The Stochastic Score Classification Problem

Dimitrios Gkenosis
Department of Informatics and Telecommunications, University of Athens, Athens, Greece
gkenosis.dimitrios@math.uoa.gr

Nathaniel Grammel
Department of Computer Science, University of Maryland, College Park, Maryland, USA
ngrammel@cs.umd.edu

Lisa Hellerstein
Department of Computer Science and Engineering, NYU Tandon School of Engineering, Brooklyn, NY, USA
lisa.hellerstein@nyu.edu

Devorah Kletenik[1]
Department of Computer and Information Science, Brooklyn College, CUNY, Brooklyn, New York, USA
kletenik@sci.brooklyn.cuny.edu

Abstract

Consider the following Stochastic Score Classification Problem. A doctor is assessing a patient's risk of developing a certain disease, and can perform n tests on the patient. Each test has a binary outcome, positive or negative. A positive result is an indication of risk, and a patient's score is the total number of positive test results. Test results are accurate. 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. Test i has probability p_i of being positive, and it costs c_i to perform. To reduce costs, instead of performing all tests, the doctor will perform them sequentially and stop testing when it is possible to determine the patient's risk category. The problem is to determine the order in which the doctor should perform the tests, so as to minimize expected testing cost. We provide approximation algorithms for adaptive and non-adaptive versions of this problem, and pose a number of open questions.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

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

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.36

Related Version Full version on arXiv [9], `https://arxiv.org/abs/1806.10660`.

Funding The authors were partially supported by NSF Award IIS-1217968.

Acknowledgements We thank Zach Pomerantz for experiments that gave us useful insights into goal functions. We thank the ESA referees for helpful comments.

[1] Partially supported by a PSC-CUNY Award, jointly funded by The Professional Staff Congress and The City University of New York

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 tests on the patient. Each test has a binary outcome, positive or negative. A positive result is an indication of risk, and a patient's score is the total number of positive test results. Test results are accurate. 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. Test i has probability p_i of being positive, and it costs c_i to perform. To reduce costs, instead of performing all tests, the doctor will perform them sequentially and stop testing when it is possible to determine the risk category for the patient.

To reduce costs, instead of performing all tests 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 expected cost.

Formally, the Stochastic Score Classification 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* $[\alpha_j, \alpha_j + 1, \ldots, \alpha_{j+1} - 1]$. The α_j 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* $\sum_i X_i$. Thus B is the number of classes. Each variable X_i is independently 1 with given probability p_i, where $0 < p_i < 1$, and is 0 otherwise. The value of X_i can only be determined by asking a query (or performing a test), which incurs a given positive, real-valued cost c_i.

An *evaluation strategy* for f is a sequential adaptive or non-adaptive order in which to ask the queries. Querying must continue until the value of f can be determined, i.e., until the value of f would be the same, no matter how the remainder of the n queries were answered. In an *adaptive evaluation strategy*, the choice of the next query can depend on the outcomes of previous queries. An adaptive strategy corresponds to a decision tree, although we do not require the tree to be output explicitly (it may have exponential size). A *non-adaptive strategy* is a permutation of the queries. With a non-adaptive strategy, querying proceeds in the order specified by the permutation until the value of f can be determined.

Repeated queries always receive the same response, so it is never useful to ask a particular query more than once. The goal is to design an evaluation strategy for f with minimum expected total query cost. We consider both adaptive and non-adaptive versions of the problem, in which we are restricted to adaptive or non-adaptive strategies respectively.

We also consider a *weighted* variant of the problem, where query i has given integer weight a_i, the score is $\sum_i a_i X_i$, and $\alpha_1 < \alpha_2 < \ldots < \alpha_B < \alpha_{B+1}$ where α_1 equals the minimum possible value of the score $\sum_i a_i X_i$, and $\alpha_{B+1} - 1$ equals the maximum possible value. We refer to the standard version of the problem, with score $\sum_i X_i$, as the *unweighted* version.

While we have described the problem above 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 [25, 23, 15, 27, 26]. 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. [22, 14]) because of the requirement that testing must continue until the unique interval containing the score has been determined.

Restricted versions of the weighted and unweighted SSClass problem have been studied previously. In the algorithms literature, Deshpande et al. presented two approximation algorithms solving the *Stochastic Boolean Function Evaluation* (SBFE) problem for linear threshold functions [8]. 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 the weighted adaptive SSClass problem. One of the two algorithms of Deshpande et al. achieves an $O(\log W)$-approximation factor for this problem using the submodular goal value approach; it involves construction of a goal utility function and application of the Adaptive Greedy algorithm of Golovin and Krause to that function [10]. Here W is the sum of the magnitudes of the integer weights a_i. The other algorithm achieves a 3-approximation by applying a dual greedy algorithm to the same goal utility function.

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 evaluating k-of-n functions is equivalent to the unweighted adaptive SSClass problem, with only two classes ($B = 2$). It has been studied previously in the VLSI testing literature. There is an elegant algorithm for the problem that computes an optimal strategy [19, 4, 20, 6].

