0 \qquad \text{and} \qquad \text{worst-cost}_c^{f_{\pi^{(i)}}}(\mathcal{A}^\circ_{\pi^{(i)}}) \leq \text{worst-cost}_c^f(\mathcal{A}^\circ) = \mathrm{opt}_0^{\mathrm{w}}(f, c).$$

Then, applying Lemma 19 to $f_{\pi^{(i)}}$ and $\mathcal{A}^\circ_{\pi^{(i)}}$ gives

$$\frac{\mathbf{Inf}_i[f_{\pi^{(i)}}]}{\theta_i^* - \beta} \geq \max \frac{\mathbf{Inf}_j[f_{\pi^{(i)}}]}{c_j} \geq \frac{\mathrm{bias}(f_{\pi^{(i)}})}{\text{avg-cost}_c^{f_{\pi^{(i)}}}(\mathcal{A}^\circ_{\pi^{(i)}})} \geq \frac{\varepsilon}{\mathrm{opt}_0^{\mathrm{w}}}\,, \tag{6}$$

where the last inequality relies on:

- The observation

$$\text{avg-cost}_c^{f_{\pi^{(i)}}}(\mathcal{A}^\circ_{\pi^{(i)}}) \leq \text{worst-cost}_c^{f_{\pi^{(i)}}}(\mathcal{A}^\circ_{\pi^{(i)}}) \leq \mathrm{opt}_0^{\mathrm{w}}(f, c)\,;$$

  as well as

- The fact that $\mathrm{bias}(f_{\pi^{(i)}}) > \varepsilon$ by design of Algorithm 2.

[4] Technically, $\mathrm{opt}_0^{\mathrm{w}}$ is defined as an infimum and might not be obtained by any algorithm, though the rest of the proof would still go through by considering a sequence of zero-error algorithms with worst-case costs approaching $\mathrm{opt}_0^{\mathrm{w}}$.

We can rewrite Equation (6) as follows:

$$\theta_i^* \le \beta + \frac{\mathrm{opt}_0^{\mathrm{w}}}{\varepsilon} \cdot \mathbf{Inf}_i[f_{\pi^{(i)}}] \le \beta + \frac{\mathrm{opt}_0^{\mathrm{w}}}{\varepsilon} \cdot \max_{\pi \in \Pi(x)} \mathbf{Inf}_i[f_\pi]\,. \tag{7}$$

Recall that $\Pi(x)$ is the collection of all restrictions encountered by WARMUP-IPRR on input $x \in \{0,1\}^n$. The second step above follows from the observation that $\pi^{(i)} \in \Pi(x)$.

Note that because $\mathbf{Inf}_i[f_{\pi^{(i)}}] \in [0,1]$, Equation (7) immediately implies that

$$\text{worst-cost}_c(\textsc{Warmup-IPRR}) \le \beta n + \frac{\mathrm{opt}_0^{\mathrm{w}}}{\varepsilon} n\,.$$

We will obtain our improved bound by considering the expected cost of WARMUP-IPRR. Indeed, it follows from Equation (7) and the definition of $\text{avg-cost}_c(\cdot)$ that

$$\text{avg-cost}_c(\textsc{Warmup-IPRR}) \le \beta n + \frac{\mathrm{opt}_0^{\mathrm{w}}}{\varepsilon} \sum_{i=1}^{n} \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n} \left[\max_{\pi \in \Pi(\boldsymbol{x})} \mathbf{Inf}_i[f_\pi] \right].$$

So in order to complete the proof, it suffices to show that

$$\mathbf{E}\left[\max_{\pi \in \Pi(\boldsymbol{x})} \mathbf{Inf}_i[f_\pi] \right] \le \mathbf{Inf}_i[f] \cdot \left(1 + \ln \frac{1}{\mathbf{Inf}_i[f]}\right). \tag{8}$$

The rest of the argument will establish Equation (8).

For each $t \in \{0,1,2,\ldots,n\}$, let $\boldsymbol{\pi}_t$ denote the restriction encountered by WARMUP-IPRR when it runs on a uniformly random input $\boldsymbol{x} \sim \{0,1\}^n$ and has revealed the value of exactly t variables. If the algorithm reveals $t' < t$ variables before halting, we define $\boldsymbol{\pi}_t = \boldsymbol{\pi}_{t'}$ instead. Then, define the random variable $\boldsymbol{X}_t^{(i)}$ as follows:

- If $\boldsymbol{x}_i$ is not among the first t revealed variables, then $\boldsymbol{X}_t^{(i)} = \mathbf{Inf}_i[f_{\boldsymbol{\pi}_t}]$.
- If $\boldsymbol{x}_i$ is the t^*-th revealed variable for some $t^* \le t$, then $\boldsymbol{X}_t^{(i)} = \boldsymbol{X}_{t^*-1}^{(i)}$. In other words, the value of $\boldsymbol{X}_t^{(i)}$ is frozen to its last value before $\boldsymbol{x}_i$ is revealed.

It is readily verified that $(\boldsymbol{X}_0^{(i)}, \boldsymbol{X}_1^{(i)}, \ldots, \boldsymbol{X}_n^{(i)})$ forms a martingale with the following properties:

- $\boldsymbol{X}_0^{(i)} = \mathbf{Inf}_i[f]$ with probability 1.
- $\boldsymbol{X}_t^{(i)} \in [0,1]$ for all $t \in \{0,\ldots,n\}$.

Furthermore, we note that $\Pi(x) = \{\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_n\}$, so it holds that $\max_{\pi \in \Pi(x)} \mathbf{Inf}_i[f_\pi] = \max_{0 \le t \le n} \boldsymbol{X}_t^{(i)}$.

Let r^* be a parameter that we will set shortly. We then have

$$\begin{aligned}
\text{L.H.S. of (8)} = \mathbf{E}\left[\max_{0\le t\le n} \boldsymbol{X}_t^{(i)}\right] &= \int_{r=0}^{\infty} \mathbf{Pr}\left[\max_{0\le t\le n} \boldsymbol{X}_t^{(i)} \ge r\right] \mathrm{d}r && (9)\\
&= \int_{r=0}^{1} \mathbf{Pr}\left[\max_{0\le t\le n} \boldsymbol{X}_t^{(i)} \ge r\right] \mathrm{d}r && (\text{Since } \boldsymbol{X}_t^{(i)} \in [0,1] \text{ for all } t)\\
&= \int_{r=0}^{r^*} \mathbf{Pr}\left[\max_{0\le t\le n} \boldsymbol{X}_t^{(i)} \ge r\right] \mathrm{d}r + \int_{r=r^*}^{1} \mathbf{Pr}\left[\max_{0\le t\le n} \boldsymbol{X}_t^{(i)} \ge r\right] \mathrm{d}r\\
&\le r^* + \int_{r=r^*}^{1} \mathbf{Pr}\left[\max_{0\le t\le n} \boldsymbol{X}_t^{(i)} \ge r\right] \mathrm{d}r\\
&\le r^* + \int_{r=r^*}^{1} \frac{\mathbf{E}\left[\boldsymbol{X}_n^{(i)}\right]}{r} \mathrm{d}r && (\text{Doob's inequality, Lemma 20})\\
&= r^* + \int_{r=r^*}^{1} \frac{\mathbf{Inf}_i[f]}{r} \mathrm{d}r && (\text{Since } \boldsymbol{X}_t^{(i)} \text{ is a martingale})\\
&= r^* + \mathbf{Inf}_i[f] \cdot \ln\left(\frac{1}{r^*}\right). && (10)
\end{aligned}$$

The final quantity above is minimized for $r^* = \mathbf{Inf}_i[f]$, and so

$$\mathbf{E}\left[\max_{\pi\in\Pi(\boldsymbol{x})} \mathbf{Inf}_i[f_\pi]\right] \le \mathbf{Inf}_i[f] \cdot \left(1 + \ln \frac{1}{\mathbf{Inf}_i[f]}\right),$$

establishing Equation (8) and in turn completing the proof. □

4.2 A Hard Instance for Warmup-IPRR

In this section, we will consider an instance of the online query problem which shows that the analysis of Algorithm 2 from the previous section is tight. In particular, this "hard instance" suggests a natural modification to Algorithm 2 which will allow us to establish Theorem 16. Throughout this section, we will suppress dependence on the resolution or unit-cost parameter β and will assume that β is n^{-c} for some sufficiently large constant c. This is without loss of generality since for any choice of β, the cost vector c can be scaled appropriately.

Proposition 21 (Formal version of Proposition 8)**.** *For every integer $k \ge 1$ and $n = 2^{2k} + 2k$, there is an n-variable function h and a cost vector $c \in \mathbb{R}^n_{\ge 0}$ such that, for any accuracy parameter $\varepsilon \in (0, 1/4)$,* Warmup-IPRR(h, ε) *gives an expected cost of*

$$\text{avg-cost}_c^h(\textsc{Warmup-IPRR}) = \text{opt}_0^{\text{avg}}(h, c) \cdot \Omega\left(\mathbf{I}[f] \cdot \frac{\sqrt{n}}{\log n}\right).$$

We will require a slight modification of the well-known "address function" from Boolean function complexity:

Definition 22. For $n = 2^{2k}$ for some $k \in \mathbb{N}$, consider the function $f : \{0,1\}^{\log\sqrt{n}} \times \{0,1\}^n \to \{0,1\}$ defined as

$$f(x, y) := \bigoplus_{i=1}^{\sqrt{n}} y_{x,i}$$

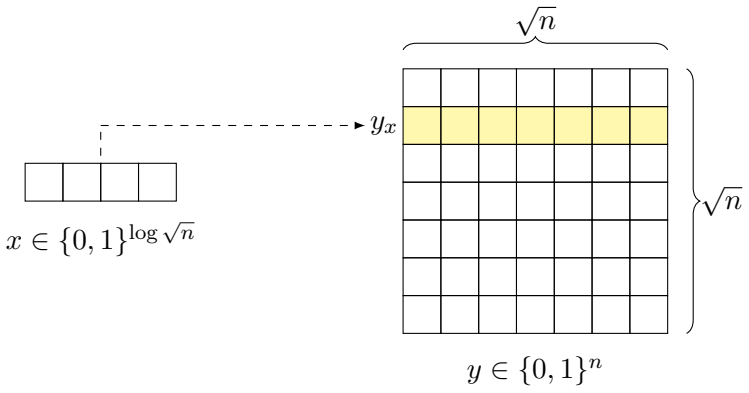

Figure 1: The function $f : \{0,1\}^{\log \sqrt{n}} \times \{0,1\}^n \to \{0,1\}$ from Definition 22.

where we identify $x \in \{0,1\}^{\log \sqrt{n}}$ with the binary representation of an index $j \in [\sqrt{n}]$ and view $y \in \{0,1\}^n$ as a $\sqrt{n} \times \sqrt{n}$ Boolean matrix. (Here, $x \oplus y = (x + y) \bmod 2$ is the XOR operation.)

It will be convenient to refer to the variables of $x \in \{0,1\}^{\log \sqrt{n}}$ as *control variables*, and to the variables of $y \in \{0,1\}^n$ as *action variables*. (See Figure 1 for an illustration of Definition 22.) It is readily verified that

$$\mathbf{Inf}_i[f] = \begin{cases} 0.5 & i \text{ indexes a control variable} \\ \frac{1}{\sqrt{n}} & i \text{ indexes an action variable} \end{cases}$$

and consequently $\mathbf{I}[f] = \Theta(\sqrt{n})$.

Consider a cost vector $c \in \mathbb{R}^{\log \sqrt{n}+n}$ where

$$c_i = \Theta_\beta\big(\mathbf{Inf}_i[f]\big)\,.$$

It is readily seen that $\mathrm{opt}_0^{\mathrm{avg}}(f,c) = \Theta_\beta(\log n)$ by considering the algorithm which queries the control variables at cost $\Theta_\beta(\log n)$ to obtain $x \in \{0,1\}^{\log \sqrt{n}}$ and then computes $\bigoplus_i y_{x,i}$ at cost $\Theta_\beta(1)$. On the other hand, since Warmup-IPRR (Algorithm 2) invests in proportion to the coordinate influences (which are all equal for the action variables), we have that

$$\text{avg-cost}_c(\textsc{Warmup-IPRR}) = \Theta_\beta(\sqrt{n})$$

since each of the n action variables has cost $\Theta_\beta(n^{-1/2})$. In particular, this example establishes that

$$\text{avg-cost}_c(\textsc{Warmup-IPRR}) = \Theta_\beta(\sqrt{n}) \geq \Omega\left(\mathrm{opt}_0^{\mathrm{avg}} \cdot \frac{\mathbf{I}[f]}{\log n}\right)\,. \tag{11}$$

Next, we show how to bootstrap this example to give a query instance of a Boolean function $h : \{0,1\}^n \to \{0,1\}$ where

$$\text{avg-cost}_c(\textsc{Warmup-IPRR}) = \mathrm{opt}_0^{\mathrm{avg}}(h,c) \cdot \Omega\left(\mathbf{I}[h] \cdot \frac{\sqrt{n}}{\log n}\right)\,. \tag{12}$$

We first describe the instance which establishes Equation (12). Let $\ell \geq 1$ be a parameter to be set shortly, and let $g : \{0,1\}^\ell \to \{0,1\}$ be a *decision list* [Riv87] (or equivalently, rank-1 decision tree) such that: (i) the decision list computing $g(z)$ queries variables $z_1, z_2, \ldots, z_\ell$ in order; (ii) the labels

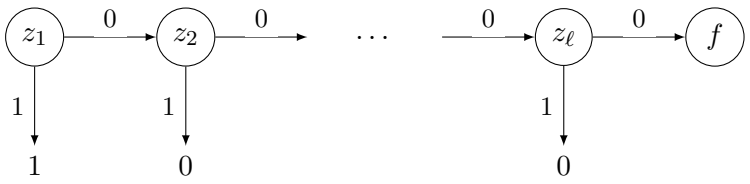

Figure 2: The function $h : \{0,1\}^\ell \times \{0,1\}^{\log\sqrt{n}} \times \{0,1\}^n \to \{0,1\}$ where f is as in Definition 22.

of the decision list's rules alternate (i.e., the first rule is "If $z_1 = 1$, output 1; else if $z_2 = 1$, output 0; ..."); (iii) the final rule is "If $z_\ell = 1$, output $\ell \bmod 2$; else, output 1."

