Algorithmica manuscript No.
(will be inserted by the editor)

Algorithms for the Unit-Cost Stochastic Score Classification Problem

Nathaniel Grammel · Lisa Hellerstein* · Devorah Kletenik · Naifeng Liu

Submitted June 25, 2020

Abstract Consider the following Stochastic Score Classification problem. A doctor is assessing a patient's risk of developing a disease and can perform n different binary tests on the patient. The probability that test i is positive is p_i and the outcomes of the n tests are independent. A patient's score is the total number of positive tests. Possible scores thus range between 0 and n. This range is divided into subranges, corresponding to risk classes (e.g., LOW, MEDIUM, or HIGH risk). Each test has an associated cost. To reduce testing cost, instead of performing all tests and determining an exact score, the doctor can perform tests sequentially and stop testing when it is possible to determine the patient's risk class. The problem is to determine the order in which the doctor should perform the tests, so as to minimize expected testing cost. We address the unit-cost case of the Stochastic Score Classification problem, and provide polynomial-time approximation algorithms for adaptive and non-adaptive versions of the problem. We also pose a number of open questions.

Keywords approximation algorithms, symmetric Boolean functions, stochastic probing, sequential testing, adaptivity

Partial support for this work came from NSF Award IIS-1217968 (all authors), NSF Award IIS-1909335 (L. Hellerstein), and a PSC-CUNY Award, jointly funded by The Professional Staff Congress and The City University of New York (D. Kletenik).

Nathaniel Grammel
University of Maryland, College Park, MD, USA E-mail: ngrammel@cs.umd.edu

*Lisa Hellerstein (corresponding author)
NYU Tandon School of Engineering, Brooklyn, NY, USA E-mail: lisa.hellerstein@nyu.edu

Devorah Kletenik
Brooklyn College (CUNY), Brooklyn, NY, USA E-mail: kletenik@sci.brooklyn.cuny.edu

Naifeng Liu
CUNY Graduate Center, New York, NY, USA E-mail: nliu@gradcenter.cuny.edu

1 Introduction

We consider the following *Stochastic Score Classification (SSClass)* problem. A doctor wants to assess a patient's risk of developing a certain disease and can perform n different binary tests on the patient. Test i has known probability p_i of yielding a positive result and the results of the n tests are independent. The patient's score is the total number of positive test results. Test results are accurate, so the same test is never performed more than once on the patient. The doctor needs to classify the patient into one of B risk classes, depending on the score (e.g., LOW, MEDIUM, and HIGH risk). Each of these classes corresponds to a contiguous range of scores. Each test performed incurs a testing cost; we assume each test has unit cost.

To reduce testing cost, instead of performing all tests on a patient and computing an exact score, the doctor will perform them sequentially, stopping when the class becomes a foregone conclusion. For example, suppose there are 10 tests and the MEDIUM class corresponds to a score between 4 and 7, inclusive. If the doctor performed 8 tests, of which 5 were positive, the doctor would not perform the remaining 2 tests, because the patient's risk class will be MEDIUM regardless of the outcome of the 2 remaining tests. The problem is to determine the optimal (adaptive or non-adaptive) order in which to perform the tests, so as to minimize the expected testing cost. Since we have assumed unit-cost tests, this is equivalent to minimizing the expected number of tests.

In this paper, we present polynomial-time, constant-factor approximation algorithms for three SSClass problems, in the unit-cost case: the general SSClass problem (for an arbitrary score classification problem), the SSClass problem for the *Unanimous Vote* function, and the SSClass problem for the k-of-n function.

Formally, the Stochastic Score Classification (SSClass) problem is as follows. Given $B+1$ integers $0 = \alpha_1 < \alpha_2 < \ldots < \alpha_B < \alpha_{B+1} = n+1$, let *class* j correspond to the *scoring interval* $[\alpha_j, \alpha_j + 1, \ldots, \alpha_{j+1} - 1]$. Thus B is the number of classes. The α_j values define an associated pseudo-Boolean *score classification function* $f : \{0,1\}^n \to \{1, \ldots, B\}$, such that $f(X_1, \ldots, X_n)$ is the class whose scoring interval contains the *score* $\sum_i X_i$. For $i \in \{1, \ldots, n\}$, X_i is an independent Bernoulli random variable. Probability $p_i := Prob[X_i = 1]$ is given, and $0 < p_i < 1$. The value of X_i can only be determined by performing a test (or asking a query), which incurs a given testing cost c_i, where $c_i > 0$. In this paper, we consider the unit-cost version of the SSClass problem, so we assume each $c_i = 1$.

An *evaluation strategy* for f is a sequential adaptive or non-adaptive order in which to perform the tests. Testing must continue until the value of f can be determined, i.e., until the value of f would be the same, no matter which results were obtained for the remainder of the n tests. In an *adaptive strategy*, the choice of the next test can depend on the outcomes of previous tests. An adaptive strategy corresponds to a decision tree. A *non-adaptive strategy* is a permutation of the tests. With a non-adaptive strategy, testing proceeds in the order specified by the permutation until the value of f can be determined.

The goal of an algorithm for the SSClass problem is to find an evaluation strategy for f with minimum expected total testing cost.

We consider both adaptive and non-adaptive versions of these problems, in which the algorithm must find optimal adaptive or non-adaptive strategies, respectively. In the adaptive version, the algorithm does not need to output the full decision tree corresponding to the adaptive strategy (which may be of exponential size). It is sufficient if the algorithm can implement the strategy in an on-line setting, by computing the next test to perform, based on the results obtained from the previous tests.

Before describing our results in detail, we discuss relevant prior work.

2 Prior work

The unit-cost adaptive SSClass problem was previously studied in the information theory literature [5,1,13]. The main novel contribution there was to establish an equivalence between verification and evaluation. Specifically, Das et al. [5] showed that for the unit-cost adaptive SSClass problem, optimal expected verification cost equals optimal expected evaluation cost.

In the algorithms literature, Deshpande et al. presented a dual-greedy 3-approximation algorithm solving the *Stochastic Boolean Function Evaluation* (SBFE) problem for linear threshold functions [6]. The general SBFE problem is similar to the adaptive SSClass problem, but instead of evaluating a given score classification function f defined by inputs α_j, you need to evaluate a given Boolean function f. The SBFE problem for linear threshold functions is equivalent to a *weighted* version of the adaptive SSClass problem with only two classes ($B = 2$), which allows each variable X_i to have an associated integer weight a_i. The score is then the sum of the weights a_i of the tests i with positive outcomes.

A k-of-n function is a Boolean function f such that $f(x) = 1$ iff $x_1 + \ldots + x_n \geq k$. The SBFE problem for k-of-n functions is equivalent to the two-class SSClass problem. It has been studied previously in the VLSI testing literature. There is an elegant algorithm for the adaptive version of the problem, due to Salloum, Breuer, and (independently) Ben-Dov, that computes an optimal strategy [14,2,15,4].

While we have described the SSClass problem in the context of assessing disease risk, score classification is also used in other contexts, such as assigning letter grades to students, giving a quality rating to a product, or deciding whether a person charged with a crime should be released on bail. In Machine Learning, the focus is on learning the score classification function [20,18,12, 22,21]. In contrast, here our focus is on reducing the cost of evaluating the classification function. We note that the SSClass problem differs from many other stochastic probing problems previously considered (e.g. [17,11]) because of the requirement that testing must continue until the unique interval containing the score has been determined.

3 Results and open questions

A table summarizing our approximation bounds can be found at the end of this section.

Our first algorithm gives a 2-approximation for both the adaptive and non-adaptive versions of the unit-cost SSClass problem. This is the best approximation bound known for both versions of the problem.

Our second algorithm is for a special case of the unit-cost, non-adaptive version of the SSClass problem, where the problem is to evaluate what we call the *Unanimous Vote* function. The Unanimous Vote function outputs POSITIVE if $X_1 = \ldots = X_n = 1$, NEGATIVE if $X_1 = \ldots = X_n = 0$, and UNCERTAIN otherwise. Equivalently, it is a score classification function with $B = 3$ and scoring intervals $\{0\}, \{1, \ldots, n-1\}$ and $\{n\}$. Our algorithm produces a φ-approximation to the optimal solution, where $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ is the golden ratio. The Unanimous Vote function is closely related to the Boolean *consensus* function, whose output is 1 iff its input variables are all equal, and to its complement, the Boolean *Not-All-Equal* function. Our algorithm can also be viewed as providing a φ-approximation for a non-adaptive version of the unit-cost SBFE problem for these Boolean functions.