The unweighted adaptive SSClass problem for arbitrary numbers of classes was studied in the information theory literature [7, 1, 17], but only for unit costs. The main novel contribution there was to establish an equivalence between verification and evaluation, which we discuss below.

2 Results and open questions

We give approximation results for adaptive and non-adaptive versions of the SSClass problem. We describe most of our results here, but leave description of some others to the full version of our paper [9]. Omitted proofs also appear there. A table with all our bounds can be found at the end of this paper.

We begin by using the submodular goal value approach of Deshpande et al. to obtain an $O(\log W)$ approximation algorithm for the weighted adaptive SSClass problem. This immediately gives an $O(\log n)$ approximation for the unweighted adaptive problem. We also present a simple alternative algorithm achieving a $B - 1$ approximation for the unweighted adaptive problem, and a $3(B-1)$-approximation algorithm for the weighted adaptive problem, again using an algorithm of Deshpande et al.

We then present our two main results, which are both for the case of unit costs. The first is a 4-approximation algorithm for the adaptive and non-adaptive versions of the unweighted SSClass problem. The second is a φ-approximation for a special case of the non-adaptive unweighted version, where the problem is to evaluate what we call the *Unanimous Vote* function. Here $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ is the golden ratio. 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 $\{n\}$. The proofs of our two main results imply upper bounds of 4 and φ for the adaptivity gaps of the corresponding problems.

We use both existing techniques and new ideas in our algorithms. We use the submodular goal value approach of Deshpande et al. to get our $O(\log W)$ bound for the weighted adaptive SSClass problem. This approach cannot yield a bound better than $O(\log n)$ for SSClass problems, since they involve evaluating a function of n relevant Boolean variables [3].

For some of our other bounds, we exploit the exact algorithm for k-of-n evaluation, and the ideas used in its analysis. To obtain non-adaptive algorithms for the unit-cost case, we perform a round robin between 2 subroutines, one performing queries in increasing order of c_i/p_i, while the second performs them in increasing order of $c_i/(1-p_i)$. For arbitrary costs, instead of standard round robin, we use the modified round robin approach of Allen et al [2]. As has been repeatedly shown, the c_i/p_i ordering and the $c_i/(1-p_i)$ ordering are optimal for evaluation of the Boolean OR (1-of-n) and AND (n-of-n) functions respectively (cf. [24]). Intuitively, the first ordering (for OR) favors queries with low cost and high probability of producing the value 1, while the second (for AND) favors queries with low cost and high probability of producing the value 0. The proof of optimality follows from the fact that given any ordering, swapping two adjacent queries that do not follow the designated increasing order will decrease expected evaluation cost.

While the algorithm for our first main result is very simple, the proof of its 4-approximation bound is not. It uses ideas from the existing analysis of the k-of-n algorithm, but that analysis is simpler because $B = 2$. We perform a new, careful analysis to obtain our 4-approximation result. Unlike the analysis of the k-of-n algorithm, our analysis only works for unit costs.

To develop our φ-approximation for the Unanimous Vote function, we first note that for such a function, if you perform the first query and observe its outcome, the optimal ordering of the remaining queries can be determined by evaluating a Boolean OR function, or a Boolean AND function. We then address the problem of determining an approximately optimal permutation, given the first query. A standard round robin alternating between the $c_i/p_i = 1/p_i$ ordering, and the $1/(1-p_i)$ ordering, yields a factor of 2 approximation. To obtain the φ factor, we stop the round robin at a carefully chosen point and *commit* to one of the two orderings, abandoning the other. Our full algorithm for the Unanimous Vote function works by trying all n possible first queries. For each, we generate the approximately optimal permutation given that first choice, and algebraically compute its expected cost. Finally, out of these n permutations, we choose the one with lowest expected cost.

We note that although our algorithms are designed to minimize expected cost for independent queries, the goal value function used to achieve the $O(\log W)$ approximation result can also be used to achieve a worst-case bound, and a related bound in the Scenario model [10, 12, 16].

A recurring theme in work on SSClass problems has been the relationship between the evaluation problems and their associated verification problems. In the verification problem, you are given the output class (i.e., the value of the score classification function) before querying, and just need to perform enough tests to verify that the given output class is correct. Thus optimal expected verification cost lower bounds optimal expected evaluation cost. Surprisingly, the result of Das et al. [7] showed that for the adaptive SSClass problem in the unit-cost case, optimal expected verification cost equals optimal expected evaluation cost. Prior work already implied this was true for evaluating k-of-n functions, even for arbitrary costs (cf. [5]). We give a counterexample in the full paper [9] showing that this relationship does not hold for the adaptive SSClass problem with arbitrary costs. Thus algorithmic approaches based on optimal verification strategies may not be effective for this problem.

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.