Then, we define the function

$$h : \{0,1\}^\ell \times \{0,1\}^{\log\sqrt{n}} \times \{0,1\}^n \to \{0,1\}$$

by modifying g as described in Figure 2: $h(z,x,y)$ agrees with $g(z)$ as long as $z \neq 0^\ell$; when $z = 0^\ell$, $h(z,x,y)$ is defined as $f(x,y)$ from Definition 22 instead. We will refer to the variables in $\{0,1\}^\ell$ as the *list variables*.

We thus have

$$\mathbf{Inf}_i[h] = \begin{cases} \Theta(2^{-i}) & i \text{ is a list variable,} \\ 2^{-(\ell+1)} & i \text{ is a control variable,} \\ \frac{2^{-\ell}}{\sqrt{n}} & i \text{ is an action variable.} \end{cases}$$

In particular, taking $\ell := \log\sqrt{n}$, we get

$$\mathbf{I}[h] = \left(\sum_{i=1}^{\ell} \Theta(2^{-i})\right) + 2^{-\ell} \cdot \mathbf{I}[f] = \Theta(1)\,. \tag{13}$$

Furthermore, the alternating labels in $g(z)$ ensures that

$$\mathop{\mathbf{E}}_{(\boldsymbol{z},\boldsymbol{x},\boldsymbol{y})\sim\{0,1\}^{\ell+\log\sqrt{n}+n}} \left[h_\pi(\boldsymbol{z},\boldsymbol{x},\boldsymbol{y})\right] \in \left[\frac{1}{4}, \frac{3}{4}\right] \tag{14}$$

holds for any restriction π of form $\{z_1 \mapsto 0, z_2 \mapsto 0, \ldots, z_{\ell'} \mapsto 0\}$ where $\ell' \in \{0,1,2,\ldots,\ell\}$.

Next, consider an instance of the query problem where the costs of the list variables are 0, the control variables are $\Theta_\beta(1)$, and the cost of each action variable is $\Theta_\beta(n^{-1/2})$. It follows that

$$\mathrm{opt}_0^{\mathrm{avg}}(h,c) = 2^{-\ell} \cdot \mathrm{opt}_0^{\mathrm{avg}}(f,c)\,,$$

and similarly

$$\text{avg-cost}_c^h(\textsc{Warmup-IPRR}) = 2^{-\ell} \cdot \text{avg-cost}_c^f(\textsc{Warmup-IPRR})\,.$$

In particular, by Equation (14) and the assumption that $\varepsilon < 1/4$ in Proposition 21, $\textsc{Warmup-IPRR}(h,\varepsilon)$ must query all of the ℓ list variables when they are all zeros. It then follows from Equations (11) and (13) that

$$\text{avg-cost}_c^h(\textsc{Warmup-IPRR}) \geq \mathrm{opt}_0^{\mathrm{avg}}(h) \cdot \Omega\left(\mathbf{I}[h] \cdot \frac{\sqrt{n}}{\log n}\right),$$

establishing Equation (12).

4.3 Proof of Theorem 16

It follows from Proposition 21 that we cannot hope for WARMUP-IPRR to be competitive against $\mathrm{opt}_0^{\mathrm{avg}}$ without losing a $\mathrm{poly}(n)$-factor in the competitive ratio. In this section, we describe a modification of WARMUP-IPRR which gives an algorithm that is competitive against $\mathrm{opt}_\varepsilon^{\mathrm{avg}}$, establishing Theorem 16. We will in fact obtain Theorem 16 as an immediate consequence of the following (seemingly) weaker guarantee:

Proposition 23. *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, $c \in \mathbb{R}^n_{\geq 0}$ be a cost vector, and $\varepsilon \in (0, 0.5)$ be an error parameter. There is an online algorithm* ONLINE-QUERY *(Algorithm 4) such that:*

- ONLINE-QUERY *computes f to error $O(\varepsilon)$, and*
- *Assuming $\mathrm{opt}_\varepsilon^{\mathrm{w}} = \Omega(1)$, the expected cost is at most*

$$\beta n + \mathrm{opt}_\varepsilon^{\mathrm{w}} \cdot O\left(\frac{1}{\varepsilon^2} \cdot \log\left(\frac{\log \mathrm{opt}_\varepsilon^{\mathrm{w}}}{\varepsilon}\right)\right) \cdot \sum_{i=1}^{n} \mathbf{Inf}_i[f]\left(1 + \ln \frac{1}{\mathbf{Inf}_i[f]}\right).$$

Theorem 16 follows via an application of Markov's inequality; this observation was also made by Blanc, Lange, and Tan [BLT21].

Proof of Theorem 16 from Proposition 23. It suffices to show that

$$\mathrm{opt}_{2\varepsilon}^{\mathrm{w}}(f, c) \leq \frac{\mathrm{opt}_\varepsilon^{\mathrm{avg}}(f, c)}{\varepsilon}. \tag{15}$$

Indeed, Theorem 16 follows immediately from Proposition 23 and Equation (15).

Let $\mathcal{A}$ be the ε-error strategy with expected cost $\mathrm{opt}_\varepsilon^{\mathrm{avg}}$. Consider the strategy $\mathcal{A}'$ which executes $\mathcal{A}$, except that it outputs 1 if it makes a query leading to cost greater than $\frac{\mathrm{opt}_\varepsilon^{\mathrm{avg}}}{\varepsilon}$. By Markov's inequality, note that $\mathcal{A}$ differs from $\mathcal{A}'$ with probability at most ε and consequently is a 2ε-error strategy for f. Furthermore, note that it has worst-case cost at most $\mathrm{opt}_\varepsilon^{\mathrm{avg}}/\varepsilon$, and so Equation (15) holds. □

The remainder of the section will establish Proposition 23. Our modification of WARMUP-IPRR is given by the algorithm IPRR (Algorithm 4). The algorithm IPRR differs from WARMUP-IPRR in two ways:

- First, the algorithm takes in an additional input in the form of the threshold $B > 0$. We can view B as a proxy for the "total budget" of the algorithm. (Looking ahead, our full algorithm ONLINE-QUERY (Algorithm 4) will use a doubling strategy to guess good choices of B until $B = \mathrm{opt}_\varepsilon^{\mathrm{w}}$.)
- Second, the algorithm uses a different exit condition (Step 2(b) above). Recall that the algorithm WARMUP-IPRR terminated when the bias of the restricted function was sufficiently small.

The following proposition guarantees that IPRR has small error for the right choice of threshold B:

Proposition 24. *Let $f : \{0,1\}^n \to \{0,1\}$ and c be an unknown cost vector. When $B \geq \mathrm{opt}_\varepsilon^{\mathrm{w}}$, the algorithm $\mathrm{IPRR} = \mathrm{IPRR}(f, \varepsilon, B)$ is a 2ε-error algorithm for f.*

Input: Succinct representation of $f : \{0,1\}^n \to \{0,1\}$, threshold $B > 0$, $\varepsilon \in (0, 0.5]$
Output: A bit $b \in \{0,1\}$

IPRR(f, ε, B):

1. Initialize $\theta \leftarrow 0^n$ and $\pi \leftarrow \emptyset$.
2. Repeat:
   (a) Let $i^* \in [n]$ be the index such that
   $$i^* = \arg\max_i \frac{\mathbf{Inf}_i[f_\pi]}{\theta_i}.$$
   (b) If $\frac{\mathbf{Inf}_{i^*}[f_\pi]}{\theta_{i^*}} < \frac{\varepsilon}{B}$, then halt and output $\mathbf{1}\left\{\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_\pi(\boldsymbol{x})\right] \geq 1/2\right\}$.
   (c) Else:
   - Spend cost β towards x_{i^*} and update $\theta \leftarrow \theta + \beta e_{i^*}$.
   - If x_{i^*} is revealed to be $b_{i^*} \in \{0,1\}$, then update $\pi \leftarrow \pi \cup \{x_{i^*} \mapsto b_{i^*}\}$.

Algorithm 3: The IPRR Algorithm

Proof. Let T be the ε-error algorithm witnessing $\mathrm{opt}^{\mathrm{w}}_\varepsilon$, i.e.

$$\mathrm{error}_f(T) \leq \varepsilon \quad \text{and} \quad \text{worst-cost}_c(T) = \mathrm{opt}^{\mathrm{w}}_\varepsilon(f, c).$$

As in the proof of Theorem 18, for any restriction π, we let T_π denote the algorithm obtained from T by enforcing the restriction π: T_π follows T and, whenever T queries a variable x_i such that "$x_i \mapsto b_i$" is in the restriction π, T_π simply forwards b_i to T as the value of x_i and proceeds. By construction, we have

$$\text{avg-cost}^{f_\pi}_c(T_\pi) \leq \text{worst-cost}^{f_\pi}_c(T_\pi) \leq \text{worst-cost}^{f}_c(T).$$

Applying Lemma 19 to f_π and T_π, we have the invariant that for any restriction π during the execution of Algorithm 3,

$$\max_i \frac{\mathbf{Inf}_i[f_\pi]}{c_i} \geq \frac{\mathrm{bias}(f_\pi) - \mathrm{dist}(T_\pi, f_\pi)}{\text{avg-cost}^{f_\pi}_c(T_\pi)}. \tag{16}$$

Note that Algorithm 3 only halts in Step 2(b) when

$$\frac{\varepsilon}{B} \geq \max_i \frac{\mathbf{Inf}_i[f_\pi]}{c_i}.$$

Since $B \geq \mathrm{opt}^{\mathrm{w}}_\varepsilon(f, c) = \text{worst-cost}^{f}_c(T)$, it follows that when the algorithm halts, the restriction π satisfies

$$\frac{\varepsilon}{\text{worst-cost}^{f}_c(T)} \geq \frac{\varepsilon}{B} \geq \frac{\mathrm{bias}(f_\pi) - \mathrm{dist}(T_\pi, f_\pi)}{\text{avg-cost}^{f_\pi}_c(T_\pi)} \geq \frac{\mathrm{bias}(f_\pi) - \mathrm{dist}(T_\pi, f_\pi)}{\text{worst-cost}^{f}_c(T)},$$

Input: Succinct representation of $f : \{0,1\}^n \to \{0,1\}$, $\varepsilon \in (0, 0.5]$
Output: A bit $b \in \{0,1\}$

ONLINE-QUERY(f, ε):

1. Initialize $i \leftarrow 0$.
2. Repeat:
   (a) Increment $i \leftarrow i + 1$. Set $B_i := 2^i$ and $m_i := \Theta\left(\frac{1}{\varepsilon} \log \frac{i}{\varepsilon}\right)$.
   (b) Draw $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m_i)} \sim \{0,1\}^n$ and let $\boldsymbol{b}^{(\ell)} \leftarrow \text{IPRR}(f, \varepsilon, B_i)$ by simulating IPRR on $\boldsymbol{x}^{(\ell)}$ as the unknown input.
   (c) Compute
   $$\boldsymbol{T}_i := \frac{1}{m_i} \sum_{\ell=1}^{m_i} \mathbf{1}\left\{\boldsymbol{b}^{(\ell)} \neq f(\boldsymbol{x}^{(\ell)})\right\},$$
   and if $\boldsymbol{T}_i \leq 3\varepsilon$, run $\text{IPRR}(f, \varepsilon, B_i)$ on the unknown input x.

Algorithm 4: The ONLINE-QUERY Algorithm

which can be rearranged to $\text{bias}(f_\pi) \leq \varepsilon + \text{dist}(T_\pi, f_\pi)$. As in the proof of Theorem 18, we let $\boldsymbol{\pi}$ be the restriction induced by the algorithm IPRR on a uniformly random input $\boldsymbol{x} \sim \{0,1\}^n$. Taking expectations, we get that

$$\text{error}_f(\text{IPRR}) = \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n}\left[\text{bias}(f_{\boldsymbol{\pi}})\right] \leq \varepsilon + \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n}\left[\text{dist}(T_{\boldsymbol{\pi}}, f_{\boldsymbol{\pi}})\right] \leq 2\varepsilon\,,$$

where the final inequality relies on the fact that T is an ε-error algorithm for f. □

Proposition 24 suggests a natural strategy: double our guess for B until we hit $\text{opt}^{\text{w}}_\varepsilon$. Note, however, that we do not know if our current guess is correct. We remedy this by *testing* if our budget B gives an empirical error of $O(\varepsilon)$; this in done in Step 2(b) of Algorithm 4.

