An Exact Algorithm for the Unanimous Vote Problem*

Feyza Duman Keles† Lisa Hellerstein† Kunal Marwaha‡ Christopher Musco†
Xinchen Yang§

Abstract. Consider n independent, biased coins, each with a known probability of heads. Presented with an ordering of these coins, flip (i.e., toss) each coin once, in that order, until we have observed both a *head* and a *tail*, or flipped all coins. The Unanimous Vote problem asks us to find the ordering that minimizes the expected number of flips. Gkenosis et al. [GGHK18] gave a polynomial-time ϕ-approximation algorithm for this problem, where $\phi \approx 1.618$ is the golden ratio. They left open whether the problem was NP-hard. We answer this question by giving an exact algorithm that runs in time $O(n \log n)$. The Unanimous Vote problem is an instance of the more general Stochastic Boolean Function Evaluation problem: it thus becomes one of the only such problems known to be solvable in polynomial time. Our proof uses simple interchange arguments to show that the optimal ordering must be close to the ordering produced by a natural greedy algorithm. Beyond our main result, we compare the optimal ordering with the best adaptive strategy, proving a tight adaptivity gap of $1.2 \pm o(1)$ for the Unanimous Vote problem.

1 Introduction. Suppose you have n independent biased coins with probability of heads $p_1, \ldots, p_n \in [0, 1]$. You arrange the coins in some order, and then flip (that is, toss) each coin, in that order, until either you have observed a *head* or you have flipped all coins. In what order should you flip the coins to minimize your expected number of flips? The answer is obvious: flip the coins in decreasing order of their p_i values.

Next, consider a slight variation of this problem, where you flip coins until you have observed *both* a head and a tail, or until you have flipped all coins. Again, in what order should you flip the coins to minimize your expected number of flips? Now the answer is not obvious at all. This problem, introduced by Gkenosis et al. [GGHK18], is the one we consider in this paper. Viewing the coin flips as yes/no votes of n stochastic voters, Gkenosis et al. framed the problem as seeking to determine whether the n votes are unanimously yes, unanimously no, or not unanimous. We call it the *Unanimous Vote problem*.

Gkenosis et al. gave a simple *2-approximation* algorithm for the problem, which produces an ordering whose expected number of coin flips is at most *twice* the optimal [GGHK18]. They also gave a more involved ϕ-approximation algorithm for the problem, where $\phi \approx 1.618$ is the golden ratio. They left as an open question whether the Unanimous Vote problem is NP-hard. Indeed, this problem is a *non-adaptive* Stochastic Boolean Function Evaluation (SBFE) problem (see e.g., [IK84, KBZ86, HKLW22a, GHKL22, GGN24, HNT25, Ünl25, NRS25]). Very few of these problems are known to be solvable exactly in polynomial time.

Our main contribution is a simple, exact, $O(n \log n)$-time algorithm for the Unanimous Vote problem, which resolves the question of Gkenosis et al. We obtain our result by showing that the structure of an optimal ordering must be very similar to a natural *greedy* solution to the problem. To produce the optimal ordering, our algorithm constructs this greedy solution, and rearranges the positions of at most two coins.

In addition to our main result, we prove that the greedy solution itself is near optimal, flipping at most one more coin in expectation than the optimal ordering. We also prove a result on *adaptive strategies*, which can choose the next coin based on the outcomes of previous flips. For the Unanimous Vote problem, there is a simple polynomial time optimal adaptive strategy [GGHK18]. The algorithm presented in this paper, in contrast, is *non-adaptive*. In general, the non-adaptive ordering for an SBFE problem requires at least as many flips as the best adaptive strategy, and there has been interest in quantifying the benefit of adaptivity [HKLW22b, GHKL22].

*The full version of the paper can be accessed at https://arxiv.org/pdf/2510.16678

†New York University (fd2135@nyu.edu, lisa.hellerstein@nyu.edu, cmusco@nyu.edu).
‡University of Chicago (kmarw@uchicago.edu).
§University of Maryland (xcyang@cs.umd.edu).

To that end, in the full version of this paper (available at [DHM+25]), we prove that the *adaptivity gap*, i.e., the worst possible ratio of the expected number of flips of the optimal non-adaptive ordering to the expected number of flips of the optimal adaptive strategy, is $1.2 \pm o(1)$. In this conference version of the paper, we provide a shorter and simpler proof that it lies between $1.2 - o(1)$ and 1.5.

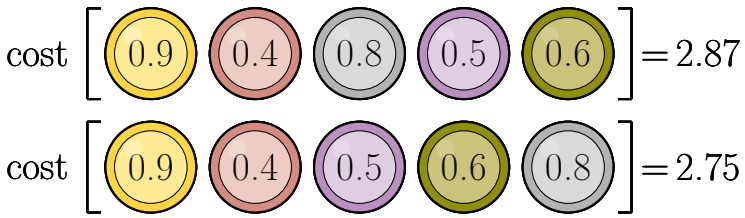

Figure 1.1: Two example orderings for an instance of the Unanimous Vote problem. The second ordering has a smaller number of expected flips (cost). One intuitive strategy is to alternate between the highest and lowest bias coins, as in the first sequence. As we can see, this is not always optimal. The second sequence is the one returned by a natural greedy algorithm described below. For this instance, it happens to be optimal.

1.1 Algorithm overview. The Unanimous Vote problem asks for a fixed ordering of the coins that minimizes the expected number of flips required until both heads and tails are seen, or all coins have been flipped. Figure 1.1 compares two orderings on a small example instance. In Sections 3 and 4 we prove our main result:

THEOREM 1.1. *There is an algorithm (Algorithm 4.2) that solves the Unanimous Vote problem (i.e., computes the minimum cost ordering) in time* $O(n \log n)$.

To understand our approach to Theorem 1.1, it is helpful to define *biased blocks* of an ordering, a concept that we introduce, and which is central to our analysis. To define biased blocks, fix an ordering, and consider some position k. Suppose we flip coins according to that ordering, terminating as soon as we observe a head and a tail, or have flipped all coins. We will only flip the k^{th} coin if the first $k-1$ coins all come up heads, or all come up tails. Using the convention that heads $= 1$, and tails $= 0$, we call the k^{th} position *1-biased* if the probability that the first $k-1$ coins all come up heads is greater than the probability that they all come up tails, *0-biased* if the opposite relationship holds, and *unbiased* if the probabilities are equal.

The n positions in the ordering can be partitioned into "blocks", according to their biases. Each block is a maximal set of contiguous positions of the same bias. See Figure 1.2 for an example.

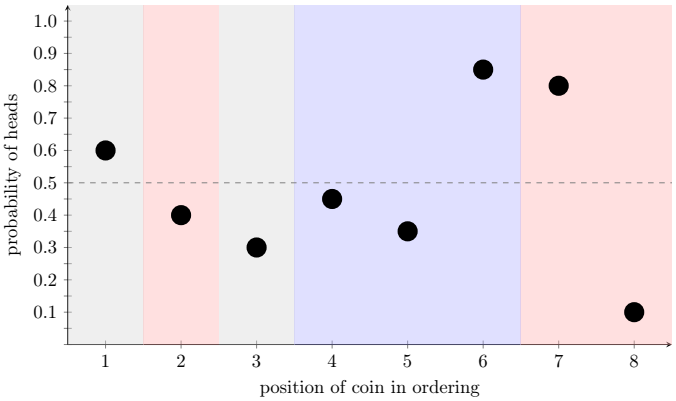

Figure 1.2: Plot of the probability of heads of a coin versus its position in an ordering. Unbiased blocks are in gray, 0-biased blocks are in blue, and 1-biased blocks are in red. For example, the fourth coin in the ordering has probability of heads equal to 0.45 and is in a 0-biased block.

At a high level, our algorithm works by executing the following three steps:

1. Generate an initial ordering of the coins using a greedy rule.
2. Generate $O(n)$ alternative orderings by making small modifications to the initial greedy ordering.
3. Calculate the expected number of flips for each generated ordering and return the best one.

Our initial ordering (formalized in Algorithm 2.1) is constructed greedily from position 1 through position n. The coin in the k^{th} position is chosen from the remaining coins according to the following rule:

Maximize the probability of terminating on the k^{th} flip, assuming that termination has not yet occurred.

We show in Subsection 2.2 that if position k is 1-biased, the greedy rule chooses the remaining coin with smallest p_i value. Symmetrically, if position k is 0-biased, then the greedy rule chooses the coin with largest p_i value. If the position is unbiased (for example, when $k = 1$), *all* coins satisfy the greedy rule. As a convention, our algorithm chooses the remaining coin with the largest p_i value when position k is unbiased.

In general, the greedy ordering is not optimal; see Section 5 for an example. Nonetheless, we prove that, surprisingly, the optimal ordering can be produced by following the greedy rule *except at one special position*: the last position in the second-to-last block. This is the *only* position where it may be better to choose a different coin. With this key fact in place, it is easy to obtain Theorem 1.1: after sorting $p_1, \ldots, p_n$ in $O(n \log n)$ time, our algorithm generates the greedy ordering and all alternatives that result from making a non-greedy choice at that position. There are at most $n - 1$ such alternatives. We can evaluate the expected cost of all of these orderings in linear time and then choose the one with minimum cost.

The key challenge in our work is proving that the optimal ordering is only "non-greedy" at one particular location. To do so, we prove in Sections 3 and 4 that any optimal ordering must obey a set of "monotonicity" properties that hold for the greedy ordering. For example, the first such property we prove is that coins within a single 1-biased (resp. 0-biased) block have increasing (resp. decreasing) probability of heads. Each property is proven using elementary *interchange* arguments, which consider the effect on the expected number of flips if you interchange (swap) a certain pair of coins in an ordering.

1.2 Additional results. Beyond our main algorithmic result, we prove two additional results on the Unanimous Vote problem. First, we show in Section 5 that, even though the greedy ordering is not optimal, it is always *close* to optimal:

Claim 1.2 (Greedy is near-optimal). *For any instance of the Unanimous Vote problem, the greedy ordering (Algorithm* 2.1*) flips at most* 1 *more coin in expectation than the optimal ordering. There is a family of instances where this claim is asymptotically tight.*

Finally, in Section 6 we prove a new bound on the adaptivity gap of the Unanimous Vote problem:

Theorem 1.3 (Bound on adaptivity gap). *The adaptivity gap of the Unanimous Vote problem is at least* $1.2 - o(1)$ *and at most* 1.5.