This is open even in the unit-cost case. It is unclear whether this problem will be easy to resolve. It is easy to show that the weighted variants are NP-hard: this follows from the NP-hardness of the SBFE problem for linear threshold functions, which is proved by a simple reduction from knapsack [8]. However, the approach used in that proof is to show that the deterministic version of the problem (where query answers are known a-priori) is NP-hard, which is not the case in the SSClass problem. Further, NP-hardness of evaluation problems is not always easy to determine. The question of whether the SBFE problem for read-once formulas is NP-hard has been open since the 1970's (cf. [13]).

Another main open question is whether there is a constant-factor approximation algorithm for the weighted SSClass problem. Our bounds depend on n or B. Other open questions concern lower bounds on approximation factors, and bounds on adaptivity gaps.

3 Further definitions and background

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 bits i with $b_i = *$, produced by fixing the remaining bits i according to their values b_i. We call f^b the function *induced from f by partial assignment b*. We use $N_0(b)$ to denote $|\{i|b_i = 0\}|$, and $N_1(b)$ to denote $|\{i|b_i = 1\}|$.

A partial assignment $b' \in \{0,1,*\}^n$ is an *extension* of b, written $b' \succeq b$, if $b'_i = b_i$ for all i such that $b_i \neq *$. We use $b' \succ b$ to denote that $b' \succeq b$ and $b' \neq b$.

A partial assignment encodes what information is known at a given point in a sequential querying (testing) environment. Specifically, for partial assignment $b \in \{0,1,*\}^n$, $b_i = *$ indicates that query i has not yet been asked, otherwise b_i equals the answer to query i. We may also refer to query i as *test* i, and to asking query i as testing or querying bit x_i,

Suppose the costs c_i and probabilities p_i for the n queries are fixed. We define the expected costs of adaptive evaluation and verification strategies 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 $p(x) = \prod_{i=1}^n p^{x_i}(1-p)^{1-x_i}$. 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 $\mathcal{X}_\ell = \{x \in \{0,1\}^n : f(x) = \ell\}$. 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 $g\colon \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$ is *monotone* if $g(b') \geq g(b)$ whenever $b' \succeq b$. It is *submodular* if for $b' \succeq b$, i such that $b'_i = b_i = *$, and $k \in \{0,1\}$, we have $g(b'_{i\leftarrow k}) - g(b') \leq g(b_{i\leftarrow k}) - g(b)$. Here $b_{i\leftarrow k}$ denotes the partial assignment produced from b by setting b_i to k, and similarly for $b'_{i\leftarrow k}$.

4 Algorithms for the weighted adaptive SSClass problem

Our first algorithm solves the weighted adaptive SSClass Problem using the *goal value approach* of Deshpande et al., a method of designing approximation algorithms for SBFE problems [8]. The approach can easily be extended to problems of evaluating pseudo-Boolean functions. It requires construction of a utility function $g\colon \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$, called a *goal function*, associated with the function f being evaluated. Function g must be monotone and submodular. The maximum value of g must be an integer $Q \geq 0$ such that $g(b) = Q$ iff $f(x)$ has the same value for all $x \in \{0,1\}^n$ such that $x \succeq b$. We call Q the *goal value* of g.

An adaptive strategy for evaluating f can then be obtained by applying the Adaptive Greedy algorithm of Golovin and Krause to solve the Stochastic Submodular Cover problem on goal function g [10]. This algorithm greedily chooses the test with highest expected increase in utility, as measured by g, per unit cost. It follows from the bound of Deshpande et al. on applying Adaptive Greedy to the Stochastic Submodular Cover problem, that this strategy is an $O(\log Q)$-approximation to the optimal adaptive strategy for evaluating f [8].[2]

We construct g as follows. Let $r(x) = a_1x_2 + \ldots + a_nx_n$. Consider an associated score classification function f defined by $\alpha_1, \ldots, \alpha_{B+1}$ and the a_i. For simplicity, we assume here that the a_i are non-negative. (The general case is similar.) We refer to the values $\alpha_2, \ldots, \alpha_B$ as *cutoffs*. For each cutoff α_j, let f_j denote the Boolean linear threshold function $f_j : \{0,1\}^n \to \{0,1\}$ where $f_j(x) = 1$ if $r(x) \geq \alpha_j$, and $f_j(x) = 0$ otherwise.

Consider a fixed cutoff α_j. Let $\omega = (\sum_i a_i) - \alpha_j + 1$. For $b \in \{0,1,*\}^n$, let $r^1(b) = \min\{\alpha_j, \sum_{i:b_i=1} a_i\}$ and $r^0(b) = \min\{\omega, \sum_{i:b_i=0} a_i\}$. Note that $r^1(b) = \alpha_j$ iff $f_j(x) = 1$ for all $x \succeq b$, and $r^0(b) = \omega$ iff $f_j(x) = 0$ for all $x \succeq b$. As shown in [8] the following function g_j is a goal function for linear threshold function f_j, with goal value $\omega\alpha_j$:

$$g_j(b) = \omega\alpha_j - (\alpha_j - r^1(b))(\omega - r^0(b)). \tag{1}$$

We combine the $B-1$ goal functions g_j using the standard "AND construction" for utility functions (cf. [8]), which yields a goal function g for score classification function f, where $g(x) = \sum_{i=1}^{B-1} g_i(x)$. Its goal value is at most $(B-1)W^2$ where $W = \sum_i a_i$.