Remark 25. We provide more details on how ONLINE-QUERY (Algorithm 4) simulates the IPRR algorithm in Step 2(b). For clarity, let θ^{outer} (resp., θ^{inner}) denote the cumulative investment vector maintained by ONLINE-QUERY (resp., a simulation of IPRR). Also, let $\boldsymbol{x}^{(\ell)}$ denote the input for the simulation, and let $\boldsymbol{x}$ denote the actual unknown input on which ONLINE-QUERY evaluates f. Recall that, in the simulations in Step 2(b), the input $\boldsymbol{x}^{(\ell)}$ is *known* to ONLINE-QUERY (and unknown to IPRR).

Whenever the simulated call to IPRR increments θ^{inner}_i for some $i \in [n]$, ONLINE-QUERY handles it differently in the following three cases:

- **Case 1:** ONLINE-QUERY has already observed the value of x_i (by reaching $\theta^{\text{outer}}_i \geq c_i$). In this case, we know the value of c_i. Then, if the incremented value of θ^{inner}_i also reaches c_i, we reveal the value of $x^{(\ell)}_i$ to IPRR; otherwise, we do nothing.
- **Case 2:** ONLINE-QUERY has not observed x_i and $\theta^{\text{inner}}_i \leq \theta^{\text{outer}}_i$. We would know that $\theta^{\text{inner}}_i \leq \theta^{\text{outer}}_i < c_i$, so $x^{(\ell)}_i$ should not be revealed to IPRR at this moment. We do nothing.

- **Case 3:** Online-Query has not observed x_i and $\theta_i^{\text{inner}} > \theta_i^{\text{outer}}$ after the increment. In this case, we actually increment θ_i^{outer} to match θ_i^{inner}. If x_i gets revealed to us as a result, we would know that $\theta_i^{\text{inner}} = \theta_i^{\text{outer}} \geq c_i$, so we reveal the value of $x_i^{(\ell)}$ to IPRR.

The simulation above ensures that, from the perspective of IPRR, it indeed runs on an instance with unknown cost vector c and input $\boldsymbol{x}^{(\ell)}$. Note that this simulation does not require the full knowledge of c. Also, we always match the investment made by IPRR whenever θ_i^{inner} exceeds θ_i^{outer}. As a result, the cumulative investment of Online-Query in each variable x_i is given by the *maximum* (rather than the sum) over all cumulative investments into $x_i^{(\ell)}$ made by different calls to IPRR throughout the execution of Online-Query.

A standard Chernoff bound gives us the following guarantee on the testing in Step 2(b):

Lemma 26. *Let* $\text{IPRR} = \text{IPRR}(f, \varepsilon, B_i)$. *We have the following:*

- *If* $\text{dist}(\text{IPRR}, f) \geq 4\varepsilon$, *then* $\mathbf{Pr}[\boldsymbol{T}_i > 3\varepsilon]$ *with probability at least* $1 - \Theta(\varepsilon i^{-2})$.
- *If* $\text{dist}(\text{IPRR}, f) \leq 2\varepsilon$, *then* $\mathbf{Pr}[\boldsymbol{T}_i \leq 3\varepsilon]$ *with probability at least* $1 - \Theta(\varepsilon i^{-2})$.

Proof. This is an easy consequence of a standard (multiplicative) Chernoff bound; see, for e.g., Exercise 2.3.5 of [Ver18]. In particular, for i.i.d. Bernoulli(p) random variables $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m$ for $t \in (0, 1)$, we have

$$\mathbf{Pr}\left[\left|\sum_{\ell=1}^{m} \boldsymbol{X}_\ell - pm\right| \geq tpm\right] \leq 2e^{-\Omega(t^2 pm)}.$$

To see Item 1, let $p := \text{dist}(\text{IPRR}, f) \geq 4\varepsilon$ by assumption. Let $\boldsymbol{X}_\ell := \mathbf{1}\{\boldsymbol{b}^{(\ell)} \neq f(\boldsymbol{x}^{(\ell)})\}$ and note that $\boldsymbol{X}_\ell \sim \text{Bernoulli}(p)$. We then have

$$\begin{aligned}
\mathbf{Pr}\left[\boldsymbol{T}_i \leq 3\varepsilon\right] &= \mathbf{Pr}\left[\sum_{\ell=1}^{m_i} \boldsymbol{X}_\ell \leq 3\varepsilon m_i\right] \\
&\leq \mathbf{Pr}\left[\left|\sum_{\ell=1}^{m_i} \boldsymbol{X}_\ell - pm_i\right| \geq 0.01 pm_i\right] \\
&\leq 2e^{-\Omega(\varepsilon m_i)} \leq \Theta\left(\frac{\varepsilon}{i^2}\right), \qquad \left(\text{Since } m_i = \Theta\big(\varepsilon^{-1}\log(\varepsilon^{-1} i)\big)\right)
\end{aligned}$$

where we relied on the fact that $p \geq 4\varepsilon$ to choose $t = 0.01$ while applying the Chernoff bound. The proof of Item 2 is identical. □

Finally, we turn to the proof of Proposition 23, which in turn completes the proof of Theorem 16:

Proof of Proposition 23. Let random variable $\boldsymbol{i}^*$ denote the value of the counter i when the algorithm calls IPRR in Step 2(c). Let $i_0 \geq 1$ be the smallest positive integer such that $2^{i_0} \geq \text{opt}_\varepsilon^{\text{w}}$. Note that, assuming $\text{opt}_\varepsilon^{\text{w}} = \Omega(1)$, we have $2^{i_0} = O(\text{opt}_\varepsilon^{\text{w}})$.

We first show that Online-Query computes f to error $O(\varepsilon)$. Let E be the event that the two conditions in Lemma 26 simultaneously hold for all $i \in [i_0]$. By Lemma 26 and the union bound, we have

$$\mathbf{Pr}[\neg E] \leq \sum_{i=1}^{i_0} \Theta(\varepsilon i^{-2}) \leq O(\varepsilon).$$

Moreover, we note that event E implies the following two conditions:

- $\boldsymbol{i}^* \le i_0$, i.e., Online-Query must run IPRR in Step 2(c) using one of the first i_0 guesses $B_1, B_2, \ldots, B_{i_0}$. This is because, by definition of i_0, we have $B_{i_0} = 2^{i_0} \ge \mathrm{opt}^{\mathrm{w}}_{\varepsilon}$. By Proposition 24, $\mathrm{IPRR}(f, \varepsilon, B_{i_0})$ computes f to error 2ε. Event E then ensures that, if Online-Query does not halt in the first $i_0 - 1$ iterations, the condition $\boldsymbol{T}_{i_0} \le 3\varepsilon$ would be true, in which case Online-Query runs IPRR in Step 2(c) with parameter B_{i_0}.
- $\mathrm{error}_f(\mathrm{IPRR}(f, \varepsilon, B_{\boldsymbol{i}^*})) \le 4\varepsilon$, i.e., when Online-Query runs IPRR in Step 2(c), the parameter $B_{\boldsymbol{i}^*}$ ensures that IPRR is a 4ε-error algorithm. This is because, for Online-Query to run IPRR in Step 2(c), the condition $\boldsymbol{T}_{\boldsymbol{i}^*} \le 3\varepsilon$ must hold. Event E then ensures that $\mathrm{IPRR}(f, \varepsilon, B_{\boldsymbol{i}^*})$ computes f to error 4ε.

The two observations above show that, conditioned on the event E, the error probability is bounded by 4ε. Thus, we have that

$$\mathrm{error}_f(\textsc{Online-Query}) \le \mathbf{Pr}[\neg E] + 4\varepsilon \cdot \mathbf{Pr}[E] = O(\varepsilon)\,.$$

Next, we turn to controlling the expected cost of Online-Query. It suffices to show that, for every $j \in [n]$, the expected investment of Online-Query in x_j is upper bounded by

$$\beta + \mathbf{Inf}_j[f] \cdot \left(1 + \ln \frac{1}{\mathbf{Inf}_j[f]}\right) \cdot O\left(\frac{\mathrm{opt}^{\mathrm{w}}_{\varepsilon}}{\varepsilon^2} \log\left(\frac{\log \mathrm{opt}^{\mathrm{w}}_{\varepsilon}}{\varepsilon}\right)\right). \tag{17}$$

The proposition would directly follow from summing the above over $j \in [n]$.

Towards proving the upper bound in Equation (17), we fix $j \in [n]$ and examine the expected investment in x_j when $\mathrm{IPRR}(f, \varepsilon, B_i)$ runs on a uniformly random input $\boldsymbol{x} \sim \{0,1\}^n$. The analysis will be similar to one from the proof of Theorem 18, so we will be succinct. On any fixed input x, let $\Pi(x)$, $\pi^{(j)}$ and θ^*_j be as in the proof of Theorem 18. Given that the algorithm has not yet halted as a result of Step 2(b) before incrementing θ_j from $\theta^*_j - \beta$ to θ^*_j, we have

$$\frac{\mathbf{Inf}_j[f_{\pi^{(j)}}]}{\theta^*_j - \beta} \ge \frac{\varepsilon}{B}\,, \qquad \text{and so} \qquad \theta^*_j \le \beta + \frac{B}{\varepsilon} \cdot \mathbf{Inf}_j[f_{\pi^{(j)}}] \le \beta + \frac{B}{\varepsilon} \cdot \max_{\pi \in \Pi(x)} \mathbf{Inf}_j[f_\pi]\,.$$

For each $i \in [\boldsymbol{i}^*]$, let

$$\boldsymbol{Y}_i \coloneqq \max_{\pi \in \Pi(\boldsymbol{x}^{(1)}) \cup \cdots \cup \Pi(\boldsymbol{x}^{(m_i)}) \cup \Pi(\boldsymbol{x})} \mathbf{Inf}_j[f_\pi]$$

denote the maximum influence $\mathbf{Inf}_j[f_\pi]$ when π ranges over all restrictions encountered by $\mathrm{IPRR}(f, \varepsilon, B_i)$ when it runs on $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m_i)}$ (and possibly the unknown input $\boldsymbol{x}$) using the i-th guess B_i. Then, the maximum investment in x_j made by any call to $\mathrm{IPRR}(f, \varepsilon, B_i)$ is upper bounded by

$$\beta + \frac{B_i}{\varepsilon} \cdot \boldsymbol{Y}_i.$$

Recall from Remark 25 that the cumulative investment made by Online-Query in each x_j is exactly the maximum investment in x_j among all calls to IPRR during its execution. Therefore, the expected investment in x_j from Online-Query is upper bounded by

$$\mathbf{E}\left[\beta + \max_{i \in [\boldsymbol{i}^*]} \frac{B_i}{\varepsilon} \cdot \boldsymbol{Y}_i\right] \le \beta + \mathbf{E}\left[\sum_{i=1}^{\boldsymbol{i}^*} \frac{B_i}{\varepsilon} \cdot \boldsymbol{Y}_i\right] = \beta + \sum_{i=1}^{\infty} \frac{B_i}{\varepsilon} \cdot \mathbf{E}\left[\boldsymbol{Y}_i \mid \boldsymbol{i}^* \ge i\right] \cdot \mathbf{Pr}\left[\boldsymbol{i}^* \ge i\right]. \tag{18}$$

To control the last summation in the above, we will prove the following two claims:

- For every $i \geq 1$, $\mathbf{E}\left[\boldsymbol{Y}_i \mid \boldsymbol{i}^* \geq i\right] \leq (m_i + 1) \cdot \mathbf{Inf}_j[f] \cdot \left(1 + \ln \frac{1}{\mathbf{Inf}_j[f]}\right)$.
- For every $i > i_0$, $\mathbf{Pr}\left[\boldsymbol{i}^* \geq i\right] \leq \frac{1}{4^{i-i_0}}$.

Assuming the two claims above, the upper bound in Equation (17) follows from a straightforward calculation: Plugging the upper bound on $\mathbf{E}\left[\boldsymbol{Y}_i \mid \boldsymbol{i}^* \geq i\right]$ as well as $B_i = 2^i$ and $m_i = O(\varepsilon^{-1} \log(i/\varepsilon))$ into the last summation in Equation (18) gives

$$\begin{aligned}
&\sum_{i=1}^{\infty} \frac{B_i}{\varepsilon} \cdot (m_i + 1) \cdot \mathbf{Inf}_j[f] \cdot \left(1 + \ln \frac{1}{\mathbf{Inf}_j[f]}\right) \cdot \mathbf{Pr}\left[\boldsymbol{i}^* \geq i\right] \\
&\leq O\left(\frac{1}{\varepsilon^2}\right) \cdot \mathbf{Inf}_j[f] \cdot \left(1 + \ln \frac{1}{\mathbf{Inf}_j[f]}\right) \cdot \sum_{i=1}^{\infty} 2^i \log \frac{i}{\varepsilon} \cdot \mathbf{Pr}\left[\boldsymbol{i}^* \geq i\right].
\end{aligned}$$

The summation $\sum_{i=1}^{\infty} 2^i \log \frac{i}{\varepsilon} \cdot \mathbf{Pr}\left[\boldsymbol{i}^* \geq i\right]$ can be further rewritten as

$$\begin{aligned}
&\sum_{i=1}^{i_0} 2^i \log \frac{i}{\varepsilon} \cdot \mathbf{Pr}\left[\boldsymbol{i}^* \geq i\right] + \sum_{i=i_0+1}^{\infty} 2^i \log \frac{i}{\varepsilon} \cdot \mathbf{Pr}\left[\boldsymbol{i}^* \geq i\right] \\
&\leq \sum_{i=1}^{i_0} 2^i \log \frac{i}{\varepsilon} + \sum_{i=i_0+1}^{\infty} 2^i \log \frac{i}{\varepsilon} \cdot \frac{1}{4^{i-i_0}} \\
&= O\left(2^{i_0} \log \frac{i_0}{\varepsilon}\right) = O\left(\mathrm{opt}_\varepsilon^{\mathrm{w}} \log\left(\frac{\log \mathrm{opt}_\varepsilon^{\mathrm{w}}}{\varepsilon}\right)\right).
\end{aligned}$$

The first step above upper bounds $\mathbf{Pr}\left[\boldsymbol{i}^* \geq 1\right]$ by 1 when $i \leq i_0$, and by $1/4^{i-i_0}$ when $i > i_0$. The second step observes that the two summations are dominated by the terms at $i = i_0$ and $i = i_0 + 1$, respectively. The last step applies $2^{i_0} = O(\mathrm{opt}_\varepsilon^{\mathrm{w}})$, which follows from the definition of i_0.