Finally, our third algorithm is a 1.5-approximation algorithm for the unit-cost, non-adaptive version of the SSClass problem where $B = 2$. Equivalently, it is a 1.5-approximation algorithm solving a non-adaptive version of the unit-cost SBFE problem for k-of-n functions.

For the Unanimous Vote and k-of-n functions, there are polynomial-time exact algorithms for the adaptive versions (see Table 1). Our algorithms provide polynomial-time approximation algorithms for the non-adaptive version.

The proofs of our approximation bounds imply upper bounds of 2, φ, and 1.5 for the adaptivity gaps of the corresponding problems.

Our first algorithm is simple. It outputs the strategy that performs a round robin between two test orderings: one which performs the tests in decreasing p_i order, and the other which performs them in increasing p_i order. Proving that this achieves a 2-approximation requires some care.

Our φ-approximation algorithm for the Unanimous Vote function is based on an approximately optimal strategy for performing the remaining $n-1$ tests, assuming the first test is given. This strategy performs the remaining tests by initiating a round robin between the increasing and decreasing p_i orderings, and then stopping the round robin at a carefully chosen point to commit to one of the two orderings, while abandoning the other. Our algorithm for the Unanimous Vote function calculates the expected cost of this strategy for each possible first test x_i (which is a straightforward calculation), and chooses the one with lowest expected cost.

To prove the 1.5 bound on k-of-n functions, we observe that the round robin strategy underperforms relative to using either the increasing p_i ordering alone, or the decreasing p_i ordering alone, whichever is cheaper.

There remain many intriguing open questions related to SSClass problems. The first, and most fundamental, is whether the (adaptive or non-adaptive)

SSClass problem is NP-hard, even in the unit-cost case. It is unclear whether this problem will be easy to resolve. We note that the NP-hardness of the stochastic evaluation problem for Boolean read-once formulas has been open since the 1970's (cf. [10,19]). Other open questions concern lower bounds on approximation factors, and bounds on adaptivity gaps.

Some of the results in this paper appeared in preliminary form in a conference paper [8]. The conference paper also included preliminary versions of results on the SSClass problem with arbitrary costs; final versions of these results, for associated Boolean function evaluation problems, have recently appeared in a journal [9]. The results include polynomial-time $O(\log n)$ and $B-1$ approximation algorithms for adaptive and non-adaptive versions of the SSClass problem with arbitrary costs, and a simple exact algorithm solving the adaptive SSClass problem for the special case of the Unanimous Vote function (we present the unit-cost version of this algorithm in Section 6). Additionally, an example was given which demonstrates that the equivalence between optimal expected verification cost and optimal expected evaluation cost in the unit-cost case does not extend to arbitrary costs.

In work that appeared subsequent to submission of this paper, Ghuge et al. gave the first polynomial-time, constant-factor approximation algorithm for both the adaptive and non-adaptive versions of the SSClass problem with arbitrary costs [7]. Their algorithm also solves the weighted version of the SSClass problem. Although Ghuge et al. do not try to optimize the constant they obtain for their constant-factor approximation, the approach used in their proof results in an approximation factor that is greater than 16. This contrasts with the factor of 2 approximation we present in this paper for the less general (unweighted) unit-cost SSClass problem.

Table 1 Results for the Unit-Cost SSClass Problem

	adaptive	non-adaptive
general (arbitrary score classification function)	2-approx [Sec. 5]	2-approx [Sec. 5]
unanimous vote function	exact algorithm [Sec. 6]	φ-approx [Sec. 6]
k-of-n function	exact algorithm [14,2, 15,4,16]	1.5-approx [Sec. 7]

4 Preliminaries

A *partial assignment* is a vector $b \in \{0,1,*\}^n$. We use f^b to denote the restriction of function $f(x_1,\ldots,x_n)$ to the variables x_i with $b_i = *$, produced by fixing the remaining variables x_i according to their values b_i. We call f^b the function *induced from* f *by partial assignment* b. For $\ell \in \{0,1\}$, we use $N_\ell(b)$ to denote $|\{i \mid b_i = \ell\}|$, the number of entries of b that are set to ℓ.

A partial assignment encodes what information is known at a given point in a sequential testing environment. Specifically, for partial assignment $b \in \{0,1,*\}^n$, $b_i = *$ indicates that test i has not yet been performed, otherwise b_i equals the outcome of test i. We sometimes refer to performing test i as testing bit x_i.

Suppose the probabilities p_i for the n tests are fixed. We define the expected cost of an adaptive evaluation strategy for $f\colon \{0,1\}^n \to \{0,1\}$ or $f\colon \{0,1\}^n \to \{1,\ldots,B\}$ as follows. (The definitions for non-adaptive strategies are analogous.) Given an adaptive evaluation strategy $\mathcal{A}$ for f, and an assignment $x \in \{0,1\}^n$, we use $C(\mathcal{A},x)$ to denote the sum of the costs of the tests performed in using $\mathcal{A}$ on x. The expected cost of $\mathcal{A}$ is $\sum_{x\in\{0,1\}^n} C(\mathcal{A},x)p(x)$, where $p(x) = \prod_{i=1}^n p^{x_i}(1-p)^{1-x_i}$ is the probability of x. We say that $\mathcal{A}$ is an *optimal* adaptive evaluation strategy for f if it has minimum possible expected cost.

Let L denote the range of f, and for $\ell \in L$, let $\mathcal{X}_\ell = \{x \in \{0,1\}^n \mid f(x) = \ell\}$. An *adaptive verification strategy* for f consists of $|L|$ adaptive evaluation strategies $\mathcal{A}_\ell$ for f, one for each $\ell \in L$. The expected cost of the verification strategy is $\sum_{\ell\in L}\left(\sum_{x\in\mathcal{X}_\ell} C(\mathcal{A}_\ell,x)p(x)\right)$ and it is optimal if it minimizes this expected cost.

If $\mathcal{A}$ is an evaluation strategy for f, we call $\sum_{x\in\mathcal{X}_\ell} C(\mathcal{A},x)p(x)$ the ℓ-cost of $\mathcal{A}$. For $\ell \in L$, we say that $\mathcal{A}$ is *ℓ-optimal* if it has minimum possible ℓ-cost. In an optimal verification strategy for f, each component evaluation strategy $\mathcal{A}_\ell$ must be ℓ-optimal.

A function $f : \{0,1\}^n \to L$ is *symmetric* if its output on $x \in \{0,1\}^n$ depends only on $N_1(x)$. The *value vector* for such a function f is the $n+1$ dimensional vector v^f, indexed from 0 to n, whose jth entry v^f_j is the value of f on inputs x where $N_1(x) = j$. We partition value vector v^f into *blocks*. A block is a maximal subvector of v^f consisting of contiguous entries having the same value. Using B to denote the number of blocks of the value vector, we define $\alpha_1,\ldots\alpha_B$ to be the minimum indices of each of the blocks, where $0 = \alpha_1 < \alpha_2 < \ldots < \alpha_B$, and we define $\alpha_{B+1} = n+1$. Block i of v^f is the subvector of v^f containing the entries in $[\alpha_i, \alpha_{i+1})$. When f is a score classification function, the blocks correspond to the score intervals.

We say that assignment x is in block i of v^f if $N_1(x)$ is in the interval $[\alpha_i, \alpha_{i+1})$.

With each block i of v^f, we associate a function f^i, where $f^i(x) = 1$ if x is in block i, and $f^i(x) = 0$ otherwise. A *verification strategy for block i* is an evaluation strategy for f^i. An *optimal verification strategy for block i* is an evaluation strategy for f^i with minimum 1-cost.

We will use the following simple fact.

Fact 1. *For real numbers $a, b \geq 0$ and $c, d > 0$, $\frac{a+b}{c+d} \leq \max\{\frac{a}{c}, \frac{b}{d}\}$.*

5 A 2-approximation for the unit-cost SSClass problem

We present and analyze a simple strategy for the unit-cost non-adaptive SSClass problem. We call it the *Up-Down* strategy.

In this section we will assume that the variables x_i are indexed such that $p_1 \geq p_2 \geq \ldots \geq p_n$.