The family of instances that give the lower bound is simple to describe and analyze: one coin has $p_i = 0$, and the remaining coins all have $p_i = \frac{1}{2}$. The proof of the upper bound analyzes a specific non-adaptive ordering (which is not necessarily optimal) that mimics properties of the optimal adaptive strategy. In the full version of this paper (available at [DHM+25]), we use an extended version of this approach to sharpen the upper bound to $1.2 + o(1)$, which essentially settles the adaptivity gap for the Unanimous Vote problem.

1.3 Related work. The Unanimous Vote problem is a Stochastic Boolean Function Evaluation (SBFE) problem. For a comprehensive survey on SBFE problems, which are also referred to as "sequential testing" problems in the Operations Research literature, we refer the reader to the surveys of Ünlüyurt [Ünl04, Ünl25].

In *unit-cost* SBFE problems, the goal is to *exactly evaluate* a given Boolean function $f : \{0,1\}^n \to \{0,1\}$, while observing as few of the inputs bits, $x_1, \ldots, x_n$, as possible. Each input bit x_i is assumed to be drawn from an independent Bernoulli distribution with given parameter p_i. The problem is to determine the optimal order in which to observe the bits, so as to minimize the expected number of observations.[1]

[1]In arbitrary-cost SBFE problems, there is a cost c_i associated with observing a bit x_i, and the expected total cost of the observations must be minimized. In what follows, any previous result said to hold for the "SBFE problem" rather than for the "unit-cost SBFE problem" should be understood to apply to the arbitrary cost version as well.

SBFE problems are studied in the *adaptive setting*, which allows adaptive strategies, and in the *non-adaptive setting*, where the observation order must be fixed in advance. One motivation for non-adaptive strategies is that they can always be represented compactly as a permutation and deployed efficiently (we simply read the ID of the next bit to reveal). Adaptive strategies typically require computation at each step. The Unanimous Vote problem is equivalent to the unit-cost, non-adaptive SBFE problem when f is the not-all-equal function, i.e., f evaluates to 1 iff $x_i \neq x_j$ for some i, j. Our work shows that this SBFE problem can be solved in polynomial time.

For some other unit-cost, non-adaptive SBFE problems, designing a polynomial-time algorithm is trivial. For example, for the Boolean OR function, the problem is equivalent to the coin-flipping problem mentioned at the start of the introduction, where you flip coins until you see the first occurrence of heads. As another example, consider the parity function: evaluating this function requires observing all bits, so the expected number of observations is the same no matter which ordering you use. Our work appears to be the first to give a polynomial-time algorithm solving a unit-cost, non-adaptive SBFE problem that is *not* trivial.

There is, however, interesting prior work on *approximately* solving non-adaptive SBFE problems. For example, constant-factor approximation algorithms have been studied for a problem called "Stochastic Score Classification" [GGN24, PS24, Liu22, GHKL22]. These methods yield constant factor approximation algorithms for the non-adaptive SBFE problems for any symmetric Boolean function and for any linear threshold function. Recall that a Boolean function is symmetric if its output depends only on the number of 1's in the input, so these results cover the Unanimous Vote problem (the not-all-equal function is symmetric). Before our work, the best previous polynomial time algorithm for the Unanimous Vote problem achieved an approximation factor of $\phi \approx 1.618$ using a method specialized to that problem [GGHK18].

Specialized approximation algorithms have also been developed for the *k-of-n function*, the symmetric Boolean function whose output is 1 iff at least k of its inputs are 1. Recent work gives a PTAS for the unit-cost, non-adaptive SBFE problem for the k-of-n function [NRS25]. There is also a very simple 1.5 approximation algorithm for this problem [GHKL22]. Beyond symmetric functions, an 8-approximation algorithm is known for the unit-cost, non-adaptive SBFE problem for Boolean read-once formulas [HHL22].

Adaptive Methods. In the *adaptive* setting, the Unanimous Vote problem can be easily solved by noting the following: once you have flipped the first coin and observed its outcome, it will be optimal to flip the remaining coins in either increasing or decreasing probability of heads, depending on whether your first flip is heads or tails [GGHK18]. Thus the only difficulty is to determine the first coin. To do so, we can just compute the expected cost associated with each of the n choices of the first coin, and choose the best one.

Other adaptive SBFE problems are more interesting. There is an elegant polynomial time algorithm that solves the adaptive SBFE problem for the k-of-n function [BD81, CSF90, SB97], and also works for the exactly-k function, whose output is 1 iff exactly k of the inputs are 1 [GGHK22, AJO11]. The adaptive SBFE problem has also been studied for functions represented by read-once CNF (dually, DNF) formulas [BÜ00] and for linear threshold functions and symmetric functions [DHK16, GGN24, AJO11, DJO$^+$12, KK13].

There has also been interest in the gap between the best adaptive and non-adaptive solutions to SBFE problems. The unit-cost SBFE problem for arbitrary symmetric functions has an adaptivity gap that is between 1.5 and 2 [GHKL22, PS24]. The unit-cost SBFE problem for the k-of-n function has an adaptivity gap of 1.5 [GHKL22, PS24]. For some other Boolean functions, the problem has non-constant gaps [HKLW22b].

Hardness. Few NP-hardness results are known for unit-cost SBFE problems. The unit-cost SBFE problem for functions represented by arbitrary CNF formulas is trivially NP-hard, in both the adaptive and non-adaptive settings: if the formula is not satisfiable, the optimal evaluation strategy does not observe any bits at all. By a simple reduction from vertex cover, NP-hardness also holds even if the CNF formula has no negations and is restricted to have 2 literals per clause [AHKÜ17]. Analogous results hold for DNF formulas by duality. We note that NP-hardness of the unit-cost SBFE problem for linear-threshold functions is still an open question. NP-hardness was shown for arbitrary costs in [CQK89], but contrary to what was stated in [DHK16], that result did not apply to the unit-cost case.

Related Problems. Finally, we note that there is a large body of work on Boolean function evaluation problems in both adversarial and probabilistic models with other assumptions or goals; see for example [KBZ86, IK84, CFG$^+$00, KKM05, CGLM11, BLT21, Ünl25, HNT25]. There has also been interest in functions over larger alphabets, like voting problems with more than two choices [HLS24, BSZ19]. More generally, stochastic probing problems (where each input is 1 with some known probability) are studied beyond function evaluation. For example, there is interest in solving encoded optimization problems, where input bits can be queried sequentially;

see e.g. [GN13, GNS17, Sin18, PRS23, SS21].

2 Preliminaries. In this section we review notation and terminology used throughout the paper. We also formalize a greedy algorithm for the Unanimous Vote problem that plays a central role in our main result.

2.1 Notation and terminology. Throughout, we use the term "increasing" (resp. "decreasing") as a synonym for "non-decreasing" (resp. "non-increasing"). We use "ordering" to refer to a permutation a on n elements, with order $a(1), a(2), \ldots, a(n)$. The outcome "heads" is synonymous with 1, and "tails" with 0. The input to our problem is a set of n coins with biases $p_1, \ldots, p_n \in [0,1]$. Without loss of generality, we assume $p_1 \leq \cdots \leq p_n$. The first step in all of our algorithms is to sort the probabilities in $O(n \log n)$ time. We denote $\overline{p}_i \stackrel{\text{def}}{=} 1 - p_i$.

The *cost* of an ordering is the expected number of flipped coins to determine whether or not a vote is unanimous. The cost can be written as the expectation of a sum of $\{0,1\}$ indicator random variables $\{X_j\}_{j \geq 1}$, where $X_j = 1$ if we reach the j^{th} coin in the ordering without terminating (i.e., we flip the j^{th} coin). We never terminate before the second coin. Moreover, for all $j \geq 3$, we only fail to terminate if the coins prior to the j^{th} coin all come up heads or all come up tails. Thus,

$$cost(a) = \sum_{j \geq 1} \mathbb{E}[X_j] = 2 + \sum_{j \geq 3} \left(\prod_{i=1}^{j-1} p_{a(i)} + \prod_{i=1}^{j-1} \overline{p}_{a(i)} \right).$$

Since these quantities will be important later, we introduce the notation $z_j^1(a) \stackrel{\text{def}}{=} \prod_{i=1}^{j-1} p_{a(i)}$ to be the chance that the first $j-1$ coins are *heads* in ordering a, and $z_j^0(a) \stackrel{\text{def}}{=} \prod_{i=1}^{j-1} \overline{p}_{a(i)}$ to be the chance that the first $j-1$ coins are *tails* in ordering a. As above, for $j \geq 3$, $\mathbb{E}[X_j] = z_j^1(a) + z_j^0(a)$. It follows that

$$cost(a) = 2 + \sum_{j \geq 3} z_j^0(a) + z_j^1(a) = 1 + \sum_{j \geq 2} z_j^0(a) + z_j^1(a). \tag{2.1}$$

We call an ordering, a, 0-*biased* at position j if $z_j^0(a) > z_j^1(a)$; i.e., it is more likely that the first $j-1$ coins flipped are all tails (0) than all heads (1). If $z_j^0(a) < z_j^1(a)$, we call the ordering 1-*biased* at position j. If $z_j^0(a) = z_j^1(a)$, we call the ordering *unbiased* at position j.

Using this language, we can uniquely partition the sequence $[a(1), \ldots, a(n)]$ into contiguous *blocks* $[B_1, \ldots, B_q]$ so that for each j, $a(j)$ and $a(j+1)$ belong to the same block if and only if a has the same bias type (0-biased, 1-biased, or unbiased) at positions j and $j+1$. See Figure 1.2 for an example partitioning of an ordering into its *biased blocks*.