It remains to verify the two claims regarding $\mathbf{E}\left[\boldsymbol{Y}_i \mid \boldsymbol{i}^* \geq i\right]$ and $\mathbf{Pr}\left[\boldsymbol{i}^* \geq i\right]$. For the first claim, note that, conditioning on $\boldsymbol{i}^* \geq i$, the m_i inputs $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m_i)}$ as well as the unknown input $\boldsymbol{x}$ are still uniformly distributed over $\{0,1\}^n$. Proceeding *mutatis mutandis* as in the martingale argument from the proof of Theorem 18, we can show that, when $\mathrm{IPRR}(f, \varepsilon, B_i)$ runs on each of these $m_i + 1$ inputs, the expected maximum value of $\mathbf{Inf}_j[f_\pi]$ is at most $\mathbf{Inf}_j[f] \cdot \left(1 + \ln \frac{1}{\mathbf{Inf}_j[f]}\right)$. Recall that $\boldsymbol{Y}_i$ is defined as the maximum over these $m_i + 1$ maxima. By relaxing the maximum to a sum, we obtain the claim that $\mathbf{E}\left[\boldsymbol{Y}_i \mid \boldsymbol{i}^* \geq i\right] \leq (m_i + 1)\mathbf{Inf}_j[f] \cdot \left(1 + \ln \frac{1}{\mathbf{Inf}_j[f]}\right)$.

For the second claim, note that for every $i \geq i_0$, it holds that $B_i \geq 2^{i_0} \geq \mathrm{opt}_\varepsilon^{\mathrm{w}}$. Then, Proposition 24 and Lemma 26 together imply that Online-Query would halt in Step 2(c) except with probability $O(\varepsilon i^{-2})$. By carefully choosing the hidden constant factor in $O(\cdot)$, this probability is at most $1/4$. Then, in order for $\boldsymbol{i}^* \geq i \geq i_0 + 1$ to happen, Online-Query must fail to halt on each of the $i - i_0$ guesses $B_{i_0}, B_{i_0+1}, \ldots, B_{i-1}$, which happens with probability at most $1/4^{i-i_0}$.

This verifies the two claims, thus establishing that Equation (17) upper bounds the expected investment in x_j and proving the proposition.

□

4.4 A Sharper Analysis for Symmetric Functions

Recall that a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *symmetric* if the following holds:

$$f(x) = f(y) \qquad \text{whenever} \qquad \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \,.$$

Note that many natural functions, including AND, OR, and Majority are symmetric. Furthermore, prior work on priced query strategies has designed algorithms tailored to symmetric functions [GGHK18, Hel18, HKLW22, HLS24].

We show that the algorithm WARMUP-IPRR achieves the following performance guarantee:

Theorem 27 (Formal version of Theorem 5)**.** *For any symmetric function $f : \{0,1\}^n \to \{0,1\}$, the algorithm* WARMUP-IPRR(f, ε) *with unit investment β is an ε-error algorithm with expected cost*

$$\beta n + \mathrm{opt}_0^{\mathrm{avg}} \cdot O\left(\log \frac{1}{\varepsilon}\right).$$

Necessity of the log factor. To see why the $\log(1/\varepsilon)$ factor is necessary, consider the case where f is the AND function over $n = \Theta(\log(1/\varepsilon))$ variables, each with a unit cost. The optimal algorithm simply queries the variables in order, and stops as soon as any of the input bits is revealed to be a zero. The expected cost is then $\mathrm{opt}_0^{\mathrm{avg}} = O(1)$. On the other hand, the WARMUP-IPRR algorithm needs to pay the cost of all the n variables before seeing any input bit, leading to a gap of $\Omega(n) = \Omega(\log(1/\varepsilon))$. Note that we cannot take n to be larger than $\log(1/\varepsilon)$; otherwise f would be ε-close to the constant zero function and WARMUP-IPRR simply outputs 0 without incurring any cost. In Section 6, we show that *any* online algorithm needs to pay this additional $\log(1/\varepsilon)$ factor.

4.4.1 Analysis of the Optimal Algorithm and Round-Robin

For brevity, we rename the input bits in increasing order of their costs, i.e., $c_1 \le c_2 \le \cdots \le c_n$. Below are the behavior and the associated costs of the optimal (offline) algorithm and the WARMUP-IPRR algorithm:

- The offline algorithm queries $x_1, x_2, \ldots, x_n$ in order, and stops as soon as for some t, $f|_{x_1,\ldots,x_t}$ becomes a constant function. Let $\boldsymbol{\tau}^{(0)}$ denote the stopping time of the algorithm, i.e., the smallest such index t on a uniformly random input $\boldsymbol{x} \sim \{0,1\}^n$. The expected cost is given by

$$\mathrm{opt}_0^{\mathrm{avg}}(f, c) = \sum_{i=1}^{n} c_i \cdot \mathbf{Pr}\left[\boldsymbol{\tau}^{(0)} \ge i\right]. \tag{19}$$

- When running WARMUP-IPRR, the input bits $x_1, x_2, \ldots, x_n$ are revealed in order, and the algorithm stops as soon as for some t, $f|_{x_1,\ldots,x_t}$ becomes ε-close to a constant function. Let $\boldsymbol{\tau}^{(\varepsilon)}$ denote this stopping time for $\boldsymbol{x} \sim \{0,1\}^n$. The expected cost of WARMUP-IPRR is then given by

$$\text{avg-cost}_c^f(\text{WARMUP-IPRR}) \le \beta n + \sum_{i=1}^{n} c_i \cdot \left[\mathbf{Pr}\left[\boldsymbol{\tau}^{(\varepsilon)} \ge i\right] + (n-i) \cdot \mathbf{Pr}\left[\boldsymbol{\tau}^{(\varepsilon)} = i\right]\right]. \tag{20}$$

  Here, the term $(n-i) \cdot \mathbf{Pr}\left[\boldsymbol{\tau}^{(\varepsilon)} = i\right]$ is due to that if $\boldsymbol{x}_1$ through $\boldsymbol{x}_{\tau^{(\varepsilon)}}$ get revealed by the end of the algorithm, we also pay a cost of $c_{\tau^{(\varepsilon)}}$ on each of the input bits $\boldsymbol{x}_{\tau^{(\varepsilon)}+1}$ through x_n.

  See Figure 3 for an illustration of this.

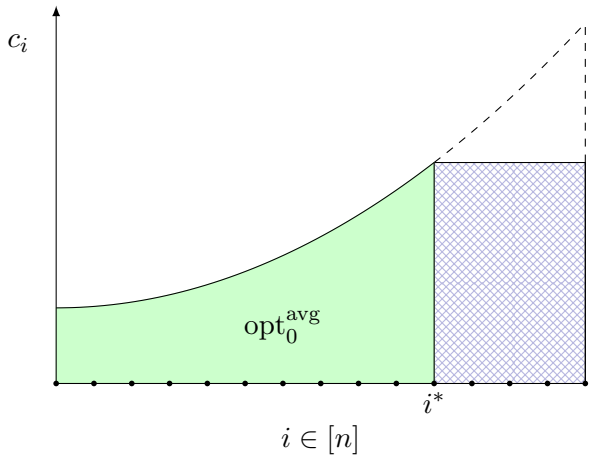

Figure 3: An illustration of the behaviors of the optimal offline algorithm and WARMUP-IPRR on a symmetric function. Here, we assume that the costs are ordered as $c_1 \le c_2 \le \cdots \le c_n$, and i^* denotes the index s.t. $\mathrm{bias}(f) \le \varepsilon$ after fixing $x_1, \ldots, x_{i^*}$. The cross-hatched region denotes the additional cost incurred by WARMUP-IPRR in contrast to the optimal offline algorithm.

4.4.2 Proof of Theorem 27

In light of Equations (19) and (20), to prove Theorem 27, it suffices to show that for any symmetric function f and any set of costs $0 \le c_1 \le c_2 \le \cdots \le c_n$, it holds that

$$\sum_{i=1}^{n} c_i \cdot \left[\mathbf{Pr}\left[\boldsymbol{\tau}^{(\varepsilon)} \ge i\right] + (n-i) \cdot \mathbf{Pr}\left[\boldsymbol{\tau}^{(\varepsilon)} = i\right] \right] \le O\left(\log \frac{1}{\varepsilon}\right) \cdot \sum_{i=1}^{n} c_i \cdot \mathbf{Pr}\left[\boldsymbol{\tau}^{(0)} \ge i\right]. \tag{21}$$

Note that it is sufficient to prove the above for costs of form $c_i = \mathbf{1}\{i \ge i^*\}$ for some $i^* \in [n]$, i.e., there is a unit cost for every index above the threshold i^*, and the cost is zero below i^*. This is because: (1) every increasing sequence of costs $c_1 \le c_2 \le \cdots \le c_n$ can be written as a conic (non-negative) combination of costs of form $c_i = \mathbf{1}\{i \ge i^*\}$; (2) both sides of Equation (21) are linear in c.

With the observation above, it remains to show that, for any symmetric f and $i^* \in [n]$, it holds that

$$(n - i^* + 1) \cdot \mathbf{Pr}\left[\boldsymbol{\tau}^{(\varepsilon)} \ge i^*\right] \le O(\log(1/\varepsilon)) \cdot \sum_{i=i^*}^{n} \mathbf{Pr}\left[\boldsymbol{\tau}^{(0)} \ge i\right],$$

which is equivalent to

$$(n - i^* + 1) \cdot \mathbf{Pr}\left[\boldsymbol{\tau}^{(\varepsilon)} \ge i^*\right] \le O(\log(1/\varepsilon)) \cdot \mathbf{E}\left[\max\{\boldsymbol{\tau}^{(0)} - i^* + 1, 0\}\right]. \tag{22}$$

We will prove an even stronger inequality than the above—after conditioning on the realization of $x_1, \ldots, x_{i^*-1}$, the inequality above still holds. Note that the input bits $x_1, \ldots, x_{i^*-1}$ determines whether $\boldsymbol{\tau}^{(\varepsilon)} < i^*$ or $\boldsymbol{\tau}^{(\varepsilon)} \ge i^*$. In the former case, the contribution to the left-hand side of Equation (22) is zero, so the inequality trivially holds. In the remaining case where $\tau^{(\varepsilon)} \ge i^*$ given x_1 through x_{i^*-1}, the left-hand side of Equation (22) reduces to $n - i^* + 1$. The right-hand side, on the other hand, is simply $O(\log(1/\varepsilon))$ times the expectation of the stopping time "$\boldsymbol{\tau}^{(0)}$" defined with respect to the symmetric function $f|_{x_1,\ldots,x_{i^*-1}}$ over $n - i^* + 1$ variables.

Therefore, it remains to prove the following lemma:

Lemma 28. *For any symmetric function f on n variables with* $\mathrm{bias}(f) \geq \varepsilon$*, it holds that*

$$\mathbf{E}\left[\boldsymbol{\tau}^{(0)}\right] \geq \Omega\left(\frac{n}{\log(1/\varepsilon)}\right),$$

where the stopping time $\tau^{(0)}$ (defined over the uniform drawing of $x_1, x_2, \ldots, x_n$) is the smallest value t such that $f|_{x_1,\ldots,x_t}$ reduces to a constant.

Proof. If $\varepsilon \leq e^{-n/100}$, we only need to show that $\mathbf{E}\left[\boldsymbol{\tau}^{(0)}\right] \geq \Omega(1)$, which is trivially true since $\boldsymbol{\tau}^{(0)}$ is at least 1. Otherwise, we claim that $\boldsymbol{\tau}^{(0)}$ must be at least $n/2$ with probability $\Omega(1)$, which gives the stronger bound of $\mathbf{E}\left[\boldsymbol{\tau}^{(0)}\right] \geq \Omega(n)$.

To see this, note that after $n/2$ input bits are revealed, with probability $\Omega(1)$, both 0 and 1 appear $\leq n/3$ times. If f reduces to a constant function, f must be constant on every input x with Hamming weight in $[n/3, 2n/3]$, which, by a standard Chernoff bound, contradicts the assumption that f is at distance $\varepsilon \geq e^{-n/100}$ from constant functions. □

5 Query Strategies for Functions with Shallow Decision Trees

In this section, we turn to the setting of Theorem 6, where the function can be represented as a shallow decision tree, either given to the algorithm or unknown. We begin by considering the special case where the decision tree is given and *everywhere-influential*, meaning that each internal node queries a variable with sufficiently high influence. Then, we deal with the more general case, where we first apply the pruning lemma of [BLQT22] to transform the tree into an everywhere-influential one, and then follow the pruned tree. When the decision tree representation is not given, we apply the learning algorithm of [BLQT22] to learn an everywhere-influential decision tree that approximates the function, and then query the variables according to the learned tree.