To evaluate f, we apply the Adaptive Greedy algorithm to g. By the $O(\log Q)$ approximation bound on Adaptive Greedy, this constitutes an algorithm for the weighted adaptive SSClass problem with approximation factor $O(\log BW^2)$, which is $O(\log W)$ since $B \leq W$. In the unweighted adaptive SSClass problem, $W = n$, so the approximation factor is $O(\log n)$.

We now describe our simple $B-1$ approximation algorithm for the adaptive unweighted SSClass problem, which takes a very different approach. It runs the k-of-n function evaluation algorithm $B-1$ times, each time setting k to be a different cutoff α_j. The resulting evaluations are sufficient to determine the correct output class. The proof that this algorithm achieves a $B-1$ approximation bound is based on the observation that any strategy solving the adaptive SSClass problem is implicitly a strategy for solving each of the $B-1$ induced k-of-n problems. Since we use an optimal algorithm for solving each of those problems, this implies the $B-1$ approximation bound. We note that although we could easily modify this algorithm to use binary search, we do not know how to prove that it results in an approximation bound that is better than $B-1$.

When B is small, as for, e.g., k-of-n functions and the Unanimous Vote function, $B-1$ is a good approximation. Otherwise, the $O(\log n)$ approximation achieved with the goal value approach may be better.

By similar arguments, the following is a $3(B-1)$ approximation for the weighted adaptive problem. For each cutoff α_j, use the 3-approximation algorithm of Deshpande et al. to evaluate linear threshold function f_j.

[2] Golovin and Krause originally claimed an $O(\log Q)$ bound for Stochastic Submodular Cover [10], but the proof was recently found to have an error [18]. They have since posted a new proof with an $O(\log^2 Q)$ bound [11]. Deshpande et al. proved an $O(\log Q)$ bound using a different proof technique [8].

Combining the above results, we have the following theorem.

▶ **Theorem 1.** *There are two different polynomial-time approximation algorithms, achieving approximation factors of $O(\log W)$ and $3(B-1)$ respectively, for the weighted adaptive SSClass problem. There is a polynomial-time algorithm that achieves a $B-1$-approximation for the unweighted adaptive SSClass problem.*

5 Constant-factor approximations for unit-cost problems

We begin by reviewing relevant existing techniques.

5.1 Adaptive Evaluation of k-of-n Functions

An optimal adaptive strategy, when f is a k-of-n function, was given by Salloum, Ben-Dov, and Breuer [19, 4, 20, 6, 21]. The difficulty in finding an optimal strategy is that you do not know a-priori whether the value of f will be 1 or 0. If 1, then (ignoring cost) it seems it would be better to choose tests with high p_i, since you want to get k 1-answers. Similarly, if 0, it seems it would be better to choose tests with low p_i. The algorithm of Salloum et al. is based on showing that when f is a k-of-n function, a 1-optimal strategy is to test the bits in increasing order of c_i/p_i until getting k 1's, while a 0-optimal strategy is to test them in increasing order of $c_i/(1-p_i)$ until getting $n-k+1$ 0's.

Since the 1-optimal strategy must perform at least the first k tests before terminating, these can be reordered within this strategy without affecting its optimality. Similarly, the first $n-k+1$ queries of the 0-optimal strategy can be reordered without affecting optimality.

The strategy of Salloum et al. is as follows. If $n = 1$, test the one bit. Else let S_1 denote the set of the k bits with smallest c_i/p_i values. Let S_0 denote the set of the $n-k+1$ bits with smallest $c_i/(1-p_i)$ values. Since $|S_0| + |S_1| = n+1$, by pigeonhole $S_0 \cap S_1 \neq \emptyset$. Test a bit in $S_0 \cap S_1$. If it is 1, the problem is reduced to evaluating the function $f^1 \colon \{0,1\}^{n-1} \to \{0,1\}$ where $f^1(x) = 1$ iff $N_1(x) \geq k-1$. If it is 0, the problem is reduced to evaluating $f^0 \colon \{0,1\}^{n-1} \to \{0,1\}$ where $f^0(x) = 1$ iff $N_1(x) \geq k$. Recursively evaluate f^1 or f^0 as appropriate. Optimality follows from the fact that the chosen bit is an optimal first bit to test in both 0-optimal and 1-optimal strategies.

5.2 Modified Round Robin

Allen et al. [2] presented a modified round robin protocol, which is useful in designing non-adaptive strategies when test costs are not all equal. Suppose that in a sequential testing environment with n tests, we have M conditions on test outcomes, corresponding to M predicates on the partial assignments in $\{0,1,*\}^n$. For example, in the k-of-n testing problem, we are interested in the following $M = 2$ predicates on partial assignments: (1) having at least k 1's and (2) having at least $n-k+1$ 0's. Suppose we are given a testing strategy for each of the M predicates; a strategy stops testing when its predicate is satisfied (by the partial assignment representing test outcomes), or all tests have been performed. Let $\mathrm{Alg}_1, \ldots, \mathrm{Alg}_M$ denote those M strategies. The modified round robin algorithm of Allen et al. interleaves execution of these strategies. We present a version of their algorithm in Algorithm 1; the difference is that their algorithm terminates as soon as one of the predicates is satisfied, while Algorithm 1 terminates when all are satisfied.