The Up-Down Strategy: Alternate (round robin) between the strategy that tests the x_i's in the order $x_1, \ldots, x_n$, and the strategy that tests the x_i's in the reverse order, until the test outcomes are sufficient to determine the value of f.

Thus if n is even, the Up-Down strategy executes the non-adaptive strategy represented by the following permutation:

$$x_1 x_n x_2 x_{n-1} \ldots x_{n/2} x_{n/2+1}$$

We will show that the expected cost of the Up-Down strategy is within a factor of 2 of the expected cost of the optimal adaptive strategy for the unit-cost SSClass problem. It follows that it is also within a factor of 2 of the expected cost of the optimal non-adaptive strategy.

At a high level, our approach is to partition the 2^n possible assignments according to the blocks to which they belong. For each block, we consider the contribution made by the assignments belonging to that block, to the total expected cost of a strategy. In order to compute the value of the function on the assignments in a block, the strategy must verify that the assignments belong to that block. That is, it must verify that the assignments have at least a certain number of ones, and at least a certain number of zeros. We show that for each block, the contribution made by the assignments in that block, to the expected cost of the Up-Down strategy, is at most twice their contribution to the expected cost of the optimal strategy. By summing over the blocks, we are then able to show that the expected cost of the Up-Down strategy is at most twice the expected cost of the optimal strategy.

We will make use of the following simple fact.

Fact 2. *If X and Y are random variables, and for all real numbers k, $P[X \leq k] \leq P[Y \leq k]$, then $E[X] \geq E[Y]$.*

We note that the above fact is one of the motivations behind the definition of "stochastic dominance" in decision theory. (If $P[X \leq k] \leq P[Y \leq k]$ for all k and the inequality is strict for some k, then X is said to stochastically dominate Y.)

Theorem 1 *The Up-Down strategy is a 2-approximation algorithm for both the adaptive and non-adaptive versions of the unit-cost SSClass problem.*

Proof. Consider an instance of the unit-cost SSClass problem. Let S be an optimal adaptive strategy for evaluating f. Let Z denote the Up-Down strategy.

We will show that the expected cost of Z is within a factor of 2 of the expected cost of S, and is thus a 2-approximation for both the adaptive and non-adaptive versions of the problem. Equivalently, we show that

$$\sum_{x \in \{0,1\}^n} p(x)C(Z,x) \leq 2 \sum_{x \in \{0,1\}^n} p(x)C(S,x) \tag{1}$$

Let M_j denote the set of assignments in block j of v^f. We will prove that for each block j,

$$\sum_{x \in M_j} p(x)C(Z,x) \leq 2 \sum_{x \in M_j} p(x)C(S,x) \tag{2}$$

Summing over all blocks j then yields (1), proving the theorem.

We now prove (2). Consider block j of v^f. Let $\beta_j = \alpha_{j+1} - 1$. The assignments in block j are those that have at least α_j ones and at least $n - \beta_j$ zeros.

Consider the execution of optimal strategy S on an assignment x in block j. Strategy S must continue performing tests until it has enough information to determine the value of $f(x)$. Therefore, during its execution on x, at least α_j of the tests it performs must have outcome 1, and at least $n - \beta_j$ must have outcome 0. Otherwise, it will not have enough information to determine whether x is in block j or in an adjacent block, and therefore will not have enough information to determine $f(x)$. Thus for all assignments x in block j, $C(S,x) \geq \alpha_j + n - \beta_j$.

The *length* of block j equals $\beta_j - \alpha_j + 1$. We have two cases, depending on the length of block j.

Case 1: Block j has length at most $n/2$.

In this case, for all $x \in M_j$, $C(S,x) \geq \alpha_j + n - \beta_j \geq n/2 + 1$. Since the Up-Down strategy (and, indeed *any* strategy) will perform at most n tests, we have $C(Z,x) < 2C(S,x)$ for all $x \in M_j$, so (2) holds.

Case 2: Block j has length greater than $n/2$.

For this case, we will show that the following inequality holds for all $k \geq 0$.

$$\Pr[(C(S,x) \leq k) \wedge (x \in M_j)] \leq \Pr[(C(Z,x) \leq 2k) \wedge (x \in M_j)] \tag{3}$$

Equivalently,

$$\Pr[(C(S,x) \leq k) \mid x \in M_j)] \leq \Pr[(C(Z,x)/2 \leq k) \mid x \in M_j]$$

Therefore, by Fact 2,

$$E[C(Z,x)/2 \mid x \in M_j)] \leq \mathbb{E}[C(S,x) \mid x \in M_j]$$

and (2) follows immediately.

It remains for us to prove (3). For convenience, let $\nu_1 := \alpha_j$ and $\nu_0 := n - \beta_j$. If $k < \nu_1 + \nu_0$, then $\Pr[(C(S,x) \leq k) \wedge (x \in M_j)] = 0$, because for $x \in M_j$, $C(S,x) \geq \nu_1 + \nu_0$. So for $k < \nu_1 + \nu_0$, (3) trivially holds. Also, if $k \geq n/2$,

then $2k \geq n$ and so clearly $\Pr[(C(Z,x) \leq 2k) \wedge (x \in M_j)] = P[x \in M_j]$. Thus, when $k \geq n/2$, (3) again trivially holds.

So suppose $\nu_0 + \nu_1 \leq k < n/2$. Consider running the Up-Down strategy Z for its first $2k$ steps. This consists of testing variables $x_1, x_2, \ldots, x_k$ (i.e., the k tests with highest p_i) as well as the variables $x_n, x_{n-1}, \ldots, x_{n-k+1}$ (i.e., the k tests with the lowest p_i). Let $N_1(Z_k^{\uparrow}, x)$ denote the number of ones among the variables $x_1, x_2, \ldots, x_k$ (i.e., the number of ones obtained from performing the k tests with highest p_i) and let $N_0(Z_k^{\downarrow}, x)$ denote the number of zeros among the variables $x_n, x_{n-1}, \ldots, x_{n-k+1}$ (i.e., the number of zeros obtained from performing the k tests with lowest p_i). Similarly, let $N_0(Z_{2k}, x)$ and $N_1(Z_{2k}, x)$ denote the number of zeros and ones obtained (respectively) from performing the first $2k$ tests of Z. If $x \in M_j$, then Z terminates within its first $2k$ tests if and only if at least ν_1 of those tests have outcome 1, and at least ν_0 have outcome 0. Further, on a given assignment x, Z can only perform at least ν_1 tests with outcome 1 and at least ν_0 tests with outcome 0 if $x \in M_j$.

We therefore have that:

$$\begin{aligned} & \Pr[(C(Z,x) \leq 2k) \wedge (x \in M_j)] \\ = \; & \Pr[(N_0(Z_{2k},x) \geq \nu_0) \wedge (N_1(Z_{2k},x) \geq \nu_1)] \\ \geq \; & \Pr[(N_0(Z_k^{\downarrow},x) \geq \nu_0) \wedge (N_1(Z_k^{\uparrow},x) \geq \nu_1)] \\ = \; & \Pr[N_0(Z_k^{\downarrow},x) \geq \nu_0] \Pr[N_1(Z_k^{\uparrow},x) \geq \nu_1] \end{aligned} \tag{4}$$

where the inequality follows from the fact that $N_0(Z_{2k}, x) \geq N_0(Z_k^{\downarrow}, x)$ and $N_1(Z_{2k}, x) \geq N_1(Z_k^{\uparrow}, x)$, and the final equality follows from the independence of tests and the fact that $\{x_1, \ldots, x_k\}$ and $\{x_{n-k+1}, \ldots x_n\}$ are disjoint (since $k < n/2$ implies that $k < n - k < n - k + 1$).

Next, we consider the first k steps of the optimal strategy S. Let $N_0(S_k, x)$ and $N_1(S_k, x)$ denote the number of zeros and ones (respectively) obtained from the first k tests of S on assignment x. Clearly, $\Pr[(C(S,x) \leq k) \wedge (x \in M_j)] = \Pr[(N_0(S_k,x) \geq \nu_0) \wedge (N_1(S_k,x) \geq \nu_1)]$.