5.1 Follow the Everywhere-Influential Tree

We adopt the definition of everywhere τ-influential decision trees in [BLQT22]. In the rest of this section, for each internal node v in a decision tree, we let $\mathrm{ind}(v) \in [n]$ denote the index of the variable queried by v.

Definition 29 (Everywhere τ-influential)**.** For $\tau \in [0, 1]$, a decision tree T is everywhere τ-influential with respect to a function $f : \{0,1\}^n \to \{0,1\}$ if, for every internal node v in T,

$$\mathbf{Inf}_{\mathrm{ind}(v)}[f_v] \geq \tau,$$

where f_v is the restriction of f induced by the root-to-v path in T.

We start by analyzing the most straightforward algorithm when a decision tree representation is given—the algorithm simply follows the tree.

Definition 30 (Follow the tree)**.** Given the decision tree T, the algorithm starts at the root of T. At each internal node v, the algorithm increments $\theta_{\mathrm{ind}(v)}$ by β until it reaches the cost $c_{\mathrm{ind}(v)}$, at which point $x_{\mathrm{ind}(v)}$ is revealed. The algorithm follows the computation path of T until a leaf node is reached, at which point the algorithm outputs the label of that leaf.

Compared to the IPRR algorithms, "follow the tree" blindly trusts the given decision tree—at any point, the algorithm concentrates its investment on the variable specified by the tree, rather than hedging among multiple variables. Note that, if the computation path of T queries a variable x_i, the investment θ_i made by "follow the tree" is at most $c_i + \beta$, where β is the unit investment of the algorithm (cf. Definition 11). This leads to the additional βn term in all of our upper bounds. Recall that the algorithm may pick a sufficiently small $\beta = 1/\mathrm{poly}(n)$ to make the βn term negligible.

When the given decision tree representation of f is everywhere τ-influential (with respect to f), "follow the tree" is $(1/\tau)$-competitive against the zero-error average-case benchmark.

Proposition 31. *For any $\tau \in (0, 1]$, when "follow the tree" is given an everywhere τ-influential decision tree representation of the function, the expected cost of the algorithm is at most*

$$\beta n + \mathrm{opt}_0^{\mathrm{avg}} \cdot \frac{1}{\tau}.$$

Proposition 31 follows from the two simple lemmas below. The first lemma lower bounds the offline benchmark $\mathrm{opt}_0^{\mathrm{avg}}$ by a cost-weighted sum of the influences.

Lemma 32. *For any function $f : \{0,1\}^n \to \{0,1\}$, the zero-error average-case benchmark satisfies*

$$\mathrm{opt}_0^{\mathrm{avg}} \geq \sum_{i=1}^{n} c_i \cdot \mathbf{Inf}_i[f].$$

Proof. Let $\mathcal{A}$ be a zero-error offline algorithm computing function $f : \{0,1\}^n \to \{0,1\}$. We claim that, when $\mathcal{A}$ runs on an input $\boldsymbol{x}$ that satisfies $f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i})$ (i.e., variable x_i is pivotal on $\boldsymbol{x}$), x_i must be revealed to $\mathcal{A}$. Otherwise, $\mathcal{A}$ would have the same behavior on the alternative input $\boldsymbol{x}^{\oplus i}$ with a different function value, and thus cannot have a zero error. Therefore, we have

$$\Pr_{\boldsymbol{x} \sim \{0,1\}^n} [\mathcal{A} \text{ reveals } x_i] \geq \Pr_{\boldsymbol{x} \sim \{0,1\}^n} \left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i}) \right] = \mathbf{Inf}_i[f].$$

It follows that, under every cost vector $c \in \mathbb{R}^n_{\geq 0}$, the expected cost of $\mathcal{A}$ is at least

$$\sum_{i=1}^{n} c_i \cdot \Pr_{\boldsymbol{x} \sim \{0,1\}^n} [\mathcal{A} \text{ reveals } x_i] \geq \sum_{i=1}^{n} c_i \cdot \mathbf{Inf}_i[f].$$

This gives the desired lower bound $\mathrm{opt}_0^{\mathrm{avg}} \geq \sum_{i=1}^{n} c_i \cdot \mathbf{Inf}_i[f]$. □

The second lemma shows that the influence of each variable x_i is lower bounded by τ times the probability that it is queried in an everywhere τ-influential decision tree.

Lemma 33. *Suppose that decision tree T is everywhere τ-influential with respect to function $f : \{0,1\}^n \to \{0,1\}$. Then, for every $i \in [n]$,*

$$\mathbf{Inf}_i[f] \geq \tau \cdot \delta_i(T).$$

Proof. Fix an index $i \in [n]$ and consider a uniformly random input $\boldsymbol{x} \sim \{0,1\}^n$. The goal is to lower bound $\mathbf{Inf}_i[f]$, namely, the probability of $f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i})$. Consider the computation path when $T(\boldsymbol{x})$ is evaluated. Let

$$V \coloneqq \{v : v \text{ is an internal node of } T, \mathrm{ind}(v) = i\}$$

be the set of internal nodes that query x_i. For each $v \in V$, let E_v denote the event that node v is reached when computing $T(\boldsymbol{x})$. Note that the events $\{E_v : v \in V\}$ are disjoint, and

$$\sum_{v \in V} \mathbf{Pr}\left[E_v\right] = \delta_i(T).$$

Let π_v denote the restriction induced by the root-to-v path. We note that event E_v is equivalent to that the input $\boldsymbol{x}$ agrees with restriction π_v. Therefore, we have

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{\boldsymbol{x} \sim \{0,1\}^n}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i}) \mid E_v\right] &= \mathop{\mathbf{Pr}}_{\boldsymbol{x} \sim \{0,1\}^n}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i}) \mid \boldsymbol{x} \triangleleft \pi_v\right] \\
&= \mathop{\mathbf{Pr}}_{\boldsymbol{x} \sim \{0,1\}^n}\left[f_v(\boldsymbol{x}) \neq f_v(\boldsymbol{x}^{\oplus i})\right] \\
&= \mathbf{Inf}_i[f_v] \geq \tau,
\end{aligned}$$

where the last step applies the fact that $\mathrm{ind}(v) = i$ and T is everywhere τ-influential with respect to f.

Therefore, we conclude that

$$\begin{aligned}
\tau \cdot \delta_i(T) &= \sum_{v \in V} \tau \cdot \mathbf{Pr}\left[E_v\right] \\
&\leq \sum_{v \in V} \mathop{\mathbf{Pr}}_{\boldsymbol{x} \sim \{0,1\}^n}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i}) \mid E_v\right] \cdot \mathbf{Pr}\left[E_v\right] \\
&\leq \mathop{\mathbf{Pr}}_{\boldsymbol{x} \sim \{0,1\}^n}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i})\right] && \text{(law of total probability)} \\
&= \mathbf{Inf}_i[f]\,, && \text{(definition of influence)}
\end{aligned}$$

which completes the proof. □

Proposition 31 is then an immediate consequence of the two lemmas above.

Proof of Proposition 31. The cost of "follow the tree" is given by

$$\sum_{i=1}^{n} (c_i + \beta) \cdot \delta_i(T) \leq \beta n + \frac{1}{\tau} \sum_{i=1}^{n} c_i \cdot \mathbf{Inf}_i[f] \leq \beta n + \mathrm{opt}_0^{\mathrm{avg}} \cdot \frac{1}{\tau},$$

where the first step applies Lemma 33 and the second step applies Lemma 32. □

5.2 Follow the Pruned Tree

In the more general case that the algorithm is given an arbitrary decision tree representation T of the function, our strategy is a natural one: First, we "prune" the decision tree T into an everywhere τ-influential tree T', so that T' agrees with T on most of the inputs. Such a pruning result was given by [BLQT22]. Then, we run the "follow the tree" algorithm on T'.

The pruning lemma of [BLQT22]. Let $\Delta(T)$ denote the average depth of decision tree T, which is formally defined as:

$$\Delta(T) := \begin{cases} 0, & \text{if } T \text{ has only one node,} \\ 1 + \frac{1}{2}\left[\Delta(T_0) + \Delta(T_1)\right], & \text{if the root of } T \text{ has subtrees } T_0 \text{ and } T_1. \end{cases}$$

We will use the following version of the pruning lemma of [BLQT22].

Lemma 34 (Theorem 4 of [BLQT22])**.** *For any decision tree T and $\tau \in [0,1]$, there is an everywhere τ-influential decision tree T' (with respect to the function computed by T) such that*

$$\mathrm{dist}(T, T') \leq \tau \cdot \Delta(T).$$

Furthermore, such a T' can be computed from T in time polynomial in the size of T.

Definition 35 (Follow the pruned tree)**.** Given the decision tree T and accuracy parameter $\varepsilon \in (0,1]$, the algorithm applies Lemma 34 to T and $\tau := \varepsilon/\Delta(T)$ and efficiently samples a pruned tree T'. Then, the algorithm runs "follow the tree" (Definition 30) on T'.

Theorem 36 (Formal version of Theorem 6)**.** *Given $\varepsilon \in (0,1]$ and a decision tree representation T of function f, "follow the pruned tree" computes f up to an error of ε and with an expected cost of at most*

$$\beta n + \mathrm{opt}_0^{\mathrm{avg}} \cdot \frac{\Delta(T)}{\varepsilon}.$$

Proof. Let $\tau := \varepsilon/\Delta(T)$ and T' be the pruned tree computed from Lemma 34. Then, the error of "follow the pruned tree" is at most

$$\mathrm{dist}(T, T') \leq \tau \cdot \Delta(T) = \varepsilon.$$

Furthermore, the expected cost of the algorithm is given by

$$\sum_{i=1}^{n} (c_i + \beta) \cdot \delta_i(T') \leq \beta n + \frac{1}{\tau} \sum_{i=1}^{n} c_i \cdot \mathbf{Inf}_i[f] \leq \beta n + \mathrm{opt}_0^{\mathrm{avg}} \cdot \frac{\Delta(T)}{\varepsilon},$$

where the first step applies Lemma 33 to T' and f, while the second step applies Lemma 32. □

Remark 37. Suppose that the function f is guaranteed to be computed by some decision tree T of average depth $d = \Delta(T)$, but T is not given. In this case, we can apply the decision tree learning algorithm of [BLQT22, Theorem 7] to compute a decision tree T' such that: (1) $\mathrm{dist}(T, T') \leq \varepsilon$; (2) T' is everywhere τ-influential with respect to f for $\tau = \Omega(\varepsilon/d)$. The algorithm only requires query access to f, and runs in time

$$\mathrm{poly}(n) \cdot \left(\frac{d}{\varepsilon}\right)^{O(d/\varepsilon)}.$$

Then, running "follow the tree" on T' gives an error $\leq \varepsilon$ and expected cost of at most

$$\beta n + \mathrm{opt}_0^{\mathrm{avg}} \cdot O\left(\frac{d}{\varepsilon}\right).$$

Furthermore, when the value of $d = \Delta(T)$ is unknown, we can obtain the same result by guessing the value of $d = 1, 2, 3, \ldots$. For each guess of d, we can estimate the error of the tree learned by the algorithm of [BLQT22] by making membership queries to f on uniformly random inputs. We increment the guess on d until the resulting error is below ε. This gives an algorithm with expected cost $\mathrm{opt}_0^{\mathrm{avg}} \cdot O(d/\varepsilon)$ without knowing the value of $\Delta(T)$ in advance.

6 Separating the Offline and Online Settings: Proof of Theorem 1

We turn to the proof of Theorem 41.

6.1 Warmup: A Linear Lower Bound against Zero-Error Algorithms

We start with an $\Omega(n)$ lower bound on the competition ratio, which applies to all *zero-error* algorithms for the AND function. The costs of the n variables are known to be a permutation of $[n] = \{1, 2, \ldots, n\}$, but the exact ordering of the costs is unknown. Formally, we consider a hard distribution over instances defined as follows.

Definition 38 (AND instance)**.** For integer $n \geq 1$, $f(x) = \bigwedge_{i=1}^{n} x_i$ is the AND function on n variables. The costs $(c_1, c_2, \ldots, c_n)$ are set to a permutation of $[n]$ chosen uniformly at random.

Proposition 39 (Formal version of Proposition 9)**.** *On the* AND *instance with n variables, we have the offline benchmark $\mathrm{opt}_0^{\mathrm{avg}} = O(1)$, while every zero-error online algorithm has an expected cost of $\Omega(n)$.*

Proof. We first note that $\mathrm{opt}_0^{\mathrm{avg}} = O(1)$: Since the offline algorithm knows the costs, it may query the variables in increasing order of costs, and stop whenever a zero is encountered. The algorithm is always correct, and the variable with cost i is queried with probability $2^{-(i-1)}$ (namely, when all the $i-1$ variables with lower costs take value 1). Thus, the algorithm has an expected cost of

$$\sum_{i=1}^{n} 2^{-(i-1)} \cdot i \leq \sum_{i=1}^{+\infty} 2^{-(i-1)} \cdot i = 4.$$

It remains to show that every zero-error algorithm must incur an expected cost of $\Omega(n)$. By an averaging argument, it suffices to prove this for deterministic algorithms. This is because any zero-error randomized algorithm can be viewed as a mixture of zero-error deterministic algorithms. If we could lower bound the cost of every zero-error deterministic algorithm by $\Omega(n)$, the same lower bound holds for the mixture as well.