2.2 The greedy algorithm. A natural greedy algorithm for the Unanimous Vote Problem is to always choose the coin that, if flipped, has the highest probability of terminating the algorithm. This probability can be expressed as follows:

FACT 2.1 (Probability of terminating the sequence). *Fix an ordering a and position $x > 1$. Assuming we flip $x-1$ coins without terminating, the probability of terminating after flipping the coin $a(x)$ is*

$$\frac{\overline{p}_{a(x)} z_x^1(a) + p_{a(x)} z_x^0(a)}{z_x^1(a) + z_x^0(a)} = 1 - \frac{z_{x+1}^1(a) + z_{x+1}^0(a)}{z_x^1(a) + z_x^0(a)}.$$

The value of $p_{a(x)}$ that maximizes the expression in Fact 2.1 depends on the *bias type* of a at position x. If it is 0-biased, the expression is increasing with $p_{a(x)}$; if it is 1-biased, the expression is *decreasing* with $p_{a(x)}$; if a is unbiased at position x, then the expression always equals $\frac{1}{2}$. So, in a 0-biased block, the choice that maximizes the probability of termination is to select the remaining coin with the *largest* heads probability. In a 1-biased block, it is to select the coin with the *smallest* heads probability. For unbiased blocks, selecting any coin terminates the sequence with probability exactly $\frac{1}{2}$, so the greedy choice is not unique. By convention, we always choose the remaining coin with *largest* heads probability, although our analysis would also works if we chose the coin with smallest heads probability. We may now construct a "greedy" algorithm:

GREEDY ALGORITHM (ALGORITHM 2.1)
Assume coins are in increasing order; i.e. $p_1 \leq \cdots \leq p_n$.
Start with an empty ordering. Repeat the following rule until all coins are chosen:
If the ordering is 1-biased, choose the remaining coin with smallest probability of heads.
Otherwise, choose the remaining coin with largest probability of heads.

We assume the algorithm can break ties arbitrarily (i.e. if two coins have the same probability of heads). See Figure 2.1 for an example output of the algorithm.

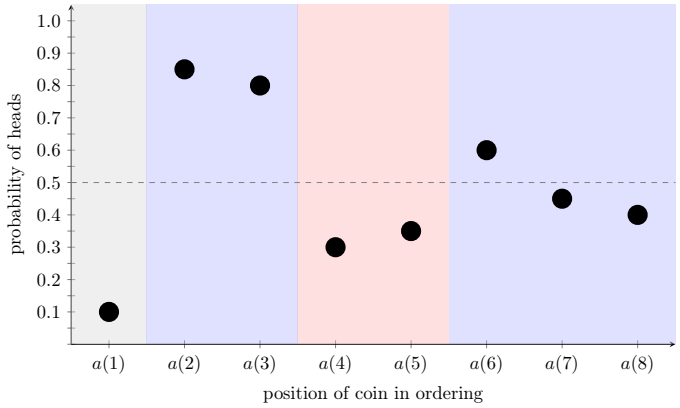

Figure 2.1: Plot of the probability of heads of a coin versus its position in an example greedy ordering. This is the same instance as in Figure 1.2. Unbiased blocks are in gray, 0-biased blocks are in blue, and 1-biased blocks are in red.

3 Structural results. In this section, we show that any optimal ordering for the Unanimous Vote problem must share several natural characteristics of the *greedy* ordering (Algorithm 2.1).

3.1 Monotonicity within a block. Consider the coins in a single 0-biased biased block of the greedy solution. Since, in a 0-biased block, Algorithm 2.1 chooses the next coin with largest probability of heads, the coins in this block will have *decreasing* probability of heads. Similarly, the coins in any 1-biased block of the greedy solution will be sorted so that the probability of heads is *increasing* in the block. Our first structural result is that the same fact holds for *any optimal ordering*.

The proof proceeds via an intuitive "swap" argument: if we have not terminated by the time we reach a 0-biased block, this is most likely because we have observed all tails (0s) up until that point. This remains true even if we swap the positions of coins in the block. Accordingly, we can increase our probability of termination by reordering the coins to flip a heads as quickly as possible, i.e., by moving coins with larger probability of heads to the front of the block. Formally, we require the following expression for how the cost of an ordering changes if adjacent coins are swapped:

CLAIM 3.1 (Cost of swapping adjacent positions). *Choose an ordering a and position $x \in [1, n-1]$. Let b be the ordering starting from a but swapping position x and $x+1$. Then*

$$cost(b) - cost(a) = \left(z_x^1(a) - z_x^0(a)\right)\left(p_{a(x+1)} - p_{a(x)}\right) .$$

Proof. The orderings a and b are identical except at positions x and $x+1$. So we have that $z_j^0(a) = z_j^0(b)$

and $z_j^1(a) = z_j^1(b)$ for all $j \in [1, \ldots, n] \setminus \{x+1\}$. Using the cost expression (2.1), we then have:

$$\begin{aligned} cost(b) - cost(a) &= \left(1 + \sum_{j \geq 2} \left(z_j^1(b) + z_j^0(b)\right)\right) - \left(1 + \sum_{j \geq 2} \left(z_j^1(a) + z_j^0(a)\right)\right) \\ &= z_{x+1}^1(b) - z_{x+1}^1(a) + z_{x+1}^0(b) - z_{x+1}^0(a) \\ &= z_x^1(a)\left(p_{a(x+1)} - p_{a(x)}\right) + z_x^0(a)\left(\overline{p}_{a(x+1)} - \overline{p}_{a(x)}\right) \\ &= \left(z_x^1(a) - z_x^0(a)\right)\left(p_{a(x+1)} - p_{a(x)}\right) . \end{aligned}$$

∎

Corollary 3.2 (Optimal ordering is sorted within each block). *Consider any optimal ordering a with block sequence $[B_1, \ldots, B_q]$. Within each 0-biased block, the coins have* decreasing *probability of heads. Within each 1-biased block, they have* increasing *probability of heads.*

Proof. Since a is optimal, we have $cost(b) - cost(a) \geq 0$ for all orderings b. Suppose x is a position within a 0-biased block, i.e. $z_x^0(a) > z_x^1(a)$. By Claim 3.1, we must have $p_{a(x)} \geq p_{a(x+1)}$; i.e. the next coin must have an equal or smaller probability of heads. The argument is symmetric for 1-biased blocks. ∎

3.2 Monotonicity across non-final blocks. The greedy solution obeys a stronger property than monotonicity within each block. *Every* coin in a 1-biased block has a smaller (or equal) probability of heads than all future coins. Similarly, *every* coin in a 0-biased or unbiased block (because of our convention) has a larger or equal probability of heads than all future coins. We show there is an optimal solution that also obeys this "across block" monotonicity, *at least among all coins in non-final blocks.* To prove this statement, we first require the following basic fact:

Claim 3.3 (Optimal ordering uses coins on one side of $\frac{1}{2}$ in non-final blocks). *Consider any optimal ordering a with block sequence $[B_1, \ldots, B_q]$. For all $1 \leq k < q$, if B_k is 0-biased, then it only contains coins with probability of heads $> \frac{1}{2}$. If B_k is 1-biased, then it only contains coins with heads probability of $< \frac{1}{2}$.*

Proof. Fix a block $B_k = [a(i), \ldots, a(j)]$ with $k \neq q$. If B_k is 0-biased, then in the last position $z_j^0(a) > z_j^1(a)$ but $z_{j+1}^0(a) \leq z_{j+1}^1(a)$. Since $z_{j+1}^1(a) = p_{a(j)} z_j^1(a)$ and $z_{j+1}^0(a) = \overline{p}_{a(j)} z_j^0(a)$, this implies $p_{a(j)} > \overline{p}_{a(j)}$; i.e. $p_{a(j)} > \frac{1}{2}$. By Corollary 3.2, all other coins in block B_k have probability of heads at least $p_{a(j)}$, which is greater than $\frac{1}{2}$. (The argument is symmetric for 1-biased blocks.) ∎

Claim 3.3 only considers 0-biased and 1-biased blocks. However, it is possible that the optimal ordering has unbiased blocks. This makes the optimal ordering non-unique: by Claim 3.1, swapping a coin in an unbiased block with the following coin in the ordering does not change the cost. Nonetheless, we show there is an optimal ordering where all coins in non-final blocks have probability *not* equal to $\frac{1}{2}$:

Claim 3.4. *There is an optimal ordering a with block sequence $[B_1, \ldots, B_q]$ where every unbiased block that is not B_q has just one coin, and that coin has probability of heads* not equal *to $\frac{1}{2}$. Moreover, if B_q is unbiased and contains more than one coin, all coins in the block have probability of heads equal to $\frac{1}{2}$.*

Proof. We begin with an optimal ordering a with block sequence $[B_1, \ldots, B_q]$, and argue that we can swap adjacent coins, without changing the cost of the ordering, in such a way that the properties above are eventually satisfied. Note that the last coin of any non-final unbiased block *must* have probability of heads not equal to $\frac{1}{2}$, since a is unbiased at j and *not* unbiased at position $j+1$.

Let k be the first block $B_k = [a(i), \ldots, a(j)]$ in a such that B_k is unbiased, has more than one coin, and $k \neq q$. For any $i \leq x < j$, a is unbiased at positions x and $x+1$; i.e., both $z_x^1(a) = z_x^0(a)$ and $p_{a(x)} z_x^1(a) = z_{x+1}^1(a) = z_{x+1}^0(a) = \overline{p}_{a(x)} z_x^0(a)$. So the probability $p_{a(x)}$ *must* be equal to $\frac{1}{2}$.

By Claim 3.1, for any $i \leq x \leq j$, we can swap position x and $x+1$ without changing the cost. Suppose we do this with position $x = j-1$. Since the new coin at position $j-1$ has probability of heads *not* equal to $\frac{1}{2}$, the coin *initially* at position $j-1$ moves to the block B_{k+1}. We may now swap position $j-2$ and $j-1$, moving the coin *initially* at position $j-2$ to the block B_{k+1}. We continue this process until we swap position i and $i+1$. At this point, the block B_k has one coin in it. Moreover, since B_k is unbiased, B_{k+1} must be 0-biased or 1-biased. Since the coins previously in block B_k have probability of heads equal to $\frac{1}{2}$, by Claim 3.3, B_{k+1} must be the final block. So, by making these swaps, we satisfy the first part of Claim 3.4.