Clearly

$$\Pr[N_0(Z_k^{\downarrow},x) \geq \nu_0] \geq \Pr[N_0(S_k,x) \geq \nu_0] \tag{5}$$

$$\Pr[N_1(Z_k^{\uparrow},x) \geq \nu_1] \geq \Pr[N_1(S_k,x) \geq \nu_1] \tag{6}$$

because in order to maximize the probability of obtaining at least ν_1 ones (resp. ν_0 zeros) using k of the n tests, we must perform the k tests with the highest (resp. lowest) p_i.

Let A_{11} denote the event $[(N_0(S_k,x) \geq \nu_0) \wedge (N_1(S_k,x) \geq \nu_1)]$, A_{01} denote the event $[(N_0(S_k,x) < \nu_0) \wedge (N_1(S_k,x) \geq \nu_1)]$, A_{10} denote the event $[(N_1(S_k,x) < \nu_1) \wedge (N_0(S_k,x) \geq \nu_0)]$, and A_{00} denote the event $[(N_0(S_k,x) < \nu_0) \wedge (N_1(S_k,x) < \nu_1)]$. We have assumed that $k \geq \nu_0 + \nu_1$, so by performing k tests, one is guaranteed to obtain at least ν_0 zeros or at least ν_1 ones. So

$\Pr[A_{00}] = 0$. We have:

$$\begin{aligned}
\Pr[A_{11}] &= 1 - \Pr[A_{10}] - \Pr[A_{01}] - \Pr[A_{00}] \\
&= 1 - \Pr[A_{10}] - \Pr[A_{01}] \\
&\leq 1 - \Pr[A_{10}] - \Pr[A_{01}] + \Pr[A_{10}] \Pr[A_{01}] \qquad (7) \\
&= (1 - \Pr[A_{01}])(1 - \Pr[A_{10}]) \\
&= \Pr[N_0(S_k, x) \geq \nu_0] \Pr[N_1(S_k, x) \geq \nu_1]
\end{aligned}$$

Combining (4) with (5),(6), and (7), we get:

$$\begin{aligned}
\Pr[(C(Z, x) \leq 2k) \wedge (x \in M_j)] &\geq \Pr[N_0(Z_k^{\downarrow}, x) \geq \nu_0] \Pr[N_1(Z_k^{\uparrow}, x) \geq \nu_1] \\
&\geq \Pr[N_0(S_k, x) \geq \nu_0] \Pr[N_1(S_k, x) \geq \nu_1] \\
&\geq \Pr[(N_0(S_k, x) \geq \nu_0) \wedge (N_1(S_k, x) \geq \nu_1)] \\
&= \Pr[(C(S, x) \leq k) \wedge (x \in M_j)]
\end{aligned}$$

□

6 The Unanimous Vote function

In this section, we present our φ-approximation algorithm solving the unit-cost non-adaptive SSClass problem, in the special case of the Unanimous Vote function.

Before presenting that algorithm, we first address the adaptive version of the problem.

6.1 Adaptive evaluation of the Unanimous Vote function

The optimal adaptive strategy for evaluating the Unanimous Vote function has a simple form. If the first test x_i has outcome 0, the induced function on the remaining variables is Boolean OR. In this case, it is clearly optimal to test the remaining bits non-adaptively in decreasing p_i order until either some test has outcome 1 (implying that the value of the Unanimous Vote function is UNCERTAIN), or all tests have been performed and have had outcome 0 (the value of the Unanimous Vote function is NEGATIVE). Symmetrically, if the first test x_i has outcome 1, it is optimal to perform the remaining tests non-adaptively in increasing p_i order, until either some test has outcome 0 (the value of the function is UNCERTAIN), or all tests have been performed and have had outcome 1 (the value of the function is POSITIVE). We will refer to this strategy as the *x_i-first adaptive strategy*.

Suppose we call the first test in the x_i-first adaptive strategy x_0, and renumber the remaining tests from 1 to $n-1$ so that $p_1 \geq p_2 \geq \ldots \geq p_{n-1}$. Then we can represent the strategy compactly as shown in Figure 1 (the leaves of the tree, reached when the function value can be determined, are not shown).

Algorithmically, then, the only difficulty in computing an optimal adaptive strategy is in determining which x_i should be tested first. To address this difficulty, the algorithm can just calculate the expected cost of the x_i-first adaptive strategy for each choice of x_i (which is a straightforward calculation) and then choose a strategy with minimum expected cost. Thus there is a polynomial-time algorithm solving the adaptive version of the SSClass problem for the Unanimous Vote function, with unit costs.

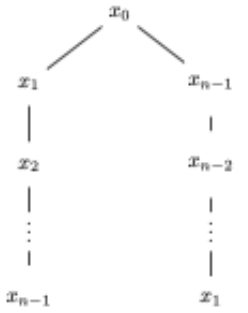

Fig. 1 Optimal adaptive strategy for the Unanimous Vote function, if x_0 is the first test

6.2 Non-adaptive evaluation of the Unanimous Vote function

We now consider the non-adaptive version of the problem. We begin by noting that it is easy to achieve a 2-approximation for this version, as follows. Begin by generating the n possible x_i-first adaptive strategies, one for each x_i. Then convert each of these strategies into a non-adaptive strategy which performs the same first test x_i, and then performs the remaining tests using a round robin (i.e., alternating) between the decreasing p_i ordering of those tests (used by the adaptive strategy when $x_i = 0$) and the increasing p_i ordering (used when $x_i = 1$).

Calculate the expected cost of each of the resulting non-adaptive strategies. Because the round-robin at most doubles the number of tests performed on each assignment in $\{0,1\}^n$, relative to the tests that would have been performed by the adaptive strategy, the resulting non-adaptive strategy having minimum expected cost is a 2-approximation to the optimal adaptive strategy.

To reduce the approximation factor from 2 to $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$, we use a related but more nuanced approach. We call our φ-approximation algorithm the Truncated Round Robin (TRR) algorithm. As in the 2-approximation algorithm, the TRR algorithm begins by generating the n different x_i-first adaptive strategies. For each of these strategies, it produces a corresponding non-adaptive strategy. This strategy performs the first test x_i of the adaptive strategy, and then executes what we call a "truncated round robin" between

the decreasing and increasing p_i orderings of the remaining tests. Finally, it computes the expected cost of the n non-adaptive strategies, and outputs the one with minimum expected cost.

To complete our description of the TRR algorithm, we now describe the truncated round robin. Consider an x_i-first adaptive strategy, after the renumbering of the tests, as shown in the tree in Figure 1 (so x_0 is the first test and $p_1 \geq p_2 \geq \ldots \geq p_{n-1}$). Let "level l" refer to the tree nodes at distance l from the root; namely, x_l and x_{n-l}. Following the first test x_0, the standard round robin (between the increasing and decreasing p_i orderings of the remaining tests) would proceed level by level, testing x_l and x_{n-l} in level l, until the value of the Unanimous Vote function could be determined. Note that, in fact, the standard round robin would terminate at or before level $\lceil \frac{n-1}{2} \rceil$ because at this point all tests have been performed. Thus in each level "reached" by the round robin, $p_l \geq p_{n-l}$.

In the truncated round robin we also proceed level by level, but we do so in two phases. Fix c to be a constant such that $0 < c < \frac{1}{2}$. (To achieve our approximation bound, we will set $c = \frac{3-\sqrt{5}}{2} \approx 0.381966$.) The first phase concludes once we reach a level l where $p_l \geq p_{n-l} \geq 1 - c$ or $c \geq p_l \geq p_{n-l}$. In the first phase, we test both x_l and x_{n-l}, testing first the variable whose probability is closest to $\frac{1}{2}$. In the second phase, we abandon the round robin and instead continue down a single branch in the adaptive tree. Specifically, at the start of the second phase, if $p_l > p_{n-l} \geq 1 - c$, then we continue down the right branch, testing the remaining variables in increasing order of p_i. If $c \geq p_l > p_{n-l}$, then we continue down the left branch, testing the remaining variables in decreasing order of p_i.

Pseudocode describing the order of tests for the resulting non-adaptive strategy is given in Strategy 1. In the pseudocode, "function value undetermined" means tests so far were insufficient to determine the output of the Unanimous Vote function. (The output, POSITIVE, NEGATIVE, or UNCERTAIN, is not shown in the pseudocode.) Pseudocode for the TRR algorithm is given in Algorithm 1.