Allen et al. showed that the modified round robin incurs a cost on x that is at most M times the cost incurred by Alg_j on x. We will use variations on this algorithm and this bound to derive approximation factors for our SSClass problems.

Algorithm 1 Modified Round Robin of M Strategies.

```
Let $C_i \leftarrow 0$ for $i = 1, \ldots, M$; let $d \leftarrow (*^n)$
while at least one of the $M$ testing strategies has not terminated do
  Let $j_1, \ldots, j_M$ be the next tests of $\mathrm{Alg}_1, \ldots, \mathrm{Alg}_M$ respectively
  Let $i^* \leftarrow \arg\min_{i \in \{1,\ldots,M\}} (C_i + c_{j_i})$
  Let $t \leftarrow j_{i^*}$; let $C_{i^*} \leftarrow C_{i^*} + c_t$
  Perform test $t$ and set $d_t$ to the newly determined value of bit $t$
end while
```

Algorithm 2 Non-adaptive Round Robin Algorithm for SSClass.

```
Let $C_0 \leftarrow 0$, $C_1 \leftarrow 0$
Let $d \leftarrow *^n$
repeat
  Let $j_0 \leftarrow$ next bit from $\mathrm{Alg}_0$
  Let $j_1 \leftarrow$ next bit from $\mathrm{Alg}_1$
  Let $j^* \leftarrow \arg\min_{i \in \{0,1\}} C_i + c_{j_i}$
  Query bit $i^*$ and set $d_{j^*}$ to the discovered value
until induced function $f^d$ is a constant function
return The constant value of $f^d$
```

5.3 A Round Robin Approach to Non-adaptive Evaluation

We now present an algorithm for the unit-cost case of the non-adaptive, unweighted SSClass problem. The pseudocode is presented in Algorithm 2, with Alg_1 denoting the strategy performing tests in increasing order of c_i/p_i and Alg_0 denoting the strategy performing tests in increasing order of $c_i/(1-p_i)$. We prove the following theorem.

▶ **Theorem 2.** *When all tests have unit cost, the expected cost incurred by the non-adaptive Algorithm 2 is at most 4 times the expected cost of an optimal adaptive strategy for the unweighted adaptive SSClass problem.*

By Theorem 2, Algorithm 2 is a 4-approximation for the adaptive *and* non-adaptive versions of the unit-cost unweighted SSClass problem. The theorem also implies an upper bound of 4 on the adaptivity gap for this problem. A simpler analysis shows that for arbitrary costs, Algorithm 2 achieves an approximation factor of $2(B-1)$ for the non-adaptive version of the problem. Since the k-of-n functions are essentially equivalent to score classification functions with $B = 2$, the $2(B-1)$-approximation is a 2-approximation for non-adaptive k-of-n function evaluation.

5.4 The Unanimous Vote Function: Adaptive Setting

Adaptive evaluation of the Unanimous Vote function can be done optimally using the following simple idea. Recall that querying the bits in increasing c_i/p_i order is optimal for evaluating OR, while querying in increasing $c_i/(1-p_i)$ is optimal for AND. Now consider the problem of adaptively evaluating the Unanimous Vote function. Suppose we know the optimal choice for the first test. After the first test, we have an induced SSClass problem on the remaining bits. If the first test has value 0, the induced function is equivalent to Boolean OR (mapping UNCERTAIN to 1, and NEGATIVE to 0). The subtree rooted at the root node's 0-child should be the optimal tree for evaluating OR. Specifically, the remaining bits should be tested

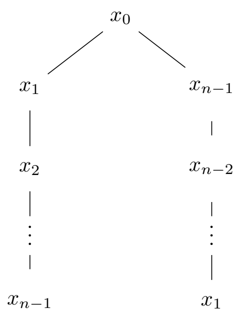

Figure 1 Decision tree T representing optimal adaptive strategy with root x_0.

in increasing order of c_i/p_i. If, instead, the first bit is 1, the induced function is equivalent to AND (mapping UNCERTAIN to 0 and POSITIVE to 1) and the remaining bits should be queried in increasing order of $c_i/(1-p_i)$.

Since we don't actually know the first bit, we can just try each bit as the root and build the rest of the tree according to the optimal OR and AND strategies. We can then calculate the expected cost of each tree, and output the tree with minimum expected cost.

For succinctness, the optimal OR and AND strategies can be represented by paths, because each performs tests in a fixed order. Figure 1 shows an example of the strategy computed by the algorithm, where the root is labeled x_0 and the OR permutation is the reversal of the AND permutation (which occurs, for example, with unit costs).