Suppose at this point, the final block $B_q = [a(x), \ldots, a(n)]$ is unbiased and has more than one coin. By the same argument as before, every coin except possibly the last coin in B_q has probability of heads equal to $\frac{1}{2}$. If $p_{a(n)} = \frac{1}{2}$, we are done. Otherwise, we may repeat the process from before until B_q has one coin in it. In this case, a new block B_{q+1} is created which is either 0-biased or 1-biased. ∎

By Claim 3.3 and Claim 3.4, there is an optimal ordering where the coins in non-final blocks can be partitioned into S_+ and S_-, such that all coins in S_+ have probability of heads greater than $\frac{1}{2}$, and all coins in S_- have probability of heads *less* than $\frac{1}{2}$. We show that all coins in S_+ are sorted in decreasing probability of heads, and all coins in S_- are sorted in *increasing* probability of heads. It is in this sense we show that there is an optimal ordering with monotonicity across non-final blocks. To show this, we give an expression for the change in cost of an ordering when coins are swapped across blocks:

Claim 3.5 (Cost of swapping across a block). *Choose an ordering a. Fix positions $s, t \in [n]$ where $s < t$. Let b be the ordering obtained from swapping positions s and t in a. Suppose $cost(a) \le cost(b)$. Then:*

- *If a is 0-biased at s and $p_{a(x)} \le \frac{1}{2}$ for all $s < x < t$, then $p_{a(s)} \ge p_{a(t)}$.*
- *If a is 1-biased at s and $p_{a(x)} \ge \frac{1}{2}$ for all $s < x < t$, then $p_{a(s)} \le p_{a(t)}$.*
- *If a is unbiased at s and $p_{a(x)} \le \frac{1}{2}$ for all $s < x < t$ and $p_{a(y)} < \frac{1}{2}$ for some $s < y < t$, $p_{a(s)} \ge p_{a(t)}$.*
- *If a is unbiased at s and $p_{a(x)} \ge \frac{1}{2}$ for all $s < x < t$ and $p_{a(y)} > \frac{1}{2}$ for some $s < y < t$, $p_{a(s)} \le p_{a(t)}$.*

Proof. By construction, $z_j^0(a) = z_j^0(b)$ and $z_j^1(a) = z_j^1(b)$ for all $j \le s$ and $j \ge t+1$. The cost difference is

$$\begin{aligned}
0 \le cost(b) - cost(a) &= \left(1 + \sum_{j \ge 2} \left(z_j^1(b) + z_j^0(b)\right)\right) - \left(1 + \sum_{j \ge 2} \left(z_j^1(a) + z_j^0(a)\right)\right) \\
&= \sum_{j=s+1}^{t} z_j^1(b) - z_j^1(a) + z_j^0(b) - z_j^0(a) \\
&= \sum_{j=s+1}^{t} \left(z_s^1(a) \cdot (p_{a(t)} - p_{a(s)}) \left(\prod_{x=s+1}^{j-1} p_{a(x)} \right) + z_s^0(a) \cdot (\overline{p}_{a(t)} - \overline{p}_{a(s)}) \left(\prod_{x=s+1}^{j-1} \overline{p}_{a(x)} \right) \right) \\
&= (p_{a(t)} - p_{a(s)}) \left(z_s^1(a) \cdot \left(\sum_{j=s+1}^{t} \prod_{x=s+1}^{j-1} p_{a(x)} \right) - z_s^0(a) \cdot \left(\sum_{j=s+1}^{t} \prod_{x=s+1}^{j-1} \overline{p}_{a(x)} \right) \right).
\end{aligned}$$

Suppose $p_{a(x)} \le \frac{1}{2}$ for all $s < x < t$. Then for each x, $p_{a(x)} \le \overline{p}_{a(x)}$, and so the first summation of products is at most the second summation of products. We can conclude then:

- If a is 0-biased at s, then $z_s^0(a) > z_s^1(a)$, and so the second term is strictly less than 0. Since the difference in cost must be non-negative, we have $p_{a(s)} \ge p_{a(t)}$.
- If instead a is unbiased at s, then $z_s^1(a) = z_s^0(a)$, and so the second term is at most 0. Since the difference in cost must be non-negative, it is possible that $p_{a(s)} < p_{a(t)}$ only if the second term is equal to 0. This can only occur if $p_{a(x)} = \frac{1}{2}$ for all $s < x < t$.

The parts of the claim when $p_{a(x)} \ge \frac{1}{2}$ for all $s < x < t$ follow from a symmetric argument. ∎

Corollary 3.6 (Optimal ordering is sorted across blocks). *There exists an optimal ordering a with block sequence $[B_1, \ldots, B_q]$ where the following holds:*

- *The coins in blocks $[B_1, \ldots, B_{q-1}]$ can be partitioned into a set S_+ and a set S_-, where all coins in S_+ have probability of heads more than $\frac{1}{2}$, and all coins in S_- have probability of heads less than $\frac{1}{2}$.*
- *The ordering a restricted to the coins in S_+ are sorted in a by decreasing probability of heads.*
- *The ordering a restricted to the coins in S_- are sorted in a by* increasing *probability of heads.*

Proof. Take the ordering guaranteed by Claim 3.4. By Claim 3.3 we can partition the coins in blocks $[B_1, \ldots, B_{q-1}]$ into S_+ and S_-. Note that no coins in S_+ or S_- have probability of heads equal to $\frac{1}{2}$. The monotonicity of coins in S_+ and S_- then follows by Claim 3.5. ∎

4 Constructing an exact algorithm. In this section, we show how to generate an optimal ordering from small modifications to the greedy ordering. This relies on the structural results from Section 3.

4.1 Monotonicity almost everywhere. In the greedy ordering, "monotonicity across blocks" holds *even for the final block.* A coin in a 0-biased block has probability of heads at least that of every following coin. Similarly, a coin in a 1-biased block has probability of heads at most that of every following coin. Surprisingly, it turns out that the optimal ordering guaranteed by Claim 3.4 *also* has this property, except possibly at one location.

CLAIM 4.1 (There exists an optimal ordering that is almost greedy). *There is an optimal ordering a with block sequence $[B_1, \ldots, B_q]$ where the following statement holds at each position x except for the position of the last coin of B_{q-1}:*

- *Suppose a is 0-biased at position x. Then for all positions $x' > x$, $p_{a(x)} \geq p_{a(x')}$.*
- *Suppose a is 1-biased at position x. Then for all positions $x' > x$, $p_{a(x)} \leq p_{a(x')}$.*
- *Suppose a is unbiased at position x and $p_{a(x)} > \frac{1}{2}$. Then for all positions $x' > x$, $p_{a(x)} \geq p_{a(x')}$.*
- *Suppose a is unbiased at position x and $p_{a(x)} < \frac{1}{2}$. Then for all positions $x' > x$, $p_{a(x)} \leq p_{a(x')}$.*

Furthermore, if the last block B_q is unbiased, then the above statement holds at all positions.

Proof. Consider an optimal ordering a guaranteed by Corollary 3.6. Suppose it has block sequence $[B_1, \ldots, B_q]$, and the coins in the non-final blocks are partitioned into S_+ and S_-. S_+ and S_- are already sorted by probability of heads, and each coin in S_+ has probability of heads larger than every coin in S_-. So if a violation of the statement occurs, the violating position x' must be in the final block.

Suppose the final block B_q is unbiased. In this case we show that the statement holds at all positions:

- Suppose B_q contains more than one coin. Then all coins in B_q have probability of heads $\frac{1}{2}$. The statement is vacuous for any position $x \in B_q$. Moreover, $\frac{1}{2}$ is smaller than the probability of heads of all coins in S_+ and larger than the probability of heads of all coins in S_-, so there are no violations for any $x \notin B_q$ and $x' \in B_q$. So the statement holds at all positions.
- Suppose B_q contains just one coin (x', at position n). Suppose the coin at position $n-1$ is in S_+. (A symmetric argument holds if it is in S_-). Since B_q is unbiased, this coin must belong to a 0-biased block; by Claim 3.5, it has probability of heads at least $p_{a(n)}$. By Corollary 3.6, all coins in S_+ have probability of heads at least $p_{a(n)}$. Now consider the last coin in S_-, which is in position at most $n-2$. Note that the coin at position $n-1$ has probability of heads at least $\frac{1}{2}$. So then by Claim 3.5, the last coin in S_- has probability of heads at most $p_{a(n)}$. By Corollary 3.6, all coins in S_- have probability of heads at most $p_{a(n)}$. So, the statement holds for any $x < n = x'$.

Next, we consider when the final block B_q is 1-biased. (A symmetric argument holds when B_q is 0-biased.) First of all, by Corollary 3.2, the coins in B_q are sorted in increasing probability of heads. Moreover, by Claim 3.5, all coins in S_- have probability of heads at most that of any coin in B_q. So if there is a violation of the statement, it occurs between a position x in S_+ and x' in B_q.

Suppose for the sake of contradiction there are *two* positions k, ℓ in S_+ where the statement does not hold. Let $k < \ell$; then $p_{a(k)} \geq p_{a(\ell)}$. So, there exists a position x' in B_q where $p_{a(x')} > p_{a(k)} \geq p_{a(\ell)}$. Since the coins in S_+ are sorted in *decreasing* probability of heads, we can take k, ℓ to be the last two coins in S_+; so $p_{a(x)} < \frac{1}{2}$ for all $k < x < \ell$. We choose x' to be the earliest position where the violation with k and ℓ occurs.

We show that swapping positions k and x' decreases the cost, which contradicts our claim of an optimal ordering. Let b be the ordering starting from a but swapping k and x'. Then (as in the proof of Claim 3.5),

$$cost(b) - cost(a) = (p_{a(x')} - p_{a(k)}) \left(z_k^1(a) \cdot \left(\sum_{j=k+1}^{x'} \prod_{x=k+1}^{j-1} p_{a(x)} \right) - z_k^0(a) \cdot \left(\sum_{j=k+1}^{x'} \prod_{x=k+1}^{j-1} \overline{p}_{a(x)} \right) \right) .$$

Since $p_{a(x')} > p_{a(k)}$, the sign of the difference in cost depends on the sign of the term in parentheses. We split this term into two parts: the first with summation terms where $j \in [k+1, \ldots, \ell-1]$, and the second with summation terms where $j \in [\ell, \ldots, x']$. Recall that at positions k, ℓ, the ordering a is *not* 1-biased; i.e. $z_k^1(a) \le z_k^0(a)$ and $z_\ell^1(a) \le z_\ell^0(a)$.