Theorem 2 *The Truncated Round Robin algorithm is a φ-approximation algorithm solving the non-adaptive, unit-cost SSClass problem, in the special case of the Unanimous Vote function.*

Proof. Consider the optimal adaptive strategy T. After performing its first test, it follows the increasing or decreasing p_i strategy, depending on whether $x_0 = 1$ or $x_0 = 0$. If the other tests are numbered from 1 to $n-1$ so $p_1 \geq p_2 \geq \ldots \geq p_{n-1}$, then T is the tree in Figure 1. Let C^*_{adapt} be the expected cost of optimal adaptive strategy T and let x_{i^*} be its first test. Let $C^*_{non-adapt}$ be the expected cost of the optimal non-adaptive strategy.

Let $C_{i,TRR}$ be the expected cost of the non-adaptive strategy produced by applying truncated round-robin to the x_i-first adaptive strategy (Strategy 1 with first test x_i). Let $C_{TRR} = \min_i C_{i,TRR}$. Since the TRR algorithm (Algorithm 1) tries all possible first tests x_i, C_{TRR} is the expected cost of its output strategy. Further, $C_{TRR} \leq C_{i^*,TRR}$.

Strategy 1 truncated round robin strategy with first test x_i

Input: Test probabilities $p_1, \ldots, p_n$, first test x_i

Renumber the tests so that x_0 is the first test x_i and the remaining tests are numbered from 1 to $n-1$ so that $p_1 \geq p_2 \geq \ldots \geq p_{n-1}$

Test bit x_0
$l \leftarrow 1$ //initialize level number
$c \leftarrow \frac{3-\sqrt{5}}{2} \approx 0.381966$
while $p_{n-l} < 1-c$ **and** $p_l > c$ **and** function value undetermined **do**
  if $|p_l - 0.5| < |p_{n-l} - 0.5|$ **then**
    Test bit x_l followed by x_{n-l}
  else
    Test bit x_{n-l} followed by x_l
  end if
  $l \leftarrow l + 1$
end while//first phase: alternate branches of tree
if $p_l \geq p_{n-l} \geq 1-c$ **then**
  test remaining bits in increasing order of probabilities starting with x_{n-l} until function value is determined
else if $c \geq p_l \geq p_{n-l}$ **then**
  test remaining bits in decreasing order of probabilities starting with x_l until function value is determined
end if//second phase: single branch in tree

Algorithm 1 φ-approximation algorithm for Unanimous Vote

Input: Test probabilities $p_1, \ldots, p_n$
for each test x_i **do**
  calculate the expected cost $C_{i,TRR}$ of the truncated round robin strategy (Strategy 1) with x_i as the first test
end for
$j = \arg\min_i C_{i,TRR}$
return the truncated round robin strategy with first test x_j

We will prove the following claim: $C_{i^*,TRR} \leq \varphi C^*_{adapt}$, and so $C_{TRR} \leq \varphi C^*_{adapt}$. Since the expected cost of the optimal adaptive strategy is bounded above by the expected cost of the optimal non-adaptive strategy, the claim also implies that $C_{TRR} \leq \varphi C^*_{non-adapt}$, which proves the theorem.

We now prove the claim. Consider the non-adaptive strategy produced by applying truncated round robin to the optimal adaptive strategy (i.e., Strategy 1 with first test x_{i^*}). In what follows, we use the renumbering of the tests where x_0 is the first test x_{i^*}, and the remaining tests are numbered from 1 to $n-1$ so $p_1 \geq p_2 \geq \ldots \geq p_{n-1}$. Let ℓ be the level number at the start of the second phase of the truncated round robin.

We will write the expected cost of the non-adaptive strategy as $C_{i^*,TRR} = 1 + E_1 + (1-P_1)E_2$. Here, the 1 is for the first test, E_1 is the expected number of bits tested in the first phase (i.e. in levels $l < \ell$), E_2 is the expected number of bits tested in the second phase (in levels $l \geq \ell$), given that the second phase is reached, and P_1 is the probability of ending during the first phase. Note that

the value of ℓ is determined only by the values of the p_i, and is independent of the test outcomes (as is required in a non-adaptive strategy).

We will write the expected cost of optimal adaptive strategy T as $C^*_{adapt} = 1+E'_1+(1-P'_1)E'_2$ where E'_1 is the expected number of bits tested in T before level ℓ, P'_1 is the probability of ending before level ℓ, and E'_2 is the expected number of bits tested in levels ℓ and higher, given that ℓ was reached.

To prove our claim, we will upper bound the ratio $\alpha := \frac{1+E_1+(1-P_1)E_2}{1+E'_1+(1-P'_1)E'_2}$. Recall that since $c < 1/2$, we have $c < 1-c$. Also, the first phase ends if all bits have been tested, which implies that for all l in the first phase, $l \leq \lceil (n-1)/2 \rceil$ so $p_{n-l} \leq p_l$. We break the first phase into two parts: (1) The first part consists of all levels l where $p_{n-l} \leq c < 1-c \leq p_l$. (2) The second part consists of all levels l where $p_l \in (c, 1-c)$ or $p_{n-l} \in (c, 1-c)$, or both.

Let us rewrite the expected cost E_1 as $E_1 = E_{1,1} + (1-P_{1,1})E_{1,2}$. where $E_{1,1}$ is the expected cost of the first part of phase 1, $E_{1,2}$ is the expected cost of the second part of phase 1, and $P_{1,1}$ is the probability of terminating during the first part of phase 1. Analogously for the cost on tree T, we can rewrite $E'_1 = E'_{1,1} + (1-P'_{1,1})E'_{1,2}$. Then, the ratio we wish to upper bound becomes $\alpha = \frac{1+E_{1,1}+(1-P_{1,1})E_{1,2}+(1-P_1)E_2}{1+E'_{1,1}+(1-P'_{1,1})E'_{1,2}+(1-P'_1)E'_2}$ which we will upper bound by examining the three ratios

$$\theta_1 := \frac{1+E_{1,1}}{1+E'_{1,1}} \qquad \theta_2 := \frac{(1-P_{1,1})E_{1,2}}{(1-P'_{1,1})E'_{1,2}} \qquad \theta_3 := \frac{(1-P_1)E_2}{(1-P'_1)E'_2}$$

Note that if strategy T terminates in the first part of phase 1, then so does the truncated round robin strategy. Thus if $1-P'_{1,1} = 0$ then $1-P_{1,1} = 0$. Also, if $1-P'_1 = 0$, then $1-P_1 = 0$. By Fact 1, we now have $\alpha \leq \max\{\theta_1, \theta_2, \theta_3\}$ and we can upper bound α by upper-bounding the three values θ_1, θ_2 and θ_3.

For ratio θ_1, notice that the truncated round robin does at most two tests for every tree level, so $E_{1,1} \leq 2E'_{1,1}$, and thus $\frac{1+E_{1,1}}{1+E'_{1,1}} \leq \frac{1+2E'_{1,1}}{1+E'_{1,1}}$. Also, the function $\frac{1+2x}{1+x}$ is increasing in x for $x > 0$. For each branch in decision tree T, for the levels in the first part of the first phase, the probability of getting a result that causes termination is at least $1-c$. This is because in the first part, $p_l \geq 1-c > c \geq p_{n-l}$. If we are taking the left branch (because $x_0 = 0$) we terminate when we get a test outcome of 1, and on the right ($x_0 = 1$), we terminate when we get a test outcome of 0. Each bit tested is an independent Bernoulli trial, so $E'_{1,1} \leq \frac{1}{1-c}$. Because $\frac{1+2x}{1+x}$ is increasing in x, we can assert that

$$\theta_1 = \frac{1+E_{1,1}}{1+E'_{1,1}} < \frac{1+2(1-c)^{-1}}{1+1(1-c)^{-1}} = \frac{3-c}{2-c}.$$

Next we will upper bound the second ratio θ_2. Let $P(l)$ represent the probability of reaching level l in the truncated round robin. Further, let q_l represent the probability of testing the second bit in level l given that we have reached level l. Observe that $(1-P_{1,1})E_{1,2}$ can be written as the sum over all levels l in phase 1, part 2 of $P(l)(1+q_l)$. Note that in phase 1, the first bit tested is the x_i such that p_i is closer to 0.5. Note also that in the second part

of the first phase, each level has at least one variable x_i such that $p_i \in (c, 1-c)$ and hence $1 - p_i \in (c, 1 - c)$ also. This means that the first test performed in any given level in phase 1, part 2 will cause the truncated round robin to terminate with probability at least c. So for each level l in this part of the truncated round robin, we have $q_l \leq 1 - c$.