Fix a deterministic algorithm $\mathcal{A}$. Let $\mathcal{A}^{(1)}$ denote the simulation of $\mathcal{A}$ on an AND instance with $c_i = n$ for every $i \in [n]$. Note that this is *not* an instance from Definition 38; later, we will couple this simulation with the execution of $\mathcal{A}$ on an actual instance. We stop the simulation as soon as either one of the following happens:

- **Case 1.** $\mathcal{A}$ "terminates voluntarily" by outputting an answer.
- **Case 2.** The total cost, $\sum_{i=1}^{n} \theta_i$, reaches $n/2$. Formally, when $\mathcal{A}$ attempts to increase some θ_i, we check whether $\sum_{i=1}^{n} \theta_i \geq n/2$ would hold after the increase. If so, we terminate the algorithm before θ_i gets increased.

Since $\mathcal{A}$ is deterministic, when $\mathcal{A}^{(1)}$ terminates, the investment in each variable x_i is a deterministic value (denoted by a_i). We note that $\sum_{i=1}^{n} a_i < n/2$; otherwise, the simulation should have stopped earlier (by Case 2). Since $c_i = n$ for every $i \in [n]$, none of the n variables is revealed before $\mathcal{A}^{(1)}$ terminates.

Now, suppose that we run $\mathcal{A}$ on an actual instance in Definition 38, i.e., the cost vector c is a random permutation of $\{1, 2, \ldots, n\}$. Call this simulation $\mathcal{A}^{(2)}$. We will show that $\mathcal{A}^{(2)}$ agrees with $\mathcal{A}^{(1)}$ with probability $\Omega(1)$. Formally, we have

$$\Pr_{c}\left[c_i > a_i,\ \forall i \in [n]\right] \geq \Omega(1). \tag{23}$$

Assuming Equation (23), with probability $\Omega(1)$ over the randomness in c, $\mathcal{A}^{(2)}$ agrees with $\mathcal{A}^{(1)}$ until $\mathcal{A}^{(1)}$ terminates. Conditioning on this agreement, Case 1—that $\mathcal{A}^{(1)}$ outputs an answer before the total investment reaches $n/2$—cannot be true. This is because none of the variables has been

revealed so far, so $\mathcal{A}^{(2)}$ would incur a non-zero error by outputting an answer at that time. Thus, we must be in Case 2, i.e., $\mathcal{A}^{(1)}$ attempts to reach $\sum_{i=1}^{n} \theta_i \geq n/2$ when we stop the simulation. Since $\mathcal{A}^{(2)}$ agrees with $\mathcal{A}^{(1)}$, $\mathcal{A}^{(2)}$ incurs a cost $\geq n/2$ before terminating. This would then imply that $\mathcal{A}^{(2)}$ has an expected cost of $\Omega(1) \cdot (n/2) = \Omega(n)$.

To prove Equation (23), we note that each constraint $c_i > a_i$ gets violated if and only if c_i is in $\{1, 2, \ldots, n\} \cap [0, a_i]$, a set of size at most a_i. Over the randomness in the permutation c, c_i is uniformly distributed over the size-n set $\{1, 2, \ldots, n\}$. Therefore, we have

$$\mathop{\mathbf{Pr}}_{c}[c_i \leq a_i] \leq \frac{a_i}{n},$$

and thus, by the union bound,

$$\mathop{\mathbf{Pr}}_{c}\left[c_i > a_i,\ \forall i \in [n]\right] \geq 1 - \sum_{i=1}^{n} \mathop{\mathbf{Pr}}_{c}[c_i \leq a_i] \geq 1 - \sum_{i=1}^{n} \frac{a_i}{n} \geq 1 - \frac{n/2}{n} = \frac{1}{2}.$$

□

6.2 A Logarithmic Lower Bound against Constant-Error Algorithms

Even if we allow the algorithm to have a constant error probability, we still have an $\Omega(\log n)$ lower bound on the competitive ratio. We prove this lower bound using the Tribes function, which we formally define as follows.

Definition 40 (Tribes instance)**.** For integer $w \geq 1$, $f(x) \coloneqq \bigvee_{i=1}^{2^w} \bigwedge_{j=1}^{w} x_{i,j}$ is the Tribes function with 2^w tribes of width w, where we rename the $n = 2^w \cdot w$ variables as $(x_{i,j})_{i \in [2^w], j \in [w]}$ for clarity. For each $i \in [2^w]$, the costs $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ are set to a permutation of $[w]$ chosen independently and uniformly at random.

Theorem 41 (Formal version of Theorem 1)**.** *For every $\varepsilon_0 \in [0, 1/4)$ and integer $w \geq 1$, on the* Tribes *instance with width w, we have the offline benchmark $\mathrm{opt}_0^{\mathrm{avg}} = O(2^w)$, while every ε_0-error online algorithm has an expected cost of $\Omega(2^w \cdot w)$.*

Since $w = \Theta(\log n)$ in Definition 40, the theorem shows that an online algorithm can be at best $\Omega(\log n)$-competitive compared to the benchmark $\mathrm{opt}_0^{\mathrm{avg}}$.

We start with an overview of the proof. Let $\mathcal{A}$ denote an algorithm that computes the width-w Tribes function accurately. Intuitively, algorithm $\mathcal{A}$ needs to look at at least one variable in a constant fraction of the 2^w tribes. Then, we can transform algorithm $\mathcal{A}$ into a new one (denoted by $\mathcal{A}'$) for the AND instance (Definition 38) over w variables, so that $\mathcal{A}'$ reveals one of the w variables with probability $\Omega(1)$. Furthermore, the expected cost of $\mathcal{A}'$ is only a 2^{-w}-fraction of that of $\mathcal{A}$. Finally, we extend our proof of Proposition 39 to show that $\mathcal{A}'$ must have an expected cost of $\Omega(w)$. This would then lower bound the expected cost of $\mathcal{A}$ by $2^w \cdot \Omega(w) = \Omega(2^w \cdot w)$.

We start by formally defining an algorithm *"revealing a tribe"* in the context of computing the Tribes function.

Definition 42 (Revealing a tribe)**.** When an algorithm runs on a Tribes instance (Definition 40), we say that the algorithm reveals the i-th tribe ($i \in [2^w]$) if at least one of the variables $x_{i,1}, x_{i,2}, \ldots, x_{i,w}$ is revealed to the algorithm.

Theorem 41 follows from the three lemmas below, which we prove in the remainder of this section.

Lemma 43. *For every $\varepsilon_0 \in [0, 1/4)$ and integer $w \geq 1$, any ε_0-error algorithm computing the width-w* Tribes *function must reveal $\Omega(2^w)$ tribes in expectation.*

Lemma 44. *Suppose that for some $\alpha, C > 0$, an algorithm for the* Tribes *instance with width w: (1) reveals at least $\alpha \cdot 2^w$ tribes in expectation; and: (2) has an expected cost of C. Then, there is an algorithm for the* AND *instance with w variables that: (1) reveals at least one of the variables with probability α; and: (2) has an expected cost of $C/2^w$.*

Lemma 45. *For any $p \in (0, 1]$, there exists $\alpha > 0$ such that the following is true for every integer $n \geq 1$: If an algorithm for the* AND *instance with n variables reveals at least one of the variables with probability at least p, the expected cost of the algorithm must be at least $\alpha \cdot n$.*

Proof of Theorem 41 assuming Lemmas 43 to 45. We first show that $\mathrm{opt}_0^{\mathrm{avg}} = O(2^w)$. Consider the algorithm that evaluates the 2^w tribes one by one. For each tribe, the algorithm queries the w variables in increasing order of costs and stops whenever a zero is revealed. The probability of querying the cost-i variable is $2^{-(i-1)}$, so the expected cost on each tribe is

$$\sum_{i=1}^{w} 2^{-(i-1)} \cdot i \leq 4.$$

This implies $\mathrm{opt}_0^{\mathrm{avg}} \leq 4 \cdot 2^w = O(2^w)$.

For the lower bound part, let $\mathcal{A}$ be an ε_0-error algorithm computing the width-w Tribes instance with an expected cost of C. By Lemma 43, for some $p > 0$, $\mathcal{A}$ reveals at least $p \cdot 2^w$ tribes in expectation. Then, by Lemma 44, there is an algorithm for the w-variable AND instance that reveals at least one of the w variables with probability p and has an expected cost of $C/2^w$. By Lemma 45, we must have $C/2^w \geq \Omega(w)$, which implies $C = \Omega(2^w \cdot w)$. □

6.3 Proof of Lemma 43

We start with Lemma 43, which states that any algorithm that computes the Tribes function with a low error must reveal a constant fraction of the tribes.

Lemma 43. *For every $\varepsilon_0 \in [0, 1/4)$ and integer $w \geq 1$, any ε_0-error algorithm computing the width-w* Tribes *function must reveal $\Omega(2^w)$ tribes in expectation.*

Note that directly applying the OSSS inequality (Theorem 10) would give a lower bound only on the number of revealed *variables*, and not on the number of revealed *tribes*. Our workaround is to transform an algorithm $\mathcal{A}$—which reveals only a few tribes—into a "canonical form" algorithm $\mathcal{A}'$, so that the number of revealed variables in $\mathcal{A}'$ is comparable to that of revealed tribes in $\mathcal{A}$. Furthermore, $\mathcal{A}'$ has the same or smaller error probability. Then, applying the OSSS inequality to $\mathcal{A}'$ would give the desired lower bound on the number of tribes revealed by $\mathcal{A}$.

Proof. Suppose that algorithm $\mathcal{A}$ solves the width-w Tribes instance with an error probability $\leq \varepsilon_0$ while revealing m tribes in expectation. We will first derive another algorithm $\mathcal{A}'$ that computes Tribes up to an error probability $\leq \varepsilon_0$ and queries at most $2m$ variables in expectation. Then, we apply the OSSS inequality to show that $2m \geq \Omega(2^w)$, which implies the desired lower bound $m \geq \Omega(2^w)$.

Translating number of tribes to query complexity. We will construct an alternative algorithm $\mathcal{A}'$ for Tribes. $\mathcal{A}'$ works in a setting where the variables are not associated with costs, i.e., only the number of queries matters. Furthermore, our construction ensures that $\mathcal{A}'$ satisfies the following two properties:

- **Property 1.** When $\mathcal{A}'$ terminates, it computes the conditional expectation of f given the revealed variables, and outputs the value rounded to $\{0,1\}$. Formally, letting π denote the restriction formed by the observed variables, the output of $\mathcal{A}'$ is always $\mathbf{1}\left\{\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_\pi(\boldsymbol{x})\right] \geq 1/2\right\}$.
- **Property 2.** $\mathcal{A}'$ never queries a variable $x_{i,j}$ if, for some $j' \in [w]$, $x_{i,j'}$ is already queried and known to take value 0.

Intuitively, Property 1 requires that the algorithm always makes the Bayes-optimal prediction. Property 2 prevents the algorithm from making an unnecessary query to a variable $x_{i,j}$ in the i-th tribe when that tribe is known to take value 0 (due to the previous revelation of $x_{i,j'} = 0$).

Formally, $\mathcal{A}'$ is defined as follows:

- $\mathcal{A}'$ simulates algorithm $\mathcal{A}$ on a width-w Tribes instance in Definition 40 by randomly choosing the costs $(c_{i,j})_{i\in[2^w],j\in[w]}$.
- Whenever $\mathcal{A}$ reveals a variable $x_{i,j}$, $\mathcal{A}'$ checks whether there exists another variable $x_{i,j'}$ in the same tribe that has already been revealed as a zero. If so, $\mathcal{A}'$ draws a random bit and feeds that random bit to $\mathcal{A}$ as the value of $x_{i,j}$; otherwise, $\mathcal{A}'$ queries $x_{i,j}$ and forwards the value to $\mathcal{A}$.
- When $\mathcal{A}$ decides to terminate and output an answer, $\mathcal{A}'$ outputs an answer according to Property 1.

In the following, we show that $\mathcal{A}'$ queries $\leq 2m$ variables in expectation, and has an error probability $\leq \varepsilon_0$.

Upper bound the number of queries. We first analyze the expected number of queries made by $\mathcal{A}'$. Fix $i \in [2^w]$. We note that, conditioning on the event that $\mathcal{A}$ reveals the i-th tribe, the conditional distribution of $(x_{i,1}, x_{i,2}, \ldots, x_{i,w})$ is still uniform over $\{0,1\}^w$. Then, by our construction of $\mathcal{A}'$, $\mathcal{A}'$ queries at least j variables in the i-th tribe only if the first $j-1$ variables that get revealed are all ones, which happens with probability $2^{-(j-1)}$. Formally, we have

$$\mathbf{Pr}\left[\mathcal{A}' \text{ queries at least } j \text{ variables in tribe } i\right] \leq \mathbf{Pr}\left[\mathcal{A} \text{ reveals tribe } i\right] \cdot 2^{-(j-1)}.$$

The expected number of variables in the i-th tribe queried by $\mathcal{A}'$ is then given by

$$\begin{aligned}
&\mathbf{E}\left[\sum_{j=1}^{w} \mathbf{1}\left\{\mathcal{A}' \text{ queries at least } j \text{ variables in tribe } i\right\}\right] \\
&= \sum_{j=1}^{w} \mathbf{Pr}\left[\mathcal{A}' \text{ queries at least } j \text{ variables in tribe } i\right] \\
&\leq \sum_{j=1}^{w} \mathbf{Pr}\left[\mathcal{A} \text{ reveals tribe } i\right] \cdot 2^{-(j-1)} \\
&\leq 2\,\mathbf{Pr}\left[\mathcal{A} \text{ reveals tribe } i\right].
\end{aligned}$$

Therefore, the expected number of queries made by $\mathcal{A}'$ is upper bounded by

$$2\sum_{i=1}^{2^w}\mathbf{Pr}\left[\mathcal{A}\text{ reveals tribe }i\right]=2\,\mathbf{E}\left[\sum_{i=1}^{2^w}\mathbf{1}\left\{\mathcal{A}\text{ reveals tribe }i\right\}\right]=2m.$$

Upper bound the error probability. Next, we show that the error probability of $\mathcal{A}'$ is at most ε_0. To this end, we consider the following modified version of $\mathcal{A}'$, denoted by $\mathcal{A}''$:

- $\mathcal{A}''$ simulates $\mathcal{A}$ almost in the same way as $\mathcal{A}'$ does. The only change is that, whenever $\mathcal{A}$ reveals a variable $x_{i,j}$, $\mathcal{A}''$ always queries that variable and forwards its value to $\mathcal{A}$. (In contrast, $\mathcal{A}'$ would feed a random bit to $\mathcal{A}$ in some cases.)
- When $\mathcal{A}$ terminates, $\mathcal{A}''$ outputs an answer according to Property 1, i.e., it outputs the conditional expectation of f (given the observed inputs) rounded to $\{0,1\}$.