- The first part of the expression involves terms $p_{a(x)}$ for $k+1 \le x \le \ell-2$. For all these terms, $p_{a(x)} < \frac{1}{2}$; so
$$z_k^1(a) \cdot \left(\sum_{j=k+1}^{\ell-1} \prod_{x=k+1}^{j-1} p_{a(x)} \right) - z_k^0(a) \cdot \left(\sum_{j=k+1}^{\ell-1} \prod_{x=k+1}^{j-1} \overline{p}_{a(x)} \right) \le 0 \,.$$
- The second part of the expression also involves terms $p_{a(x)}$ for $\ell-1 \le x \le x'-1$. Since position x' has the first coin where $x' > \ell$ such that $p_{a(x')} \ge p_{a(k)}$, we have $p_{a(x)} \le p_{a(k)}$ for all these terms. We upper-bound the second part, setting $p \stackrel{\text{def}}{=} p_{a(k)}$ for convenience:
$$\begin{aligned}
&z_k^1(a) \cdot \left(\sum_{j=\ell}^{x'} \prod_{x=k+1}^{j-1} p_{a(x)} \right) - z_k^0(a) \cdot \left(\sum_{j=\ell}^{x'} \prod_{x=k+1}^{j-1} \overline{p}_{a(x)} \right) \\
&= z_k^1(a) \cdot \left(\prod_{x=k+1}^{\ell-1} p_{a(x)} \right) \left(1 + \sum_{j=\ell+1}^{x'} \prod_{x=\ell}^{j-1} p_{a(x)} \right) - z_k^0(a) \cdot \left(\prod_{x=k+1}^{\ell-1} \overline{p}_{a(x)} \right) \left(1 + \sum_{j=\ell+1}^{x'} \prod_{x=\ell}^{j-1} \overline{p}_{a(x)} \right) \\
&\le z_k^1(a) \cdot \left(\prod_{x=k+1}^{\ell-1} p_{a(x)} \right) \left(1 + \sum_{j=\ell+1}^{x'} \prod_{x=\ell}^{j-1} p \right) - z_k^0(a) \cdot \left(\prod_{x=k+1}^{\ell-1} \overline{p}_{a(x)} \right) \left(1 + \sum_{j=\ell+1}^{x'} \prod_{x=\ell}^{j-1} \overline{p} \right) \\
&= z_k^1(a) \cdot \left(\prod_{x=k+1}^{\ell-1} p_{a(x)} \right) \cdot \frac{1 - p^{x'-\ell+1}}{\overline{p}} - z_k^0(a) \cdot \left(\prod_{x=k+1}^{\ell-1} \overline{p}_{a(x)} \right) \cdot \frac{1 - \overline{p}^{x'-\ell+1}}{p} \\
&= z_k^1(a) \cdot \left(\prod_{x=k}^{\ell-1} p_{a(x)} \right) \cdot \frac{1 - p^{x'-\ell+1}}{p \cdot \overline{p}} - z_k^0(a) \cdot \left(\prod_{x=k}^{\ell-1} \overline{p}_{a(x)} \right) \cdot \frac{1 - \overline{p}^{x'-\ell+1}}{p \cdot \overline{p}} \\
&= \frac{1}{p \cdot \overline{p}} \left(z_\ell^1(a) \cdot (1 - p^{x'-\ell+1}) - z_\ell^0(a) \cdot (1 - \overline{p}^{x'-\ell+1}) \right) .
\end{aligned}$$

  The inequality holds because $p_{a(x)} \le p_{a(k)}$ for all $\ell \le x < x'$. Since $p = p_{a(k)} > \frac{1}{2}$, we have $p^t > \overline{p}^t$ for any $t \ge 1$. Since $z_\ell^1(a) \le z_\ell^0(a)$, the first term is strictly less than the second term, so the expression is negative.

The sign of $cost(b) - cost(a)$ is the sign of the sum of these two terms. So $cost(b) - cost(a) < 0$. This contradicts our assumption that a was an optimal ordering.

Finally, we remark on which block the last coin of S_+ is in. Since B_q is 1-biased, B_{q-1} must be 0-biased or unbiased. If it is 0-biased, we are done. Otherwise, B_{q-1} is unbiased, and so it contains one coin. If the coin in B_{q-1} has probability of heads less than $\frac{1}{2}$, then B_q is 0-biased, which is a contradiction. So the coin in B_{q-1} must have probability of heads larger than $\frac{1}{2}$, and so belongs to S_+. So the only coin that can violate the statement in the claim is the last coin in S_+, which is also the last coin in B_{q-1}. ∎

4.2 Handling unbiased blocks. Claim 4.1 implies there is an optimal ordering where at all positions *except one*, the correct choice of coin is the remaining coin with largest or smallest probability of heads. This of course depends on the bias: the largest probability when 0-biased, and the smallest probability when 1-biased.

What happens for unbiased blocks? Since this optimal ordering is the same as any one guaranteed by Claim 3.4, any unbiased block that is not the last block contains only one coin. Let the block sequence of the ordering be $[B_1, \ldots, B_q]$. By Claim 4.1, unless this block is B_{q-1}, it uses either the smallest or largest probability of heads. We argue that Claim 4.1 still holds if we assume unbiased blocks (that are not next-to-last) always use the coin with *largest* remaining probability of heads. We do this to match our choice of convention in Algorithm 2.1.

CLAIM 4.2. *There is an optimal ordering a with block sequence $[B_1, \ldots, B_q]$ satisfying the properties of Claim* 4.1*, and each unbiased block B_k (for $k \ne q-1$) uses the coins with largest probability of heads among coins in $[B_k, \ldots, B_q]$.*

Proof. Start with an ordering a generated by Claim 4.1. We prove this in cases:

- Suppose the unbiased block B_i has $i < q-2$, and uses a coin with probability of heads less than $\frac{1}{2}$. Then the next block B_{i+1} is 0-biased, and its first coin has the largest probability of heads among all coins in $[B_{i+1}, \ldots, B_q]$. By Claim 3.1, we may swap the coin in B_i with the first coin in B_{i+1} without changing the cost; now the coin in B_i has the largest probability of heads among coins in the rest of the ordering.

- Suppose the unbiased block is B_{q-2}, *and* B_{q-1} has more than one coin. Then the argument in the last case works here as well.

- Suppose the unbiased block is B_{q-2}, *and* B_{q-1} has exactly one coin. If B_{q-2} uses a coin with probability less than $\frac{1}{2}$, then B_{q-1} is 0-biased and uses a coin with probability more than $\frac{1}{2}$. We may switch the two coins without changing the cost; then the ordering in B_{q-1} is now 1-biased.
  - If B_q is also 1-biased, then the two blocks "merge" into a single block after switching the two coins. Then B_{q-2} is the "next-to-last" block, and is exempt from the claim.
  - If B_q is unbiased, then by Claim 4.1, the optimal ordering can be greedily chosen. The number of blocks remains q after switching the two coins, so the new coin in B_{q-2} is larger than all subsequent coins in the ordering.

- Suppose the unbiased block B_i has $i = q$. Then it trivially uses the coin with smallest remaining probability of heads, unless it has more than one coin. If so, this is also true, because by Claim 3.4, all of its coins have probability of heads equal to $\frac{1}{2}$. ∎

4.3 A polynomial-time algorithm. Putting all of this together, Claim 4.2 guarantees an optimal ordering can be constructed by applying this local rule at every position except one:

If the ordering is 1*-biased, select the remaining coin with smallest probability of heads.*
Otherwise, select the remaining coin with largest *probability of heads.*

This is exactly the rule of the greedy algorithm in Algorithm 2.1! As a result, there is a $O(n^3)$ algorithm to find the optimal ordering, which we state as Algorithm 4.1:

> Modified Greedy Algorithm (Algorithm 4.1)
> Assume coins are in increasing order; i.e. $p_1 \leq \cdots \leq p_n$.
> For all $j \in [1, \ldots, n]$, generate the following orderings:
> (Greedy) For the first $(j-1)$ coins:
> If the ordering is 1-biased, choose the remaining coin with smallest probability of heads.
> Otherwise, choose the remaining coin with *largest* probability of heads.
> (Brute Force) For every coin x in the set of remaining coins, let $c_{j,x}$ be the following ordering:
> Choose coin x.
> (Greedy) For the remaining $(n-j)$ coins:
> If the ordering is 1-biased, choose the remaining coin with smallest probability of heads.
> Otherwise, choose the remaining coin with largest probability of heads.
> Compute the expected cost for all $c_{j,x}$, and return an ordering c that minimizes this cost.

Theorem 4.3. *Algorithm* 4.1 *exactly solves the Unanimous Vote problem and runs in time* $O(n^3)$.

Proof. Algorithm 4.1 generates *all* orderings that use the greedy algorithm's rule for at least $(n-1)$ positions. By Claim 4.2, it must find an optimal ordering. Algorithm 4.1 considers $O(n^2)$ orderings, and the cost of any ordering can be computed in $O(n)$ time. ∎

4.4 A faster algorithm. We can construct a faster algorithm to recover the optimal ordering. This is because one can use the output of the greedy algorithm (Algorithm 2.1) to identify the "non-greedy position" in the optimal ordering.

FASTER MODIFIED GREEDY ALGORITHM (ALGORITHM 4.2)
Assume coins are in increasing order; i.e. $p_1 \le \cdots \le p_n$.
Generate the ordering b from Algorithm 2.1. Let x be the position just prior to the final block.
For each $x' \in [x+1, \ldots, n]$, generate the ordering $c_{x'}$ by starting from b, moving the coin at position x' to position $x+1$, and moving the coin at position x to the end of the ordering.
Compute the expected cost for b and for all $c_{x'}$, and return an ordering c that minimizes this cost.

THEOREM 4.4. *Algorithm* 4.2 *exactly solves the Unanimous Vote problem and runs in time* $O(n^2)$.

Proof. Algorithm 4.2 generates *all* orderings that use the greedy algorithm's rule at every position except possibly position x. It generates $O(n)$ orderings. The cost of any ordering can be computed in $O(n)$ time.

We now argue that the algorithm is correct. Consider an optimal ordering a generated by Claim 4.2, and let x be the position just prior to the final block. If x was greedily chosen, then we are done. So we assume the coin at position x was not chosen greedily. By Claim 4.1, this can only happen if the final block is 0-biased or 1-biased.

We assume the final block is 1-biased; a symmetric argument holds if it is 0-biased. Then the coin at position x has probability of heads greater than $\frac{1}{2}$, yet there is a coin in the final block which has probability of heads greater than $p_{a(x)}$. Since the final block is sorted in *increasing* probability of heads, $p_{a(n)} > p_{a(x)}$. Moreover, because the final block is 1-biased, $z^1_{x'}(a) > z^0_{x'}(a)$ for all $x+1 \le x' \le n$.