Similarly, $(1 - P'_{1,1})E'_{1,2}$ is the sum over all levels l which comprise phase 1, part 2 in the truncated round robin, of $P'(l)$. Here, $P'(l)$ is defined as the probability of reaching level l in tree T. We do not multiply by $1 + q_l$ since in executing T we only perform one test at each level.

Consider the execution of tree T on an assignment. If the execution terminates upon reaching level l in the tree, for $l < \ell$, then the truncated round robin must terminate at a level $l' \leq l$. That is, the truncated round robin will terminate at level l *or earlier* for the same assignment. Thus, we get that $P(l) \leq P'(l)$. Using this, we can achieve the following bound on the second ratio (letting S_2 denote the set of all levels included in the second part of phase 1):

$$\theta_2 = \frac{(1 - P_{1,1})E_{1,2}}{(1 - P'_{1,1})E'_{1,2}} = \frac{\sum_{l \in S_2} P(l)(1 + q_l)}{\sum_{l \in S_2} P'(l)} \leq \frac{\sum_{l \in S_2} P(l)(1 + 1 - c)}{\sum_{l \in S_2} P(l)} = 2 - c.$$

Finally, we wish to upper bound the last ratio, $\theta_3 = \frac{(1-P_1)E_2}{(1-P'_1)E'_2}$. Recall that ℓ denotes the first level included in the second phase of the truncated round robin. Without loss of generality, assume that $c \geq p_\ell \geq p_{n-\ell}$ so that in the truncated round robin, the second phase tests the remaining bits in decreasing order of p_i. Thus, all bits x_i tested in the second phase satisfy $p_i \leq c$. (The argument is symmetric for the case where $p_\ell \geq p_{n-\ell} \geq 1 - c$).

In this case, any assignments that do not cause termination in the truncated round robin during the first phase, and that have $x_0 = 0$ (i.e., they would go down the left branch of T), will follow the same path through the nodes in the left branch, for levels ℓ and higher, that they would have followed in T. (In fact, tests from the right branch of the tree that were previously performed in phase 1 of the truncated round robin do not have to be repeated.)

The numerator of the third ratio θ_3 is equal to the sum, over all assignments x reaching level ℓ in the truncated round robin, of $p(x)C_2(x)$, where $C_2(x)$ is the total cost of all bits tested in phase 2 for assignment x. Let Q_0 be the subset of assignments reaching level ℓ in the truncated round robin which have $x_0 = 0$ and let Q_1 be the subset of assignments reaching level ℓ in the truncated round robin which have $x_0 = 1$. Let D_0 represent the sum over all assignments in Q_0 of $p(x)C_2(x)$ and let D_1 represent the sum over all assignments in Q_1 of $p(x)C_2(x)$. Then, letting S_ℓ represent the set of assignments reaching level ℓ in the truncated round robin, we can rewrite the numerator of the third ratio as $\sum_{x \in S_\ell} p(x)C_2(x) = \sum_{x \in Q_0} p(x)C_2(x) + \sum_{x \in Q_1} p(x)C_2(x) = D_0 + D_1$.

The denominator of the third ratio is the sum, over all assignments x reaching level ℓ in T, of $p(x)C'_2(x)$, where $C'_2(x)$ is the total cost of all bits tested by T on assignment x at level ℓ and below. Let S'_ℓ denote the set of assignments x reaching level ℓ in tree T. Next, observe that $S_\ell \subseteq S'_\ell$ since any assignment that

reaches level ℓ in the truncated round robin must also reach level ℓ in the tree. If we define $B_0 = \sum_{x \in Q_0} p(x)C'_2(x)$ and $B_1 = \sum_{x \in Q_1} p(x)C'_2(x)$, we can again rewrite the denominator as $\sum_{x \in S'_\ell} p(x)C'_2(x) \geq \sum_{x \in S_\ell} p(x)C'_2(x) = B_0 + B_1$.

The third ratio θ_3 can thus be upper bounded by $\theta_3 \leq \frac{(1-P_1)E_2}{(1-P_1)E'_2} \leq \frac{D_0+D_1}{B_0+B_1}$.

For any $x \in Q_0$, the number of bits tested in level ℓ or below in the truncated round robin is less than or equal to the number of bits tested on x in level ℓ or below in the tree. Thus $D_0 \leq B_0$.

For $x \in Q_1$, the number of bits tested at level ℓ or below in the truncated round robin is at least one. Thus $B_1 \geq J_1$, where J_1 is the probability that a random assignment x has $x_0 = 1$ and reaches level ℓ.

Note that the truncated round robin will terminate on an assignment with $x_0 = 1$ when it first tests a bit that has value 0. Also note that by our assumption, each bit x_i tested in level ℓ and below has probability $p_i \leq c$ of having value 1 and thus probability $1 - p_i \geq 1 - c$ of having value 0 and ending the truncated round robin. Since each bit tested is an independent trial, the expected number of bits tested in level ℓ and below before termination is at most $(1-c)^{-1}$. Thus, $D_1 \leq (1-c)^{-1}J_1$. Together with the fact that $D_0 \leq B_0$, we get $\frac{D_0+D_1}{B_0+B_1} \leq \frac{B_0+(1-c)^{-1}J_1}{B_0+J_1}$. Finally, we observe that since $\frac{B_0}{B_0} = 1$ and $\frac{(1-c)^{-1}J_1}{J_1} \leq \frac{1}{1-c}$, it follows that

$$\theta_3 \leq \frac{D_0 + D_1}{B_0 + B_1} \leq \frac{1}{1-c}.$$

Thus, we have three upper bounds: (1) $\theta_1 \leq \frac{3-c}{2-c}$, (2) $\theta_2 \leq 2 - c$, and (3) $\theta_3 \leq \frac{1}{1-c}$. This gives us an upper bound on the ratio of the expected cost of the truncated round robin to the tree T, and thus an upper bound on the approximation factor. This bound is simply the maximum of the three upper bounds: $\frac{1+E_1+(1-P_1)E_2}{1+E'_1+(1-P'_1)E'_2} \leq \max\left\{\frac{3-c}{2-c}, 2-c, \frac{1}{1-c}\right\}$. Setting $c = \frac{3-\sqrt{5}}{2} \approx 0.381966$ causes all three upper bounds to equal φ. Thus, the truncated round robin algorithm has an expected cost of no more than φ times the expected cost of T. Equivalently, $C_{i^*,TRR} \leq \varphi C^*_{adapt}$, which was the claim. □

7 Non-Adaptive evaluation of *k*-of-*n* functions

The Salloum-Breuer-Ben-Dov algorithm is an exact algorithm for the adaptive evaluation of k-of-n functions. It is still open, however, whether there is a polynomial-time exact algorithm for the non-adaptive version of the problem.

In this section, we present a polynomial-time approximation algorithm achieving a 1.5 approximation factor for the non-adaptive version. Before describing the algorithm, we first point out some fundamental differences between the adaptive and non-adaptive versions of the problem.

The exact algorithm for the adaptive version relies on two key facts: first, that the cost of verification is equal to the cost of evaluation and second, that the optimal ordering of tests depends only on the ranking (relative values) of

the probabilities p_i, and not on their actual values (cf. [3,5]). We begin by showing that neither fact holds in the non-adaptive setting.

Claim In the non-adaptive setting, the optimal cost of evaluating a k-of-n function can exceed the optimal cost of verifying the function.

Proof. Consider the 3-of-6 function defined on the probabilities $[p_1, \ldots, p_6] = [.5, .41, .29, .28, .18, .01]$.

The 1-optimal non-adaptive (or adaptive) strategy for this function is to evaluate the variables in decreasing order of probabilities; the 0-optimal non-adaptive (or adaptive) strategy is to evaluate variables in increasing order of probabilities. These strategies yield a total expected verification cost of 4.52.

However the optimal non-adaptive evaluation strategy evaluates the variables in the order $(x_5, x_4, x_3, x_6, x_2, x_1)$, with an expected cost of 4.81. □

Claim In the non-adaptive setting, an optimal evaluation strategy for k-of-n functions depends not only on the ranking (relative values) of the probabilities p_i, but also on their actual values.