Compared with algorithm $\mathcal{A}'$, $\mathcal{A}''$ satisfies Property 1 but not Property 2.

We first note that the error probability of $\mathcal{A}''$ is never higher than that of $\mathcal{A}$. When $\mathcal{A}$ terminates, $\mathcal{A}''$ and $\mathcal{A}$ have observed the same subset of variables, which induce the same restriction $\boldsymbol{\pi}$. Conditioning on this event, all the unobserved variables are still uniformly distributed. Then, predicting 0 leads to a conditional error probability of $\mathbf{Pr}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_{\boldsymbol{\pi}}(\boldsymbol{x})\neq 0\right]=\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_{\boldsymbol{\pi}}(\boldsymbol{x})\right]$, while predicting 1 leads to a conditional error of $1-\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_{\boldsymbol{\pi}}(\boldsymbol{x})\right]$. Then, by predicting $\mathbf{1}\left\{\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_{\boldsymbol{\pi}}(\boldsymbol{x})\right]\geq 1/2\right\}$, algorithm $\mathcal{A}''$ always has a lower or equal conditional error probability than $\mathcal{A}$ does. Applying the law of total probability shows $\mathrm{error}_{\mathcal{A}''}(f)\leq\mathrm{error}_{\mathcal{A}}(f)$.

It remains to show that $\mathrm{error}_{\mathcal{A}'}(f)\leq\mathrm{error}_{\mathcal{A}''}(f)$. We will show that these two sides are equal by coupling the two algorithms carefully. Suppose that both $\mathcal{A}'$ and $\mathcal{A}''$ simulate $\mathcal{A}$ with the same randomness in $\mathcal{A}$ and in the costs. Furthermore, we "defer" the randomness in the input $\boldsymbol{x}\in\{0,1\}^n$ by realizing each input bit only when it gets queried by an algorithm. Whenever $\mathcal{A}$ reveals a variable $x_{i,j}$, there are two cases:

- **Case 1: Some $x_{i,j'}=0$ in the same tribe is already revealed.** Recall that $\mathcal{A}'$ would feed a random bit to $\mathcal{A}$ in this case, while $\mathcal{A}''$ would query $x_{i,j}$ and feed the actual value to $\mathcal{A}$. Note that, before $\mathcal{A}''$ queries $x_{i,j}$, the input bit is uniformly distributed among $\{0,1\}$. Thus, we may couple the two executions, so that the random bit chosen by $\mathcal{A}'$ is always equal to the random realization of $x_{i,j}$ in $\mathcal{A}''$.
- **Case 2: No variable $x_{i,j'}$ is revealed to be zero.** In this case, both $\mathcal{A}'$ and $\mathcal{A}''$ would actually query $x_{i,j}$. We couple the two executions such that the realization of $x_{i,j}$ are the same.

Note that this coupling ensures that $\mathcal{A}$ follows the same computation path in both $\mathcal{A}'$ and $\mathcal{A}''$.

At the end of the simulations, let $\boldsymbol{X}',\boldsymbol{X}''\subseteq[2^w]\times[w]$ denote the indices of the variables that are queried by $\mathcal{A}'$ and $\mathcal{A}''$, respectively. By our coupling, we always have $\boldsymbol{X}'\subseteq\boldsymbol{X}''$. Let $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$ denote the restrictions naturally induced by the variables observed by $\mathcal{A}'$ and $\mathcal{A}''$, respectively. Note that $\mathcal{A}'$ and $\mathcal{A}''$ output $\mathbf{1}\left\{\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_{\boldsymbol{\pi}'}(\boldsymbol{x})\right]\geq 1/2\right\}$ and $\mathbf{1}\left\{\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_{\boldsymbol{\pi}''}(\boldsymbol{x})\right]\geq 1/2\right\}$, respectively.

We then couple the realization of the remaining randomness in $\boldsymbol{x}$ such that, for every $(i,j)\in([2^w]\times[w])\setminus\boldsymbol{X}''$, the realization of $x_{i,j}$ is the same for both $\mathcal{A}'$ and $\mathcal{A}''$. At this point, the input $\boldsymbol{x}$ is determined in both simulations, and we let $\boldsymbol{x}'$ and $\boldsymbol{x}''$ denote the two realizations for clarity. We will verify the following two facts:

- $f(\boldsymbol{x}') = f(\boldsymbol{x}'')$, i.e., $f(\boldsymbol{x})$ takes the same value in both $\mathcal{A}'$ and $\mathcal{A}''$.
- The outputs of $\mathcal{A}'$ and $\mathcal{A}''$ are equal.

Assuming the above, we immediately have $\mathrm{error}_{\mathcal{A}'}(f) = \mathrm{error}_{\mathcal{A}''}(f)$, since

$$\mathrm{error}_{\mathcal{A}'}(f) = \mathbf{Pr}\left[f(\boldsymbol{x}') \neq \text{output of } \mathcal{A}'\right] \quad \text{and} \quad \mathrm{error}_{\mathcal{A}''}(f) = \mathbf{Pr}\left[f(\boldsymbol{x}'') \neq \text{output of } \mathcal{A}''\right].$$

To verify the first fact, we note that $\boldsymbol{x}'$ and $\boldsymbol{x}''$ might differ only on coordinates $(i,j) \in \boldsymbol{X}'' \setminus \boldsymbol{X}'$. By definition of $\mathcal{A}'$, for every such coordinate (i,j), there is another variable in the i-th tribe that was revealed as a zero in both $\mathcal{A}'$ and $\mathcal{A}''$. Formally, there must exist $j' \in [w]$ such that $x'_{i,j'} = x''_{i,j} = 0$. Then, the difference $x'_{i,j} \neq x''_{i,j}$ would not matter, since the i-th tribe would take value 0 in either case.

To verify the second fact, for $\boldsymbol{x} \in \{0,1\}^n = \{0,1\}^{2^w \cdot w}$, let $g_i(\boldsymbol{x}) \coloneqq \bigwedge_{j=1}^{w} x_{i,j}$ denote the i-th tribe. For $x \in \{0,1\}^n$ and restriction π, we write $x \triangleleft \pi$ if x is consistent with the restriction. Then, for any generic restriction π, we have

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f_\pi(\boldsymbol{x})\right] = 1 - \mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f(\boldsymbol{x}) = 0 \mid \boldsymbol{x} \triangleleft \pi\right] = 1 - \prod_{i=1}^{2^w}\left[1 - \mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[g_i(\boldsymbol{x}) = 1 \mid \boldsymbol{x} \triangleleft \pi\right]\right].$$

The second step above holds since, after conditioning on $\boldsymbol{x} \triangleleft \pi$, $\boldsymbol{x}$ always follows a product distribution.

Therefore, it suffices to verify that, for every pair of restrictions $(\boldsymbol{\pi}', \boldsymbol{\pi}'')$ and every $i \in [2^w]$, it holds that

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[g_i(\boldsymbol{x}) = 1 \mid \boldsymbol{x} \triangleleft \boldsymbol{\pi}'\right] = \mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[g_i(\boldsymbol{x}) = 1 \mid \boldsymbol{x} \triangleleft \boldsymbol{\pi}''\right].$$

By our definition of $\mathcal{A}'$ and $\mathcal{A}''$, there are two possible cases:

- **Case 1: $\boldsymbol{X}'$ and $\boldsymbol{X}''$ agree on the i-th tribe.** In this case, $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$ induce the same restriction on variables $x_{i,1}, x_{i,2}, \ldots, x_{i,w}$, so we have

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[g_i(\boldsymbol{x}) = 1 \mid \boldsymbol{x} \triangleleft \boldsymbol{\pi}'\right] = \mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[g_i(\boldsymbol{x}) = 1 \mid \boldsymbol{x} \triangleleft \boldsymbol{\pi}''\right].$$

- **Case 2: For some $j \in [w]$, $(i,j) \in \boldsymbol{X}'' \setminus \boldsymbol{X}'$.** Again, this can only happen if, for some $j' \in [w]$, both $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$ contain the restriction $x_{i,j'} = 0$, in which case we have

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[g_i(\boldsymbol{x}) = 1 \mid \boldsymbol{x} \triangleleft \boldsymbol{\pi}'\right] = \mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[g_i(\boldsymbol{x}) = 1 \mid \boldsymbol{x} \triangleleft \boldsymbol{\pi}''\right] = 0.$$

Therefore, we conclude that

$$\mathrm{error}_{\mathcal{A}'}(f) = \mathrm{error}_{\mathcal{A}''}(f) \leq \mathrm{error}_{\mathcal{A}}(f) \leq \varepsilon_0,$$

and $\mathcal{A}'$ queries at most $2m$ variables in expectation.

Lower bound on the query complexity of Tribes**.** This is a consequence of the OSSS inequality (Theorem 10). In the width-w Tribes function f, we have

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[f(\boldsymbol{x}) = 0\right] = \left(1 - \frac{1}{2^w}\right)^{2^w} \in \left[\frac{1}{4}, \frac{1}{e}\right].$$

It follows that $\mathrm{bias}(f) \geq 1/4$. Furthermore, each variable is pivotal if and only if: (1) all the $2^w - 1$ other tribes take value 0; (2) all the $w - 1$ other variables in the same tribe take value 1. Thus, the influence of every variable $x_{i,j}$ is given by

$$\mathbf{Inf}_{i,j}[f] = \left(1 - \frac{1}{2^w}\right)^{2^w - 1} \cdot \frac{1}{2^{w-1}} \leq \frac{1}{2} \cdot \frac{2}{2^w} = \frac{1}{2^w}.$$

Then, for every ε_0-error algorithm $\mathcal{A}$, Theorem 10 gives

$$\frac{1}{4} - \varepsilon_0 \leq \mathrm{bias}(f) - \mathrm{error}_f(\mathcal{A}) \leq \sum_{i=1}^{2^w} \sum_{j=1}^{w} \delta_{i,j}(\mathcal{A}) \cdot \mathbf{Inf}_{i,j}[f] \leq \frac{1}{2^w} \sum_{i=1}^{2^w} \sum_{j=1}^{w} \delta_{i,j}(\mathcal{A}).$$

Therefore, the expected number of queries made by $\mathcal{A}$, $\sum_{i=1}^{2^w} \sum_{j=1}^{w} \delta_{i,j}(\mathcal{A})$, is at least $(1/4 - \varepsilon_0) \cdot 2^w$. Applying this to algorithm $\mathcal{A}'$ constructed above gives $m \geq \frac{1/4 - \varepsilon_0}{2} \cdot 2^w = \Omega(2^w)$. □

6.4 Proof of Lemma 44

Next, we turn to Lemma 44, which transforms an algorithm computing the width-w Tribes function into one for the w-variable AND function. This is done by "planting" the AND instance as one of the 2^w tribes in the Tribes instance. The transformation ensures that the resulting algorithm reveals at least one of the w variables with a sufficiently high probability, while incurring a low cost in expectation.

Lemma 44. *Suppose that for some $\alpha, C > 0$, an algorithm for the* Tribes *instance with width w: (1) reveals at least $\alpha \cdot 2^w$ tribes in expectation; and: (2) has an expected cost of C. Then, there is an algorithm for the* AND *instance with w variables that: (1) reveals at least one of the variables with probability α; and: (2) has an expected cost of $C/2^w$.*

Proof. Let $\mathcal{A}$ denote the algorithm for width-w Tribes. In the following, we construct another algorithm, denoted by $\mathcal{A}'$, for the w-variable AND function.

- Draw $i^* \in [2^w]$ uniformly at random.
- Generate a width-w Tribes instance by sampling, independently for each $i \in [2^w] \setminus \{i^*\}$, a uniformly random permutation $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ of $[w]$, and input bits $(x_{i,1}, x_{i,2}, \ldots, x_{i,w}) \in \{0,1\}^w$.
- Simulate $\mathcal{A}$ on a width-w Tribes instance with the i^*-th tribe being the actual AND instance. Formally, when $\mathcal{A}$ increments the investment $\theta_{i,j}$ for some $i \in [2^w] \setminus \{i^*\}$ and $j \in [w]$, $\mathcal{A}'$ checks whether $\theta_{i,j} \geq c_{i,j}$ holds. If so, $\mathcal{A}'$ reveals the value of $x_{i,j}$ to $\mathcal{A}$. When $\mathcal{A}$ increments $\theta_{i^*,j}$ for some $j \in [w]$, $\mathcal{A}'$ increases the investment in the j-th variable to $\theta_{i^*,j}$ in the AND instance (that $\mathcal{A}'$ is solving).