Now consider the ordering b output by Algorithm 2.1. Before position x, a and b are identical. So $z^1_x(b) = z^1_x(a) \le z^0_x(a) = z^0_x(b)$. But at position x, the greedy ordering chooses coin $a(n)$; i.e. the coin with largest remaining probability of heads. So then for all $x+1 \le x' \le n$,

$$z^1_{x'}(b) = \frac{p_{a(n)}}{p_{a(x)}} \cdot z^1_{x'}(a) > \frac{p_{a(n)}}{p_{a(x)}} \cdot z^0_{x'}(a) > \frac{\overline{p}_{a(n)}}{\overline{p}_{a(x)}} \cdot z^0_{x'}(a) = z^0_{x'}(b)\,.$$

This implies the greedy ordering is 1-biased for all positions $x' \in [x+1, \ldots, n]$, but not before. Thus, for *both* orderings a and b, position x is just prior to the final block.

As a result, Claim 4.2 implies that we can find an optimal ordering if we apply the greedy algorithm's rule to all positions except possibly position x. Since Algorithm 4.2 searches over all such orderings, it will find an optimal ordering. ∎

In fact, the cost of the all orderings from Algorithm 4.2 may be simultaneously computed in $O(n)$ time by storing partial sums. This allows us to achieve the runtime claimed in the introduction.

THEOREM 4.5 (Theorem 1.1, restated). *Algorithm* 4.2 *exactly solves the Unanimous Vote problem and runs in time* $O(n \log n)$. *If the input probabilities are already sorted, the algorithm runs in time* $O(n)$.

Proof. We know that Algorithm 4.2 is correct by Theorem 4.4. It remains to show that the cost of all generated orderings can be simultaneously computed in time $O(n)$.

In Algorithm 4.2, we create c_n starting from b and then swapping positions x and n. But for any $x' \in [x+1, n-1]$, we may create $c_{x'}$ starting from $c_{x'+1}$ and then swapping positions x and x'. So we set $b \stackrel{\text{def}}{=} c_{n+1}$ (and $b(n+1) \stackrel{\text{def}}{=} b(x)$), and look at the difference for all $x' \in [x+1, n]$:

$$\begin{aligned} cost(c_{x'}) - cost(c_{x'+1}) = &\left(\prod_{i=1}^{x-1} p_{b(i)}\right)(p_{b(x')} - p_{b(x'+1)})(p_{b(x+1)} + p_{b(x+1)}p_{b(x+2)} + \cdots + \prod_{i=x+1}^{x'-1} p_{b(i)}) \\ &+ \left(\prod_{i=1}^{x-1} \overline{p}_{b(i)}\right)(\overline{p}_{b(x')} - \overline{p}_{b(x'+1)})(\overline{p}_{b(x+1)} + \overline{p}_{b(x+1)}\overline{p}_{b(x+2)} + \cdots + \prod_{i=x+1}^{x'-1} \overline{p}_{b(i)})\,. \end{aligned}$$

We may factor out $(p_{b(x')} - p_{b(x'+1)})$ from the above expression:

$$\begin{aligned}\frac{cost(c_{x'}) - cost(c_{x'+1})}{p_{b(x')} - p_{b(x'+1)}} &= \left(\prod_{i=1}^{x-1} p_{b(i)}\right) \sum_{j=x+1}^{x'-1} \prod_{i=x+1}^{j} p_{b(i)} - \left(\prod_{i=1}^{x-1} \overline{p}_{b(i)}\right) \sum_{j=x+1}^{x'-1} \prod_{i=x+1}^{j} \overline{p}_{b(i)} \\ &= \frac{1}{p_{b(x)}} \sum_{j=x+1}^{x'-1} \prod_{i=1}^{j} p_{b(i)} - \frac{1}{\overline{p}_{b(x)}} \sum_{j=x+1}^{x'-1} \prod_{i=1}^{j} \overline{p}_{b(i)} \\ &= \sum_{j=x+2}^{x'} \left(\frac{z_j^1(b)}{p_{b(x)}} - \frac{z_j^0(b)}{\overline{p}_{b(x)}}\right).\end{aligned}$$

We now explain how to compute the cost of all orderings:

1. Compute the cost of the greedy ordering $b \stackrel{\text{def}}{=} c_{n+1}$.
2. For all $j \in [x+2, \ldots, n]$:
   - Compute $z_j^1(b)$ and $z_j^0(b)$ using dynamic programming (e.g., $z_{j+1}^1(b) = p_{b(j)} z_j^1(b)$).
   - Compute $y_j \stackrel{\text{def}}{=} \frac{z_j^1(b)}{p_{b(x)}} - \frac{z_j^0(b)}{\overline{p}_{b(x)}}$.
3. Let $\Delta_{x+1} \stackrel{\text{def}}{=} 1$. For each $x' \in [x+2, \ldots, n]$, compute the partial sum $\Delta_{x'} \stackrel{\text{def}}{=} \sum_{j=x+2}^{x'} y_j$.
4. For each $x' \in [n, \ldots, x+1]$, compute $cost(c_{x'}) = cost(c_{x'+1}) + (p_{b(x')} - p_{b(x'+1)}) \cdot \Delta_{x'}$. ∎

Each of these steps takes $O(n)$ time. We can then identify the minimum-cost ordering in $O(n)$ time.

The algorithm of Gkenosis et al. [GGHK18] for the *adaptive* variant of the Unanimous Vote problem, described in Section 1.3, also computes the cost of n orderings and chooses the ordering with lowest cost. We remark that a very similar idea can be used to improve the runtime of their algorithm from $O(n^2)$ to $O(n \log n)$, or $O(n)$ if the input probabilities are already sorted.

5 Comparing the greedy ordering with the optimal ordering. The greedy ordering constructed by Algorithm 2.1 is *not* always optimal. Here is an explicit example:

CLAIM 5.1. *The greedy ordering for the instance* $(p_1, p_2, p_3, p_4) = (0.49, 0.99, 0.99, 1)$ *is not optimal.*

Proof. Using Algorithm 2.1, the greedy ordering for this instance is $(1, 0.49, 0.99, 0.99)$, which has cost

$$2 + 1 \cdot 0.49(1 + 0.99) + 0 \cdot 0.51(1 + 0.01) = 2.9751\,.$$

Meanwhile, the ordering $[0.49, 0.99, 0.99, 1]$ has cost

$$2 + 0.49(0.99 + 0.99^2) + 0.51(0.01 + 0.01^2) = 2.9705\,.$$ ∎

In fact, there are instances where *every* algorithm using the natural greedy rule is suboptimal. In particular, consider algorithms that may choose any coin that, if flipped, has the largest probability of terminating the algorithm. At unbiased positions, including the first position, such an algorithm is free to choose any remaining coin. It can be checked that any such algorithm is suboptimal on the instance $(p_1, p_2, p_3, p_4, p_5, p_6) = (0.02, 0.24, 0.7, 0.8, 0.9, 0.993)$.

For any $\delta > 0$, we can construct an instance where the greedy algorithm (Algorithm 2.1) returns an ordering that on average flips $(1 - \delta)$ more coins than the optimal ordering:

CLAIM 5.2 (Lower bound of Claim 1.2). *For all* $0 < \delta < 1$, *there exists a length-*$\lceil \frac{4}{\delta^2} \rceil$ *instance where Algorithm* 2.1 *returns an ordering with cost greater than* $1 - \delta$ *plus the cost of an optimal ordering.*

Proof. Consider one coin with probability of heads 1 and n coins with probability of heads $1 - \epsilon$. There are $(n+1)$ coins in this instance. Let a be the optimal ordering and b be the ordering returned by Algorithm 2.1.

Algorithm 2.1 will first select the coin with probability of heads 1, and then use all other coins. This has cost

$$cost(b) = 2 + \sum_{j=2}^{n} \left(\prod_{i=1}^{j} p_{b(i)} + \prod_{i=1}^{j} \overline{p}_{b(i)} \right) = 2 + \sum_{j=1}^{n-1} (1-\epsilon)^j \,.$$

Meanwhile, consider the ordering that first uses all coins with probability of heads $1-\epsilon$. Then

$$cost(a) = 2 + \sum_{j=2}^{n} \left((1-\epsilon)^j + (\epsilon)^j \right) \,.$$

Then the difference in cost is at least

$$cost(b) - cost(a) = (1-\epsilon) - (1-\epsilon)^n - \sum_{j=2}^{n} \epsilon^j \geq 1 - \frac{\epsilon}{1-\epsilon} - (1-\epsilon)^n \,.$$

We set $\epsilon \stackrel{\text{def}}{=} \frac{\ln n}{n}$; then $\frac{\epsilon}{1-\epsilon} \leq 2\epsilon$ and $(1-\epsilon)^n \leq e^{-\ln n} = \frac{1}{n}$ for all $n \geq 1$. So the difference in cost is at least $1 - 2\epsilon - \frac{1}{n} = 1 - \frac{2\ln n + 1}{n}$. So for any desired δ, we set n large enough so that $\delta > \frac{2\ln n+1}{n}$, and this instance has a difference in cost at least $1-\delta$. One can verify that $\delta > \frac{2\ln\lceil 4/\delta^2 \rceil + 1}{\lceil 4/\delta^2 \rceil}$ for all $0 < \delta < 1$. ∎

We show that the cost of Algorithm 2.1 is always at most 1 more than the optimal solution. Thus the instances in Claim 5.2 are the asymptotically worst possible for Algorithm 2.1.

Claim 5.3 (Upper bound of Claim 1.2). *For all instances, Algorithm* 2.1 *returns an ordering with cost at most* 1 *plus the cost of an optimal ordering.*

Proof. Let a be the optimal ordering guaranteed by Claim 4.2 and b be the ordering returned by Algorithm 2.1. We need to prove that $cost(b) \leq cost(a) + 1$. Let k be the position in a just before the last block. By Claim 4.2, a and b agree on all coins in positions $< k$.

Suppose the last block of a is unbiased. Then by Claim 4.1, $a = b$ and so $cost(b) = cost(a)$. There are two remaining cases:

Case 1: The last block of a is 1-biased.

Consider the coin in position k of a; it is the only coin that might not have been greedily chosen. Since the last block is 1-biased, this coin must have probability of heads greater than $\frac{1}{2}$.

We split into two subcases: $p_{a(k)} \geq p_{a(n)}$, and $p_{a(k)} < p_{a(n)}$.

In the first subcase, $p_{a(k)} \geq p_{a(n)}$, and it follows from Corollary 3.2 that $p_{a(k)} \geq p_{a(x)}$ for all $k < x \leq n$. The second-to-last block must be 0-biased or unbiased. Either way, $a = b$, so $cost(b) = cost(a)$.