Proof. To demonstrate this claim, we consider evaluating the 3-of-6 function with respect to two different vectors of probabilities: $p = [p_1, \ldots, p_n]$ where $p_1 > p_2 > \ldots > p_n$; and $p' = [p'_1, \ldots, p'_n]$ where $p'_1 > p'_2 > \ldots > p'_n$. We show that the optimal permutation for p is not optimal for p' (with the probabilities in p replaced by probabilities in p').

Let $[p_1, \ldots, p_6] = [.5, .41, .29, .28, .18, .01]$. As stated above, the optimal permutation for this function is $(x_5, x_4, x_3, x_6, x_2, x_1)$ with an expected cost of 4.81. The non-adaptive strategy defined by decreasing order of probabilities, $(x_1, \ldots, x_6)$, is suboptimal, with an expected cost of 4.99.

In contrast, let $p' = [.6, .5, .4, .3, .2, .1]$. The optimal permutation for p' orders the variables in decreasing order of proabaility, $(x_1, \ldots, x_6)$, and has an expected cost of 4.93. The strategy $(x_5, x_4, x_3, x_6, x_2, x_1)$, optimal for p, is suboptimal for p', with an expected cost of 4.96. □

Because of the above differences, the approach used in the Salloum-Breuer-Ben-Dov algorithm is not suitable for the non-adaptive problem. Our approximation algorithm for the non-adaptive problem uses a different approach. It relies on a dynamic programming algorithm that takes as input a non-adaptive strategy for evaluating a k-of-n function and returns its expected cost. The dynamic programming algorithm works by constructing an $(n+1) \times (k+1)$ table P and an $(n+1) \times (n-k+2)$ table Q. For $j \leq i$, table entry $P[i,j]$ stores the probability that the outcomes of the tests on $x_1, \ldots, x_i$ will consist of exactly j ones and $i-j$ zeros. Table entry $Q[i,j]$ stores the probability that the outcomes of the tests on $x_1, \ldots, x_i$ will consist of exactly j zeros and $i-j$ ones.

The base cases of the recurrence for $P[i,j]$ are $P[0,0] = 1$, $P[0,j] = 0$ for $j \in \{1, \ldots, k\}$, and $P[i,0] = \prod_{a=1}^{i}(1-p_a)$ for $i \in \{1, \ldots, n\}$. For the other i,j entries, if $j \leq i$, $P[i,j] = P[i-1, j-1](p_i) + P[i-1,j](1-p_i)$. A symmetric recurrence holds for Q.

When evaluating a k-of-n function, testing stops immediately after the kth positive test result, or immediately after the $(n-k+1)$th negative test result. Let $z = n - k + 1$. Thus the probability that testing stops immediately after the ith test is $(P[i-1, k-1] \times p_i + Q[i-1, z-1] \times (1-p_i))$. Thus the dynamic programming algorithm outputs the quantity $\sum_{i=1}^{n}(P[i-1, k-1] \times p_i + Q[i-1, z-1] \times (1-p_i)) \times i$ as the expected testing cost. We present the pseudocode for the dynamic programming algorithm in Algorithm 2.

Algorithm 2 calculating expected cost of a non-adaptive k-of-n strategy

Input: permutation of variables $x_1, \ldots, x_n$ with probabilities p_i, int k
Let $z = n - k + 1$
Table $P[0 \ldots n][0 \ldots k]$ //$P[i, j]$ denotes the probability of having exactly j ones in $x_1, \ldots, x_i$
$P[0, 0] \leftarrow 1$
$P[0, j] \leftarrow 0$ for $j \in \{1, \ldots, k\}$
$P[i, 0] \leftarrow \prod_{a=1}^{i}(1 - p_a)$ for $i \in \{1, \ldots, n\}$
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** k **do**
    if $j > i$ **then**
      $P[i, j] = 0$
    else
      $P[i, j] = P[i-1, j-1](p_i) + P[i-1, j](1 - p_i)$
    end if
  end for
end for
Table $Q[0 \ldots n][0 \ldots z]$ //$Q[i, j]$ denotes the probability of having exactly j zeros in $x_1, \ldots x_i$
$Q[0, 0] \leftarrow 1$
$Q[0, j] \leftarrow 0$ for $j \in \{1, \ldots, z\}$
$Q[i, 0] \leftarrow \prod_{a=1}^{i} p_a$ for $i \in \{1, \ldots, n\}$
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** z **do**
    if $j > i$ **then**
      $Q[i, j] = 0$
    else
      $Q[i, j] = Q[i-1, j-1](1 - p_i) + Q[i-1, j](p_i)$
    end if
  end for
end for
$c = \sum_{i=1}^{n}(P[i-1, k-1] \times p_i + Q[i-1, z-1] \times (1 - p_i)) \times i$
return c

We now present our 1.5-approximation algorithm for evaluating k-of-n functions in the non-adaptive setting. Our algorithm simply uses the dynamic programming algorithm to compute the expected cost of two different non-adaptive strategies: the strategy that tests the bits in increasing p_i order, and the strategy that tests the bits in decreasing p_i order. It then outputs the strategy with smaller expected cost. Pseudocode is presented in Algorithm 3.

We will show that Algorithm 3 achieves a 1.5-approximation for the non-adaptive version of the k-of-n evaluation problem.

Algorithm 3 Non-adaptive evaluation of k-of-n functions

Input: test probabilities $p_1, \ldots, p_n$
Let P_1 denote the permutation of $x_1, \ldots, x_n$, sorted in decreasing order of p_i values
Let P_0 denote the permutation of $x_1, \ldots, x_n$, sorted in increasing order of p_i values
c_1 = expected cost of P_1 (computed using Algorithm 2)
c_0 = expected cost of P_0 (computed using Algorithm 2)
if $c_1 < c_0$ then $minperm = P_1$ else $minperm = P_0$
return minperm

For a given k-of-n function f, and test probabilities $p_1, \ldots, p_n$, let ϱ denote the probability that $f = 1$. Let $\lambda = k/n$. We assume, without loss of generality, that $\lambda \geq \frac{1}{2}$.

We begin by bounding the expected costs of the 0- and 1-optimal non-adaptive strategies (which test the bits in increasing and decreasing p_i orders respectively) in terms of the expected cost of an optimal adaptive strategy. Let OPT denote an optimal adaptive strategy and let 0_{OPT} and 1_{OPT} denote the 0- and 1-optimal non-adaptive strategies, respectively. We denote the expected cost of a strategy S by $E[S]$.

Lemma 1

$$\frac{E[0_{\mathsf{OPT}}]}{E[\mathsf{OPT}]} \leq R_0 := \frac{\varrho + (1-\varrho)(1-\lambda)}{\varrho\lambda + (1-\varrho)(1-\lambda)}.$$

Proof. Clearly, $E[0_{\mathsf{OPT}}|f = 1] \leq n$. So

$$E[0_{\mathsf{OPT}}] = \varrho E[0_{\mathsf{OPT}}|f = 1] + (1-\varrho)E[0_{\mathsf{OPT}}|f = 0] \leq \varrho n + (1-\varrho)E[0_{\mathsf{OPT}}|f = 0].$$

We also have that

$$E[\mathsf{OPT}] = \varrho E[\mathsf{OPT}|f = 1] + (1-\varrho)E[\mathsf{OPT}|f = 0] \geq \varrho\lambda n + (1-\varrho)E[0_{\mathsf{OPT}}|f = 0]$$

because any strategy must perform at least λn tests when $f = 1$, and $E[\mathsf{OPT}|f = 0] \geq E[0_{\mathsf{OPT}}|f = 0]$.