5.5 A Non-adaptive φ-approximation for the Unanimous Vote Function

A simple modification of the round robin makes the algorithm from the previous section non-adaptive, yielding a 2-approximation. But we now show how to achieve a non-adaptive φ-approximation in the unit-cost case, where $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ is the golden ratio. We call the algorithm Truncated Round Robin. We describe the algorithm by describing a subroutine which generates a permutation of input bits to query, given an initial (root) bit. The algorithm then tries all possible bits for the root and chooses the resulting permutation that achieves the lowest expected cost.

Without loss of generality, assume the first bit (the root node) is x_0, and the rest are $x_1, \ldots, x_{n-1}$, and $1 > p_1 \geq p_2 \geq \cdots \geq p_{n-1} > 0$. Fix c to be a constant such that $0 < c < \frac{1}{2}$.

The subroutine is shown in Algorithm 3. "Evaluation unknown" means tests so far were insufficient to determine the output of the Unanimous Vote function. (The output, POSITIVE, NEGATIVE, or UNCERTAIN, is not shown.)

Given x_0 as the root, the optimal adaptive strategy continues with the OR strategy (increasing $1/p_i$) when $x_0 = 0$, and the AND strategy (increasing $1/(1-p_i)$) when $x_0 = 1$. This is shown in Figure 1, where $x_0 = 0$ is the left branch and $x_0 = 1$ is the right. On the left, we stop querying when we find a bit with value 1 (or all bits are queried). On the right, we stop when we find a bit with value 0.

Let "level l" refer to the tree nodes at distance l from the root; namely, x_l and x_{n-l}. When all costs are 1, the standard round robin technique of the previous section in effect tests, for $l = 1 \ldots \lceil \frac{n-1}{2} \rceil$, the bit x_l followed by x_{n-l}. Note that the algorithm will terminate by level $\lceil \frac{n-1}{2} \rceil$ because at this point all bits will have been queried. Thus in the algorithm, $p_l \geq p_{n-l}$.

Algorithm 3 Truncated Round Robin Subroutine for Unanimous Vote Fn.

```
Require: $1 > p_1 \geq p_2 \geq \cdots \geq p_{n-1}$
  Query bit $x_0$
  Let level $l \leftarrow 1$
  while $p_{n-l} < 1 - c$ and $p_l > c$ and evaluation unknown do
    if $|p_l - 0.5| < |p_{n-l} - 0.5|$ then
      Query $x_l$ followed by $x_{n-l}$
    else
      Query $x_{n-l}$ followed by $x_l$
    end if
    $l \leftarrow l + 1$
  end while{first phase: alternate branches of tree}
  while evaluation unknown do
    if $p_l \geq p_{n-l} \geq 1 - c$ then
      Query $x_{n-l}$
    else if $c \geq p_l \geq p_{n-l}$ then
      Query $x_l$
    end if
    $l \leftarrow l + 1$
  end while{second phase: single branch in tree}
```

In the Truncated Round Robin, we proceed level by level, in two phases. The first phase concludes once we reach a level l where $p_l > p_{n-l} \geq 1 - c$ or $c \geq p_l > p_{n-l}$. Let ℓ denote this level. 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, in 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. Fixing $c = \frac{3-\sqrt{5}}{2} \approx 0.381966$ in the algorithm, the following holds.

▶ **Theorem 3.** *When all tests have unit cost, the Truncated Round Robin Algorithm achieves an approximation factor of φ for non-adaptive evaluation of the Unanimous Vote function.*

Proof. Consider the optimal adaptive strategy T. It tests a bit x_0 and then follows the optimal AND or OR strategy depending on whether $x_0 = 1$ or $x_0 = 0$. Assume the other bits are indexed so $p_1 \geq p_2 \geq \ldots \geq p_{n-1}$. Thus T is the tree in Figure 1. Let C^*_{adapt} be the expected cost of T. Let $C^*_{non-adapt}$ be the expected cost of the optimal non-adaptive strategy. Let $C_{i,TRR}$ be the cost of running the TRR subroutine in (Algorithm 3) with root x_i. We use x_0 to denote the root of T. Since the TRR algorithm tries all possible roots, its output strategy has expected cost $\min_i C_{i,TRR}$. We will prove the following claim: $C_{0,TRR} \leq \varphi C^*_{adapt}$. Since the expected cost of the optimal adaptive strategy is bounded above by the expected cost of the optimal non-adaptive strategy, the claim implies that $\min_i C_{i,TRR} \leq C_{0,TRR} \leq \varphi C^*_{adapt}$. Further, $C^*_{adapt} \leq C^*_{non-adapt}$, which proves the theorem.

We now prove the claim. We will write the expected cost of the TRR (with root x_0) as $C_{0,TRR} = 1 + E_1 + (1 - P_1)E_2$. Here, E_1 is the expected number of bits tested in T in the first phase (i.e. in levels $l < \ell$), E_2 is the expected number of variables tested among levels in T in the second phase (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 it is independent of the test outcomes.