In the second subcase, $p_{a(k)} < p_{a(n)}$, we generate two intermediate orderings:

1. Let a' be the length-$(n+1)$ ordering produced by starting from a, and then inserting a coin with probability $p_{a(n)}$ at position k.
2. Let a'' be the length-$(n+1)$ ordering produced by starting from a', and then sorting all coins in position $\geq k+1$ in increasing probability of heads. ∎

Since we added one coin to create a', $cost(a') \leq cost(a) + 1$. Since $p_{a(n)} > p_{a(k)} > \frac{1}{2}$, the last block of a' begins at position $k+1$. By Claim 3.1, sorting the last block of an ordering can only reduce its cost, so $cost(a'') \leq cost(a')$. At position k, ordering a'' is either 0-biased or unbiased. It follows that the first n coins of a'' exactly match b, so $cost(b) \leq cost(a'') \leq cost(a') \leq cost(a) + 1$.

Case 2: The last block of a is 0-biased.

The second-to-last block can be either 1-biased or unbiased. If it is 1-biased, then the claim follows from a symmetric version of the argument used in Case 1.

Suppose that the second-to-last block is unbiased. Let a' be the ordering starting from a, but switching the coins at positions k and $k+1$. Since a is unbiased at position k, $cost(a') = cost(a)$, and so a' is also optimal. Then a' is unbiased at position k, either 0-biased, unbiased, or 1-biased at position $k+1$, and 0-biased from position $k+2$ through position n. We consider each subcase:

- If a' is 0-biased at position $k+1$, then the last block of a' starts at this position. By Corollary 3.2, this block is sorted in decreasing probability of heads. Then $b = a'$, and so $cost(b) = cost(a') = cost(a)$.
- If a' is unbiased at position $k+1$, then the coin at position k has probability of heads $\frac{1}{2}$. Since a' is 0-biased at position $k+2$, the coin at position $k+1$ has probability of heads less than $\frac{1}{2}$. If $k+1 = n$, then $b = a'$, and so $cost(b) = cost(a') = cost(a)$. Otherwise, we repeat *Case 2* on the ordering a'.
- If a' is 1-biased at position $k+1$, then the claim follows from a symmetric version of the argument used in Case 1.

The proof of Claim 5.3 relies on the convention used by Algorithm 2.1 for choosing the coins in unbiased positions. It is natural to consider an algorithm that also generates a greedy ordering using the opposite convention (smallest p_i rather than largest p_i) and outputs the best of the two orderings. This algorithm is optimal unless $p_1 \le \frac{1}{2} \le p_n$. If $p_1 \le \frac{1}{2} \le p_n$, there is at least a 50% chance of terminating in two flips. One can modify the analysis in Claim 5.3 to show that this algorithm finds an ordering with cost at most $\frac{1}{2}$ more than the cost of the optimal ordering. We may observe that this difference is asymptotically tight by adding a coin with probability of heads $0.5 - \epsilon$ to the instance in Claim 5.2.

6 Adaptivity gap. The *adaptivity gap* is the worst-case ratio of the cost of the optimal non-adaptive ordering and the cost of the optimal adaptive strategy. We first show this is at least $1.2 - O(\frac{1}{2^n})$, where n is the number of coins.

CLAIM 6.1 (Lower bound of Theorem 1.3). *The adaptivity gap is at least* $1.2 - O\left(\frac{1}{2^n}\right)$.

Proof. Recall that the cost of a non-adaptive ordering a is

$$cost(a) = 2 + \sum_{j \ge 3} \left(\prod_{i=1}^{j-1} p_{a(i)} + \prod_{i=1}^{j-1} \overline{p}_{a(i)}) \right) .$$

Consider one coin with 0 probability of heads and $n-1$ coins with $\frac{1}{2}$ probability of heads. Any non-adaptive ordering is determined by the position of the coin with 0 probability of heads. Suppose it is at position k. Then the cost is thus

$$2 + \sum_{j=3}^{k} (0.5^{j-1} + 0.5^{j-1}) + \sum_{j=k+1}^{n} (0.5^{j-2} + 0) = 2 + \sum_{j=3}^{k} 0.5^{j-2} + \sum_{j=k+1}^{n} 0.5^{j-2} = 2 + \sum_{j=3}^{n} 0.5^{j-2} .$$

This sum is independent of k; thus, all non-adaptive orderings have the same cost of

$$2 + \frac{0.5 - 0.5^{n-1}}{0.5} = 3 - \frac{1}{2^{n-2}} .$$

Consider the adaptive strategy that flips a coin with $\frac{1}{2}$ probability of heads, then chooses all other coins optimally. The strategy terminates unless all observed coins are tails, or we run out of coins to flip. The cost of this strategy is

$$2 + \sum_{j=3}^{n} 0.5^{j-1} = 2 + \frac{0.5 - 0.5^{n}}{0.5} = 2.5 - \frac{1}{2^{n-1}} .$$

So the adaptivity gap is at least

$$\frac{3 - 0.5^{n-2}}{2.5 - 0.5^{n-1}} = \frac{3}{2.5} - \frac{0.5^{n-2} - \frac{3}{2.5} \cdot 0.5^{n-1}}{2.5 - 0.5^{n-1}} = 1.2 - \frac{0.8 \cdot 0.5^{n-1}}{2.5 - 0.5^{n-1}} = 1.2 - O\left(\frac{1}{2^n}\right) . \qquad \square$$

Our proof of the upper bound uses a structure in any ordering we call *crossover point*. Each coin before or at the crossover point has a good chance of terminating the sequence, but this is not true after this point.

DEFINITION 6.2 (Crossover point). *Fix an ordering a. The* crossover point *of a is the largest position t where for all $2 \le x \le t$, the probability of terminating after flipping the coin at position $a(x)$, assuming termination has not yet occurred, is at least $\frac{1}{2}$. If there is no such t, we say $t = 1$.*

The crossover point t of any ordering a naturally induces a partition of $[a(1), \ldots, a(n)]$ with a first *region* $[a(1), \ldots, a(t)]$ and a (possibly empty) second region $[a(t+1), \ldots, a(n)]$. In the first region, the sequence is likely to terminate quickly. By Claim 3.3, the crossover point of an optimal ordering guaranteed by Claim 3.4 occurs in the final block.

We now prove that the adaptivity gap is at most 1.5.

CLAIM 6.3 (Upper bound of Theorem 1.3). *The adaptivity gap is at most* 1.5.

Proof. We try to construct the following two orderings:

- The first coin has the smallest probability of heads. For every subsequent position, if the ordering is 0-biased, use the coin with *smallest* probability of heads at *least* $\frac{1}{2}$; else, use the coin with *smallest* probability of heads.
- The first coin has the *largest* probability of heads. For every subsequent position, if the ordering is 1-biased, use the coin with *largest* probability of heads at *most* $\frac{1}{2}$; else, use the coin with *largest* probability of heads.

We explain why one of these orderings can be successfully created. Suppose in the first ordering, we run out of coins with probability at least $\frac{1}{2}$. Since the ordering is 0-biased at this point, and all other coins have probability of heads less than $\frac{1}{2}$, we have $\prod_{i=1}^{n} p_i < \prod_{i=1}^{n} \overline{p}_i$. However, if in the second ordering, we run out of coins with probability *at most* $\frac{1}{2}$, then $\prod_{i=1}^{n} p_i > \prod_{i=1}^{n} \overline{p}_i$. Only one of these events can occur.

These orderings are constructed to guarantee the following property. Suppose the crossover point is t. Then for all $s \geq t$, the smallest (or the largest) s coins are in the first s positions. This makes the last few coins in the ordering look similar to the adaptive strategy.

Choose an ordering a that was successfully created. By definition, the probability of terminating the sequence is at least $\frac{1}{2}$ for any coin in position at most t. The cost of this algorithm is thus at most

$$cost(a) = 2 + \sum_{j=3}^{n} (z_j^1(a) + z_j^0(a)) \leq 2 + \left(\frac{1}{2} + \cdots + \frac{1}{2^{t-1}}\right) + \sum_{j=t+2}^{n} (z_j^1(a) + z_j^0(a)) .$$

For the rest of this proof, we assume that we have chosen the first ordering; the argument if we choose the second ordering is symmetric. By our assumption, all coins in the second region (if it exists) have probability of heads greater than $\frac{1}{2}$. So then $z_{j+1}^0(a) = \overline{p}_j \cdot z_j^0(a) \leq \frac{1}{2} z_j^0(a)$ for all $j > t$. So $\sum_{j=t+2}^{n} z_j^0(a) \leq z_{t+1}^0(a) \leq \frac{1}{2^{t-1}}$. So the algorithm has cost at most

$$cost(a) \leq 1 + \left(1 + \frac{1}{2} + \cdots + \frac{1}{2^{t-1}}\right) + \frac{1}{2^{t-1}} + \sum_{j=t+2}^{n} z_j^1(a) = 3 + \sum_{j=t+2}^{n} z_j^1(a) .$$

Meanwhile, the cost of any adaptive strategy for the Unanimous Vote problem is lower-bounded by the cost of the best strategy to see at least one tails while flipping at least two coins, which is $2 + \sum_{j=2}^{n-1} \prod_{i=1}^{j} p_i$ (i.e. flip the coins in increasing order of p_i). So the adaptivity gap is at most

$$\frac{3 + \sum_{j=t+2}^{n} z_j^1(a)}{2 + \sum_{j=2}^{n-1} \prod_{i=1}^{j} p_i} \leq \max\left(\frac{3}{2}, \frac{\sum_{j=t+2}^{n} z_j^1(a)}{\sum_{j=2}^{n-1} \prod_{i=1}^{j} p_i}\right) .$$

By construction, for all $j \geq t$, we have $z_{j+1}^1(a) = \prod_{i=1}^{j} p_i$. So the second ratio equals

$$\frac{\sum_{j=t+1}^{n-1} \prod_{i=1}^{j} p_i}{\sum_{j=2}^{n-1} \prod_{i=1}^{j} p_i} ,$$

which is at most 1 since the crossover point $t \geq 1$. So the adaptivity gap is at most $\max(\frac{3}{2}, 1) = \frac{3}{2}$. ∎

In the full version of the paper, we use the techniques in Claim 6.3 to sharpen the upper bound to $1.2 + o(1)$.

7 Open questions. We conclude by highlighting open questions related to the Unanimous Vote problem.

First, as discussed in Subsection 1.3, SBFE problems are often studied with *costs*, and it is natural to consider the problem in that setting, i.e., where it costs c_i to flip coin i and the goal is to minimize the expected cost of coins flipped before termination. It is unclear if our algorithm would generalize to this setting.