Therefore:

$$\frac{E[0_{\mathsf{OPT}}]}{E[\mathsf{OPT}]} \leq \frac{\varrho n + (1-\varrho)E[0_{\mathsf{OPT}}|f = 0]}{\varrho\lambda n + (1-\varrho)E[0_{\mathsf{OPT}}|f = 0]}.$$

Because this expression increases as the value of $E[0_{\mathsf{OPT}}|f = 0]$ decreases, and because every strategy must perform at least $(1-\lambda)n$ tests if $f = 0$, we can upper bound this ratio:

$$\frac{E[0_{\mathsf{OPT}}]}{E[\mathsf{OPT}]} \leq \frac{\varrho n + (1-\varrho)(1-\lambda)n}{\varrho\lambda n + (1-\varrho)(1-\lambda)n}.$$

The claim follows easily. □

A symmetric argument proves the analogous lemma:

Lemma 2

$$\frac{E[1_{\mathsf{OPT}}]}{E[\mathsf{OPT}]} \leq R_1 := \frac{\varrho\lambda + (1-\varrho)}{\varrho\lambda + (1-\varrho)(1-\lambda)}.$$

Theorem 3 *Algorithm 3 is a 1.5-approximation algorithm solving the non-adaptive, unit-cost SSClass problem, in the special case of k-of-n functions.*

Proof. Clearly, the strategy output by Algorithm 3 has an expected cost that is at most a factor of $\min\{R_0, R_1\}$ larger than that of the optimal strategy. We now argue that $\min\{R_0, R_1\} \leq 1.5$.

By definition, each p_i is strictly between 0 and 1, so there is a non-zero probability both that $f = 0$ and that $f = 1$. Therefore $0 < \varrho < 1$. Recall also that $1/2 \leq \lambda \leq 1$.

When $\lambda = 1$, $R_0 = 1$ and $\min\{R_0, R_1\} \leq 1.5$ clearly holds. Assume therefore that $1/2 \leq \lambda < 1$.

Suppose $R_0 \leq R_1$. Then, because R_0 and R_1 have the same denominator,

$$\varrho + (1 - \varrho)(1 - \lambda) \leq \varrho\lambda + (1 - \varrho)$$

which implies $\varrho \leq \lambda$. Combining this with the fact that $\frac{\partial R_0}{\partial \varrho} \geq 0$ for the possible values of ϱ and λ, we can substitute λ for ϱ in the expression for R_0 to get

$$R_0 \leq \frac{\lambda + (1 - \lambda)^2}{\lambda^2 + (1 - \lambda)^2}.$$

As a function of λ, this expression is maximized when $\lambda = \frac{1}{2}$, and therefore $\min\{R_0, R_1\} = R_0 \leq 1.5$.

Now suppose $R_0 > R_1$. Combining this with the fact $\frac{\partial R_1}{\partial \varrho} \leq 0$ for the possible values of ϱ and λ, we can substitute λ for ϱ in the expression for R_1 to get

$$R_1 \leq \frac{\lambda^2 + (1 - \lambda)}{\lambda^2 + (1 - \lambda)^2} = \frac{\lambda + (1 - \lambda)^2}{\lambda^2 + (1 - \lambda)^2}.$$

This is the same expression that upper bounded R_0 in the previous case, and we again have $\min\{R_0, R_1\} \leq 1.5$. □

The bound in Theorem 3 is tight relative to the optimal adaptive strategy: Consider a Majority function on n variables, where n is odd (i.e., a k-of-n function where n is odd and $k = \frac{n-1}{2} + 1$). Suppose $\frac{n-1}{2}$ variables have probabilities p_i very close to 1, $\frac{n-1}{2}$ variables have probabilities p_i very close to 0, and one variable has a probability of $p_i = \frac{1}{2}$. Then the optimal adaptive strategy will incur an expected cost of approximately $\frac{n-1}{2} + 1$ and Algorithm 3 will incur an expected cost of approximately $\frac{3n}{4} + \frac{1}{4}$. (In contrast, the Up-Down strategy given in Section 5 will achieve an expected cost of approximately n here.) However, it may not be tight relative to the optimal non-adaptive strategy.

Acknowledgements We thank an anonymous referee for suggesting we present our results in terms of SSClass.

References

1. Acharya, J., Jafarpour, A., Orlitsky, A.: Expected query complexity of symmetric Boolean functions. In: IEEE 49th Annual Allerton Conference on Communication, Control, and Computing, pp. 26–29 (2011)
2. Ben-Dov, Y.: Optimal testing procedure for special structures of coherent systems. Management Science (1981)
3. Boros, E., Ünlüyurt, T.: Diagnosing double regular systems. Annals of Mathematics and Artificial Intelligence **26**(1-4), 171–191 (1999). DOI 10.1023/A:1018958928835. URL `http://dx.doi.org/10.1023/A:1018958928835`
4. Chang, M.F., Shi, W., Fuchs, W.K.: Optimal diagnosis procedures for k-out-of-n structures. IEEE Transactions on Computers **39**(4), 559–564 (1990)
5. Das, H., Jafarpour, A., Orlitsky, A., Pan, S., Suresh, A.T.: On the query computation and verification of functions. In: IEEE International Symposium on Information Theory (ISIT), pp. 2711–2715 (2012)
6. Deshpande, A., Hellerstein, L., Kletenik, D.: Approximation algorithms for stochastic submodular set cover with applications to boolean function evaluation and min-knapsack. ACM Trans. Algorithms **12**(3), 42:1–42:28 (2016). DOI 10.1145/2876506. URL `http://doi.acm.org/10.1145/2876506`
7. Ghuge, R., Gupta, A., Nagarajan, V.: Non-adaptive stochastic score classification and explainable halfspace evaluation. CoRR **abs/2111.05687** (2021). URL `https://arxiv.org/abs/2111.05687`
8. Gkenosis, D., Grammel, N., Hellerstein, L., Kletenik, D.: The Stochastic Score Classification Problem. In: Y. Azar, H. Bast, G. Herman (eds.) 26th Annual European Symposium on Algorithms (ESA 2018), *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 112, pp. 36:1–36:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). DOI 10.4230/LIPIcs.ESA.2018.36. URL `http://drops.dagstuhl.de/opus/volltexte/2018/9499`
9. Gkenosis, D., Grammel, N., Hellerstein, L., Kletenik, D.: The stochastic boolean function evaluation problem for symmetric boolean functions. Discrete Applied Mathematics **309**, 269–277 (2022). DOI https://doi.org/10.1016/j.dam.2021.12.001. URL `https://www.sciencedirect.com/science/article/pii/S0166218X21004790`
10. Greiner, R., Hayward, R., Jankowska, M., Molloy, M.: Finding optimal satisficing strategies for and-or trees. Artificial Intelligence **170**(1), 19–58 (2006)
11. Gupta, A., Nagarajan, V.: A stochastic probing problem with applications. In: International Conference on Integer Programming and Combinatorial Optimization, pp. 205–216. Springer (2013)
12. Jung, J., Concannon, C., Shroff, R., Goel, S., Goldstein, D.G.: Simple rules for complex decisions. arXiv preprint arXiv:1702.04690 (2017)
13. Kowshik, H., Kumar, P.: Optimal computation of symmetric boolean functions in collocated networks. IEEE Journal on Selected Areas in Communications **31**(4), 639–654 (2013)
14. Salloum, S.: Optimal testing algorithms for symmetric coherent systems. Ph.D. thesis, University of Southern California (1979)
15. Salloum, S., Breuer, M.: An optimum testing algorithm for some symmetric coherent systems. Journal of Mathematical Analysis and Applications **101**(1), 170 – 194 (1984). DOI 10.1016/0022-247X(84)90064-7. URL `http://www.sciencedirect.com/science/article/pii/0022247X84900647`
16. Salloum, S., Breuer, M.A.: Fast optimal diagnosis procedures for k-out-of-n:g systems. IEEE Transactions on Reliability **46**(2), 283–290 (1997). DOI 10.1109/24.589958
17. Singla, S.: The price of information in combinatorial optimization. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2523–2532. SIAM (2018)
18. Tran, T., Luo, W., Phung, D., Morris, J., Rickard, K., Venkatesh, S.: Preterm birth prediction: Deriving stable and interpretable rules from high dimensional data. In: Conference on Machine Learning in Healthcare, LA, USA (2016)
19. Ünlüyurt, T.: Sequential testing of complex systems: a review. Discrete Applied Mathematics **142**(1-3), 189–205 (2004)

20. Ustun, B., Rudin, C.: Supersparse linear integer models for optimized medical scoring systems. Machine Learning **102**(3), 349–391 (2016)
21. Ustun, B., Rudin, C.: Optimized risk scores. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1125–1134. ACM (2017)
22. Zeng, J., Ustun, B., Rudin, C.: Interpretable classification models for recidivism prediction. Journal of the Royal Statistical Society: Series A (Statistics in Society) **180**(3), 689–722 (2017)