We will write the expected cost of T (the adaptive tree which is optimal w.r.t all trees with root x_0) as $C^*_{adapt} = 1 + E'_1 + (1 - P'_1)E'_2$ where E'_1 is the expected number of bits queried in T before level ℓ, P'_1 is the probability of ending before level ℓ, and E'_2 is the expected number of bits queried 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

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

For ratio θ_1, notice that the TRR 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, $\frac{\mathrm{d}}{\mathrm{d}\,x}\left(\frac{1+2x}{1+x}\right) = \frac{1}{(1+x)^2} > 0$ for $x > 0$. For each path in 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 queried is an independent Bernoulli trial, so $E'_{1,1} \leq \frac{1}{1-c}$. Because $\frac{1+2x}{1+x}$ is increasing, 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 TRR. Further, let q_l represent the probability of querying the second bit in level l given that we have reached level l. Then, 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 queried is the bit x_i such that p_i is closest to 0.5. Notice also that in the second part of the first phase, each level has at least one variable x_i such that $p_i \in (c, 1-c)$. This also means that $1 - p_i \in (c, 1-c)$. This means that the first test performed in any given level in phase 1, part 2 will cause the TRR to terminate with probability at least c. This means that for each level l in this part of the TRR, we will 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 TRR 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 the evaluation of T we only perform one test at each level.

Consider the evaluation of tree T on an assignment. If the evaluation terminates upon reaching level l in the tree, for $l < \ell$, then the evaluation using the TRR must terminate at a level $l' \leq l$. That is, the TRR 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, $\theta_3 = \frac{(1-P_1)E_2}{(1-P_1')E_2'}$. Let $l^* = \ell$ denote the first level included in the second phase of the TRR. Without loss of generality, assume that $c \geq p_{l^*} \geq p_{n-l^*}$ so that in the TRR, the second phase queries the remaining bits in decreasing order of p_i. Thus, all bits x_i queried in the second phase satisfy $p_i \leq c$. (The argument is symmetric for the case where $p_{l^*} \geq p_{n-l^*} \geq 1-c$).

In this case, any assignments that do not cause termination in the TRR 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 left branch, for levels l^* and higher, that they would have followed in the optimal strategy T. (In fact, tests from the right branch of the tree that were previously performed in phase 1 of the TRR 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 TRR, of $Pr(x)C_2(x)$, where $C_2(x)$ is the total cost of all bits queried in phase 2 for assignment x. Let Q_0 be the subset of assignments reaching level l^* in the TRR which have $x_0 = 0$ and let Q_1 be the subset of assignments reaching level l^* in the TRR which have $x_0 = 1$. Let D_0 represent the sum over all assignments in Q_0 of $Pr(x)C_2(x)$ and let D_1 represent the sum over all assignments in Q_1 of $Pr(x)C_2(x)$. Then, letting S_{l^*} represent the set of assignments reaching level l^* in the TRR, we can rewrite the numerator of the third ratio as $\sum_{x\in S_{l^*}} Pr(x)C_2(x) = \sum_{x\in Q_0} Pr(x)C_2(x) + \sum_{x\in Q_1} Pr(x)C_2(x) = D_0 + D_1$.

The denominator of the third ratio is the sum, over all assignments x reaching level l^* in the tree, of $Pr(x)C_2'(x)$, where $C_2'(x)$ is the total cost of all bits queried in tree T at level l^* and below. Let S_{l^*}' denote the set of assignments x reaching level l^* in tree T. Next, observe that $S_{l^*} \subseteq S_{l^*}'$ since any assignment that reaches level l^* in the TRR must also reach level l^* in the tree. We can again rewrite the denominator as $\sum_{x\in S_{l^*}'} Pr(x)C_2'(x) \geq \sum_{x\in S_{l^*}} Pr(x)C_2'(x) = B_0 + B_1$ where $B_0 = \sum_{x\in Q_0} Pr(x)C_2'(x)$ and $B_1 = \sum_{x\in Q_1} Pr(x)C_2'(x)$. 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 queried in level l^* or below in the TRR is less than or equal to the number of bits queried 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 queried at level l^* or below 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 TRR will terminate on an assignment with $x_0 = 1$ when it first tests a bit that has value 0. Also note that each bit x_i 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 TRR. Since each bit queried is an independent trial, the expected number of bits queried 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 from our earlier upper bound on θ_3, namely $\theta_3 \leq \frac{D_0+D_1}{B_0+B_1}$, 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 TRR 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, running the TRR algorithm with $c = \frac{3-\sqrt{5}}{2}$ produces an expected cost of no more than φ times the expected cost of an optimal strategy. ◀

Table 1 Results for the Adaptive SSClass Problem.

	unit costs	arbitrary costs
weighted	$O(\log W)$-approx [Section 4]; $3(B-1)$ [Section 4]	$O(\log W)$-approx [Section 4]; $3(B-1)$ [Section 4]
unweighted	4-approx [Section 5.3]; $(B-1)$-approx [Section 4]	$O(\log n)$-approx; $(B-1)$-approx [Section 4]
k-of-n function	exact algorithm [known]	exact algorithm [known]
Unanimous Vote function	exact algorithm [Section 5.4]	exact algorithm [Section 5.4]