By construction of $\mathcal{A}'$, $\mathcal{A}$ runs on a random width-w Tribes instance. For each $i \in [2^w]$, let p_i denote the probability that the i-th tribe is revealed by $\mathcal{A}$, and let C_i denote the expected cost that $\mathcal{A}$ invests in the w variables in the i-th tribe. Note that, conditioning on the event that $i^* = i$, the probability that algorithm $\mathcal{A}'$ reveals at least one of the w variables (in the AND instance) is exactly p_i. Furthermore, the expected cost incurred by $\mathcal{A}'$ (in the AND instance) is exactly C_i.

By our assumptions on $\mathcal{A}$ and the linearity of expectation, we have

$$\sum_{i=1}^{2^w} p_i \geq \alpha \cdot 2^w \quad \text{and} \quad \sum_{i=1}^{2^w} C_i \leq C.$$

Since i^* is drawn uniformly at random from $[2^w]$, the overall probability that $\mathcal{A}'$ reveals at least one of the w variables is

$$\frac{1}{2^w}\sum_{i=1}^{2^w} p_i \geq 2^{-w} \cdot (\alpha \cdot 2^w) = \alpha,$$

and the expected cost of $\mathcal{A}'$ is

$$\frac{1}{2^w}\sum_{i=1}^{2^w} C_i = C/2^w.$$

Therefore, $\mathcal{A}'$ would be the algorithm that the lemma asks for. □

6.5 Proof of Lemma 45

Lemma 45. *For any $p \in (0,1]$, there exists $\alpha > 0$ such that the following is true for every integer $n \geq 1$: If an algorithm for the* AND *instance with n variables reveals at least one of the variables with probability at least p, the expected cost of the algorithm must be at least $\alpha \cdot n$.*

The proof follows the same idea as the one for Proposition 39: We simulate the algorithm on an AND instance in which every cost is high, so that none of the variables can be revealed. Then, we argue that the simulation coincides with the execution of the algorithm on an actual instance from Definition 38. The proof needs to be slightly modified for two reasons: (1) we can no longer assume that the algorithm is deterministic; (2) The algorithm is only assumed to reveal a variable with probability $\Omega(1)$, rather than having a zero error for computing the AND function.

Proof. Let $\mathcal{A}$ denote the hypothetical algorithm for the AND instance. Set $m = \frac{p}{2} \cdot n$. We simulate $\mathcal{A}$ on an alternative instance—in which the function is the AND over n variables, each of which has a cost of $c_i = n$—until either one of the following two happens:

- **Case 1:** $\mathcal{A}$ terminates voluntarily.
- **Case 2:** $\mathcal{A}$ attempts to increase the total investment beyond m.

Let $\mathcal{A}^{(1)}$ denote the simulation above. For each $i \in [n]$, let a_i denote the total investment in variable x_i when $\mathcal{A}^{(1)}$ terminates. Since $\mathcal{A}$ can be randomized in general, $a_1, a_2, \ldots, a_n$ are random variables that depend on the internal randomness of $\mathcal{A}^{(1)}$. Nevertheless, it always holds that $\sum_{i=1}^n a_i \leq m$; otherwise the simulation should have been terminated earlier by Case 2. Furthermore, since $m = (p/2) \cdot n \leq n/2 < n$, none of the n variables gets revealed in simulation $\mathcal{A}^{(1)}$.

Let $\mathcal{A}^{(2)}$ denote the simulation of $\mathcal{A}$—using the same internal randomness as in $\mathcal{A}^{(1)}$—on an actual AND instance from Definition 38. Then, the two simulations $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ agree except with probability at most

$$\sum_{i=1}^{n} \mathop{\mathbf{Pr}}_{c}[c_i \leq a_i] \leq \sum_{i=1}^{n} \frac{a_i}{n} \leq \frac{m}{n} = \frac{p}{2}.$$

Moreover, since $\mathcal{A}^{(2)}$ actually runs on an AND instance from Definition 38, our assumption on $\mathcal{A}^{(2)}$ implies that it reveals at least one variable with probability at least p.

Let E_1 denote the event that $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ agree, and E_2 denote the event that $\mathcal{A}^{(2)}$ eventually reveals a variable. The analysis above gives

$$\mathbf{Pr}\left[\overline{E_1}\right] \leq \frac{p}{2} \quad \text{and} \quad \mathbf{Pr}[E_2] \geq p,$$

which implies

$$\mathbf{Pr}\left[E_1 \wedge E_2\right] = \mathbf{Pr}\left[E_2\right] - \mathbf{Pr}\left[E_2 \wedge \overline{E_1}\right] \geq \mathbf{Pr}\left[E_2\right] - \mathbf{Pr}\left[\overline{E_1}\right] \geq p - \frac{p}{2} = \frac{p}{2}.$$

By our definition of the simulations, if both E_1 and E_2 happen, $\mathcal{A}^{(2)}$ must reveal one of the n variables after the total investment reaches m. Therefore, the expected cost of $\mathcal{A}^{(2)}$ is at least

$$\mathbf{Pr}\left[E_1 \wedge E_2\right] \cdot m \geq \frac{p}{2} \cdot \frac{p}{2} n = \frac{p^2}{4} \cdot n.$$

In other words, the claim holds for $\alpha = p^2/4$. □

Acknowledgements

R.R. is supported by the NSF TRIPODS program (award DMS-2022448) and CCF-2310818.

References

[AHKÜ17] Sarah R Allen, Lisa Hellerstein, Devorah Kletenik, and Tonguç Ünlüyurt. Evaluation of monotone DNF formulas. *Algorithmica*, 77:661–685, 2017. 1, 6, 7

[BDHK18] Eric Bach, Jérémie Dusart, Lisa Hellerstein, and Devorah Kletenik. Submodular goal value of Boolean functions. *Discrete Applied Mathematics*, 238:1–13, 2018. 6

[BLQT22] Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. *Journal of the ACM*, 69(6):1–19, 2022. 10, 28, 30, 31

[BLT21] Guy Blanc, Jane Lange, and Li-Yang Tan. Query strategies for priced information, revisited. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1638–1650. SIAM, 2021. 1, 2, 3, 4, 5, 6, 8, 9, 12, 14, 20

[BOL85] M. Ben-Or and N. Linial. Collective coin flipping. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 408–416, 1985. 3

[Bop97] Ravi B. Boppana. The average sensitivity of bounded-depth circuits. *Inf. Process. Lett.*, 63(5):257–261, 1997. 4

[BSU17] Mark Bun, Thomas Steinke, and Jonathan Ullman. Make up your mind: The price of online queries in differential privacy. In *Proceedings of the twenty-eighth annual ACM-SIAM symposium on discrete algorithms*, pages 1306–1325. SIAM, 2017. 6

[BT96] Nader Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996. 4

[CFG+00] Moses Charikar, Ronald Fagin, Venkatesan Guruswami, Jon Kleinberg, Prabhakar Raghavan, and Amit Sahai. Query strategies for priced information (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, page 582–591, New York, NY, USA, 2000. Association for Computing Machinery. 1, 2, 6, 7

[CGLM11] Ferdinando Cicalese, Travis Gagie, Eduardo Laber, and Martin Milanič. Competitive boolean function evaluation: Beyond monotonicity, and the symmetric case. *Discrete applied mathematics*, 159(11):1070–1078, 2011. 1, 6

[CGT+20] Shuchi Chawla, Evangelia Gergatsouli, Yifeng Teng, Christos Tzamos, and Ruimin Zhang. Pandora's box with correlations: Learning and approximation. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1214–1225. IEEE, 2020. 7

[CL05a] Ferdinando Cicalese and Eduardo Sany Laber. A new strategy for querying priced information. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 674–683, New York, NY, USA, 2005. Association for Computing Machinery. 1, 6

[CL05b] Ferdinando Cicalese and Eduardo Sany Laber. An optimal algorithm for querying priced information: Monotone boolean functions and game trees. In Gerth Stølting Brodal and Stefano Leonardi, editors, *Algorithms – ESA 2005*, pages 664–676, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. 1, 6, 7

[CL08] Ferdinando Cicalese and Eduardo Sany Laber. Function evaluation via linear programming in the priced information model. In *Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I 35*, pages 173–185. Springer, 2008. 1, 6

[CL11] Ferdinando Cicalese and Eduardo Sany Laber. On the competitive ratio of evaluating priced functions. *Journal of the ACM (JACM)*, 58(3):1–40, 2011. 1, 6

[CLN+24] Edith Cohen, Xin Lyu, Jelani Nelson, Tamás Sarlós, and Uri Stemmer. Lower bounds for differential privacy under continual observation and online threshold queries. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1200–1222. PMLR, 2024. 6

[CM11] Ferdinando Cicalese and Martin Milanič. Competitive evaluation of threshold functions in the priced information model. *Annals of Operations Research*, 188(1):111–132, 2011. 1, 6

[DHK14] Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation Algorithms for Stochastic Boolean Function Evaluation and Stochastic Submodular Set Cover. In *Proceedings of the twenty-fifth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1453–1466. SIAM, 2014. 1, 6, 7, 12

[DNS23] Marina Drygala, Sai Ganesh Nagarajan, and Ola Svensson. Online algorithms with costly predictions. In *International Conference on Artificial Intelligence and Statistics*, pages 8078–8101. PMLR, 2023. 7

[GGHK18] Dimitrios Gkenosis, Nathaniel Grammel, Lisa Hellerstein, and Devorah Kletenik. The stochastic score classification problem. In *26th European Symposium on Algorithms, ESA 2018*. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2018. 1, 5, 7, 26

[GGN24] Rohan Ghuge, Anupam Gupta, and Viswanath Nagarajan. Nonadaptive stochastic score classification and explainable half-space evaluation. *Operations Research*, 2024. 1, 6, 7

[GHKL22] Nathaniel Grammel, Lisa Hellerstein, Devorah Kletenik, and Naifeng Liu. Algorithms for the unit-cost stochastic score classification problem. *Algorithmica*, 84(10):3054–3074, 2022. 6

[GJSS19] Anupam Gupta, Haotian Jiang, Ziv Scully, and Sahil Singla. The markovian price of information. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 233–246. Springer, 2019. 7

[GK01] Anupam Gupta and Amit Kumar. Sorting and selection with structured costs. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 416–425. IEEE, 2001. 1, 6

[GS14] Christophe Garban and Jeffrey E Steif. *Noise sensitivity of Boolean functions and percolation*, volume 5. Cambridge University Press, 2014. 4

[Hel18] Lisa Hellerstein. Stochastic Evaluation of Symmetric Boolean Functions. In *ISAIM*, 2018. 1, 5, 6, 7, 26

[HKLW22] Lisa Hellerstein, Devorah Kletenik, Naifeng Liu, and R Teal Witter. Adaptivity Gaps for the Stochastic Boolean Function Evaluation Problem. In *International Workshop on Approximation and Online Algorithms*, pages 190–210. Springer, 2022. 1, 5, 6, 26

[HLS24] Lisa Hellerstein, Naifeng Liu, and Kevin Schewior. Quickly Determining Who Won an Election. In *15th Innovations in Theoretical Computer Science Conference, ITCS 2024*, page 61. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2024. 1, 5, 6, 7, 26

[Juk12] S. Jukna. *Boolean Function Complexity: Advances and Frontiers:*. Springer, 2012. 4

[JZ11] Rahul Jain and Shengyu Zhang. The influence lower bound via query elimination. *Theory of Computing*, 7(10):147–153, 2011. 12

[Kal04] G. Kalai. Social indeterminacy. *Econometrica*, 72(5):1565–1581, 2004. 4

[Kan14] D. M. Kane. The average sensitivity of an intersection of half spaces. In *Symposium on Theory of Computing, STOC 2014*, pages 437–440, 2014. 4

[KK03] Sampath Kannan and Sanjeev Khanna. Selection with monotone comparison costs. In *Symposium on Discrete Algorithms: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, volume 12, pages 10–17, 2003. 1, 6

[KKL88] J. Kahn, G. Kalai, and N. Linial. The influence of variables on Boolean functions. In *Proc. 29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 68–80, 1988. 3

[KKM05] Haim Kaplan, Eyal Kushilevitz, and Yishay Mansour. Learning with attribute costs. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 356–365, New York, NY, USA, 2005. Association for Computing Machinery. 1, 6, 7

[LMN93] Nati Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993. 4

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. 3, 4, 11, 12

[OSSS05] Ryan O'Donnell, Michael Saks, Oded Schramm, and Rocco A Servedio. Every decision tree has an influential variable. In *Proc. 46th Symposium on Foundations of Computer Science (FOCS)*, pages 31–39, 2005. 12

[QV23] Mingda Qiao and Gregory Valiant. Online Pen Testing. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, volume 251 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 91:1–91:26, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 7

[Riv87] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987. 18

[Sin18] Sahil Singla. The price of information in combinatorial optimization. In *Proceedings of the twenty-ninth annual ACM-SIAM symposium on discrete algorithms*, pages 2523–2532. SIAM, 2018. 1, 7

[Ünl04] Tonguç Ünlüyurt. Sequential testing of complex systems: a review. *Discrete Applied Mathematics*, 142(1-3):189–205, 2004. 1

[Ver18] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*, volume 47. Cambridge University Press, 2018. 23

[Wei78] Martin Weitzman. *Optimal search for the best alternative*, volume 78. Department of Energy, 1978. 7

[Wil91] David Williams. *Probability with Martingales*. Cambridge University Press, 1991. 15