It is also interesting to consider the unit-cost setting for other symmetric Boolean functions beyond the Unanimous Vote (i.e., not-all-equal) function. For example, recent work provides a PTAS for the unit-cost, non-adaptive SBFE problem for the k-of-n function [NRS25]. We conjecture that there is a polynomial-time exact algorithm. More generally, It is an open question whether there is a polynomial-time algorithm for the unit-cost SBFE problem for arbitrary symmetric functions, in either the adaptive or the non-adaptive setting. Some related work in the adaptive setting appeared in the information theory literature, motivated by problems involving distributed sensors [AJO11, DJO$^+$12, KK13].

Finally, the interchange arguments used in our analysis suggest that there may be a simple local-search algorithm for the Unanimous Vote problem based on interchanges. It would be interesting to find such an algorithm that converges to an optimal solution in polynomial time. We note that the algorithm would need to interchange non-adjacent elements of the ordering; under adjacent interchanges only, there can be locally optimal solutions that are not globally optimal.

Acknowledgements. Thanks to R. Teal Witter for collaborating on early stages of this project. K.M acknowledges support from AFOSR (FA9550-21-1-0008). This material is based upon work partially supported by the National Science Foundation Graduate Research Fellowship under Grant No. 2140001. C.M. was partially supported by NSF Award No. 2045590. L.H. was partially supported by NSF Award No. 1909335.

References

[AHKÜ17] Sarah R. Allen, Lisa Hellerstein, Devorah Kletenik, and Tonguç Ünlüyurt. Evaluation of Monotone DNF Formulas. *Algorithmica*, 77(3):661–685, 2017. URL https://doi.org/10.1007/s00453-015-0092-9.

[AJO11] Jayadev Acharya, Ashkan Jafarpour, and Alon Orlitsky. Expected query complexity of symmetric boolean functions. In *49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 26–29, 2011. URL https://doi.org/10.1109/Allerton.2011.6120145.

[BD81] Yosi Ben-Dov. Optimal Testing Procedures for Special Structures of Coherent Systems. *Management Science*, 27(12):1410–1420, 1981. URL http://www.jstor.org/stable/2630998.

[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. URL https://doi.org/10.1137/1.9781611976465.99.

[BSZ19] Domagoj Bradac, Sahil Singla, and Goran Zuzic. Near Optimal Adaptivity Gaps for Stochastic Multi-Value Probing. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145, pages 49:1–49:21, 2019. URL https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.49.

[BÜ00] Endre Boros and Tonguç Ünlüyurt. Sequential Testing of Series-Parallel Systems of Small Depth. In Manuel Laguna and José Luis González Velarde, editors, *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, pages 39–73. Springer, 2000. URL https://doi.org/10.1007/978-1-4615-4567-5_3.

[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 32nd annual ACM Symposium on Theory of Computing*, STOC '00, pages 582–591, New York, NY, USA, 2000. ACM. URL http://doi.acm.org/10.1145/335305.335382.

[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. URL https://doi.org/10.1016/j.dam.2010.05.016.

[CQK89] L. Cox, Y. Qiu, and W. Kuehner. Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations Research*, 21:1–29, 1989. URL http://dx.doi.org/10.1007/BF02022091.

[CSF90] M.-F. Chang, W. Shi, and W. K. Fuchs. Optimal diagnosis procedures for k-out-of-n structures. *IEEE Transactions on Computers*, 39(4):559—-564, 1990. URL https://doi.org/10.1109/12.54850.

[DHK16] Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation Algorithms for Stochastic Submodular Set Cover with Applications to Boolean Function Evaluation and Min-Knapsack. *ACM Trans. Algorithms*, 12(3), 2016. URL https://doi.org/10.1145/2876506.

[DHM+25] Feyza Duman Keles, Lisa Hellerstein, Kunal Marwaha, Christopher Musco, and Xinchen Yang. An Exact Algorithm for the Unanimous Vote Problem, 2025. arXiv:2510.16678.

[DJO+12] Hirakendu Das, Ashkan Jafarpour, Alon Orlitsky, Shengjun Pan, and Ananda Theertha Suresh. On the query computation and verification of functions. In *Proceedings of the 2012 IEEE International Symposium on Information Theory (ISIT)*, pages 2711–2715, 2012. URL https://doi.org/10.1109/ISIT.2012.6284010.

[GGHK18] Dimitrios Gkenosis, Nathaniel Grammel, Lisa Hellerstein, and Devorah Kletenik. The Stochastic Score Classification Problem. In *Proceedings of the 26th European Symposium on Algorithms (ESA)*, pages 36–1, 2018. URL https://doi.org/10.4230/LIPIcs.ESA.2018.36.

[GGHK22] Dimitrios Gkenosis, Nathaniel Grammel, Lisa Hellerstein, and Devorah Kletenik. The Stochastic Boolean Function Evaluation problem for symmetric Boolean functions. *Discrete Applied Mathematics*, 309:269–277, 2022. URL https://doi.org/10.1016/j.dam.2021.12.001.

[GGN24] Rohan Ghuge, Anupam Gupta, and Viswanath Nagarajan. Nonadaptive Stochastic Score Classification and Explainable Half-Space Evaluation. *Operations Research*, 2024. URL https://doi.org/10.1287/opre.2023.0431.

[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. URL https://doi.org/10.1007/s00453-022-00982-4.

[GN13] Anupam Gupta and Viswanath Nagarajan. A Stochastic Probing Problem with Applications. In *Integer Programming and Combinatorial Optimization*, pages 205–216. Springer Berlin Heidelberg, 2013. URL https://doi.org/10.1007/978-3-642-36694-9_18.

[GNS17] Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Adaptivity gaps for stochastic probing: submodular and XOS functions. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1688–1702, 2017. URL https://doi.org/10.1137/1.9781611974782.111.

[HHL22] Felix Happach, Lisa Hellerstein, and Thomas Lidbetter. A General Framework for Approximating Min Sum Ordering Problems. *INFORMS J. Comput.*, 34(3):1437–1452, 2022. URL https://doi.org/10.1287/ijoc.2021.1124.

[HKLW22a] Lisa Hellerstein, Devorah Kletenik, Naifeng Liu, and R. Teal Witter. Adaptivity Gaps for the Stochastic Boolean Function Evaluation Problem. In *Approximation and Online Algorithms (WAOA)*, pages 190–210, 2022. URL https://doi.org/10.1007/978-3-031-18367-6_10.

[HKLW22b] Lisa Hellerstein, Devorah Kletenik, Naifeng Liu, and R. Teal Witter. Adaptivity Gaps for the Stochastic Boolean Function Evaluation Problem. In Parinya Chalermsook and Bundit Laekhanukit, editors, *Approximation and Online Algorithms - 20th International Workshop, WAOA 2022, Potsdam, Germany, September 8-9, 2022, Proceedings*, volume 13538 of *Lecture Notes in Computer Science*, pages 190–210. Springer, 2022. URL https://doi.org/10.1007/978-3-031-18367-6_10.

[HLS24] Lisa Hellerstein, Naifeng Liu, and Kevin Schewior. Quickly Determining Who Won an Election. In *Proceedings of the 15th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 287, pages 61:1–61:14, 2024. URL https://doi.org/10.4230/LIPIcs.ITCS.2024.61.

[HNT25] Blake Harris, Viswanath Nagarajan, and Rayen Tan. Sequential Testing with Subadditive Costs, 2025. URL https://doi.org/10.1007/978-3-031-93112-3_21.

[IK84] Toshihide Ibaraki and Tiko Kameda. On the optimal nesting order for computing N-relational joins. *ACM Trans. Database Syst.*, 9(3):482–502, 1984. URL https://doi.org/10.1145/1270.1498.

[KBZ86] Ravi Krishnamurthy, Haran Boral, and Carlo Zaniolo. Optimization of Nonrecursive Queries. In *VLDB*, pages 128–137, 1986. URL http://web.cs.ucla.edu/~zaniolo/papers/vldb86b.pdf.

[KK13] Hemant Kowshik and P. R Kumar. Optimal Computation of Symmetric Boolean Functions in Collocated Networks. *IEEE Journal on Selected Areas in Communications*, 31(4):639–654, 2013. URL https://doi.org/10.1109/JSAC.2013.130403.

[KKM05] H. Kaplan, E. Kushilevitz, and Y. Mansour. Learning with attribute costs. In *Symposium on the Theory of Computing*, pages 356–365, 2005. URL http://doi.acm.org/10.1145/1060590.1060644.

[Liu22] Naifeng Liu. Two 6-approximation Algorithms for the Stochastic Score Classification Problem, 2022. arXiv:2212.02370.

[NRS25] Mads Anker Nielsen, Lars Rohwedder, and Kevin Schewior. Non-Adaptive Evaluation of k-of-n Functions: Tight Gap and a Unit-Cost PTAS, 2025. arXiv:2507.05877.

[PRS23] Kalen Patton, Matteo Russo, and Sahil Singla. Submodular Norms with Applications To Online Facility Location and Stochastic Probing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275, pages 23:1–23:22, 2023. URL https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2023.23.

[PS24] Benedikt M Plank and Kevin Schewior. Simple algorithms for stochastic score classification with small approximation ratios. *SIAM Journal on Discrete Mathematics*, 38(3):2069–2088, 2024. URL https://doi.org/10.1137/22M1523492.

[SB97] S. Salloum and M.A. Breuer. Fast optimal diagnosis procedures for k-out-of-n: G systems. *IEEE Transactions on Reliability*, 46(2):283–290, 1997. URL https://doi.org/10.1109/24.589958.

[Sin18] Sahil Singla. The price of information in combinatorial optimization. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2523–2532, 2018. URL doi.org/10.1137/1.9781611975031.161.

[SS21] Danny Segev and Sahil Singla. Efficient Approximation Schemes for Stochastic Probing and Prophet Problems. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 793–794, 2021. URL https://doi.org/10.1145/3465456.3467614.

[Ünl04] Tonguç Ünlüyurt. Sequential testing of complex systems: a review. *Discrete Applied Mathematics*, 142(1-3):189–205, 2004. URL https://doi.org/10.1016/j.dam.2002.08.001.

[Ünl25] Tonguç Ünlüyurt. Sequential testing problem: A follow-up review. *Discrete Applied Mathematics*, 377:356–369, 2025. URL https://doi.org/10.1016/j.dam.2025.07.039.