Table 2 Results for the Non-Adaptive SSClass Problem.

	unit costs	arbitrary costs
weighted	open	open
unweighted	4-approx [Section 5.3]	$2(B-1)$-approx [Section 5.3]
k-of-n function	2-approx [Section 5.3]	2-approx [Section 5.3]
Unanimous Vote function	φ-approx [Section 5.5]	2-approx [Section 5.5]

References

1 Jayadev Acharya, Ashkan Jafarpour, and Alon Orlitsky. Expected query complexity of symmetric Boolean functions. In *IEEE 49th Annual Allerton Conference on Communication, Control, and Computing*, pages 26–29, 2011.

2 Sarah R. Allen, Lisa Hellerstein, Devorah Kletenik, and Tonguç Ünlüyurt. Evaluation of monotone dnf formulas. *Algorithmica*, 77(3):661–685, 2017.

3 Eric Bach, Jérémie Dusart, Lisa Hellerstein, and Devorah Kletenik. Submodular goal value of boolean functions. *Discrete Applied Mathematics*, 238:1–13, 2018. doi:10.1016/j.dam.2017.10.022.

4 Yosi Ben-Dov. Optimal testing procedure for special structures of coherent systems. *Management Science*, 1981.

5 Endre Boros and Tonguç. Ünlüyurt. Diagnosing double regular systems. *Annals of Mathematics and Artificial Intelligence*, 26(1-4):171–191, September 1999. doi:10.1023/A:1018958928835.

6 Ming-Feng Chang, Weiping Shi, and Kent Fuchs, W.˙ Optimal diagnosis procedures for k-out-of-n structures. *IEEE Transactions on Computers*, 39(4):559–564, April 1990.

7 Hirakendu Das, Ashkan Jafarpour, Alon Orlitsky, Shengjun Pan, and Ananda Theertha Suresh. On the query computation and verification of functions. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2711–2715, 2012.

8 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):42:1–42:28, April 2016. doi:10.1145/2876506.

9 Dimitrios Gkenosis, Nathaniel Grammel, Lisa Hellerstein, and Devorah Kletenik. The stochastic score classification problem. *CoRR*, 2018. arXiv:1806.10660.

10 Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

11 Daniel Golovin and Andreas Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization (version 5). *CoRR*, abs/1003.3967, 2017. arXiv:1003.3967.

12 Nathaniel Grammel, Lisa Hellerstein, Devorah Kletenik, and Patrick Lin. Scenario submodular cover. In *Proceedings of the 14th International Workshop on Approximation and Online Algorithms*, pages 116–128. Springer, 2016.

13 Russell Greiner, Ryan Hayward, Magdalena Jankowska, and Michael Molloy. Finding optimal satisficing strategies for and-or trees. *Artificial Intelligence*, 170(1):19–58, 2006.

14 Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 205–216. Springer, 2013.

15 Jongbin Jung, Connor Concannon, Ravi Shroff, Sharad Goel, and Daniel G Goldstein. Simple rules for complex decisions. *arXiv preprint arXiv:1702.04690*, 2017.

16 Prabhanjan Kambadur, Viswanath Nagarajan, and Fatemeh Navidi. Adaptive submodular ranking. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 317–329. Springer, 2017.

17 Hemant Kowshik and PR Kumar. Optimal computation of symmetric boolean functions in collocated networks. *IEEE Journal on Selected Areas in Communications*, 31(4):639–654, 2013.

18 Feng Nan and Venkatesh Saligrama. Comments on the proof of adaptive stochastic set cover based on adaptive submodularity and its implications for the group identification problem in "group-based active query selection for rapid diagnosis in time-critical situations". *IEEE Trans. Information Theory*, 63(11):7612–7614, 2017. `doi:10.1109/TIT.2017.2749505`.

19 Salam Salloum. *Optimal testing algorithms for symmetric coherent systems.* PhD thesis, University of Southern California, 1979.

20 Salam Salloum and Melvin Breuer. 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`.

21 Salam Salloum and Melvin A. Breuer. Fast optimal diagnosis procedures for k-out-of-n:g systems. *IEEE Transactions on Reliability*, 46(2):283–290, Jun 1997. `doi:10.1109/24.589958`.

22 Sahil Singla. The price of information in combinatorial optimization. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2523–2532. SIAM, 2018.

23 Truyen Tran, Wei Luo, Dinh Phung, Jonathan Morris, Kristen Rickard, and Svetha Venkatesh. Preterm birth prediction: Deriving stable and interpretable rules from high dimensional data. In *Conference on Machine Learning in Healthcare, LA, USA*, 2016.

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

25 Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, 2016.

26 Berk Ustun and Cynthia Rudin. Optimized risk scores. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1125–1134. ACM, 2017.

27 Jiaming Zeng, Berk Ustun, and Cynthia Rudin. Interpretable classification models for recidivism prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 180(3):689–722, 2017.