

Approximation Algorithms for Stochastic Boolean Function Evaluation and Stochastic Submodular Set Cover

Amol Deshpande
University of Maryland
amol@cs.umd.edu

Lisa Hellerstein
Polytechnic Institute of NYU
hstein@poly.edu

Devorah Kletenik
Polytechnic Institute of NYU
dkletenik@cs.poly.edu

Abstract

We present approximation algorithms for two problems: Stochastic Boolean Function Evaluation (SBFE) and Stochastic Submodular Set Cover (SSSC). Our results for SBFE problems are obtained by reducing them to SSSC problems through the construction of appropriate utility functions.

We give a new algorithm for the SSSC problem that we call Adaptive Dual Greedy. We use this algorithm to obtain a 3-approximation algorithm solving the SBFE problem for linear threshold formulas. We also get a 3-approximation algorithm for the closely related Stochastic Min-Knapsack problem, and a 2-approximation for a natural special case of that problem. In addition, we prove a new approximation bound for a previous algorithm for the SSSC problem, Adaptive Greedy.

We consider an approach to approximating SBFE problems using existing techniques, which we call the Q -value approach. This approach easily yields a new result for evaluation of CDNF formulas, and we apply variants of it to simultaneous evaluation problems and a ranking problem. However, we show that the Q -value approach provably cannot be used to obtain a sublinear approximation factor for the SBFE problem for linear threshold formulas or read-once DNF.

1 Introduction

Stochastic Boolean Function Evaluation (SBFE) is the problem of determining the value of a given Boolean function f on an unknown input x , when each bit x_i of x can only be determined by paying a cost c_i . The assumption is that x is drawn from a given product distribution, and the goal is to minimize expected cost. SBFE problems arise in diverse application areas. For example, in medical diagnosis, the x_i might correspond to medical tests, and $f(x) = 1$ if the patient has a particular disease. In database query optimization, f could correspond to a Boolean query on predicates corresponding to x_1, \dots, x_n , that has to be evaluated for every tuple in the database to find tuples that satisfy the query [30, 34, 16, 42]. In Operations Research, the SBFE problem is known as “sequential testing” of Boolean functions [43]. In learning

theory, the SBFE problem has been studied in the context of learning with attribute costs [33].

We focus on developing approximation algorithms for SBFE problems. There have been previous papers on exact algorithms for these problems, but there is very little work on approximation algorithms [43, 33]. Our approach is to reduce the SBFE problems to Stochastic Submodular Set Cover (SSSC). The SSSC problem was introduced by Golovin and Krause, who gave an approximation algorithm for it called *Adaptive Greedy* [22].¹

Adaptive Greedy is a generalization of the greedy algorithm for the classical Set Cover problem. We present a new algorithm for the SSSC problem, which we call Adaptive Dual Greedy. We prove that it achieves an approximation factor of α , where α is a ratio that depends on the cover constructed by the algorithm. Adaptive Dual Greedy is an extension of the Dual Greedy algorithm for Submodular Set Cover due to Fujito, which is a generalization of Hochbaum’s primal-dual algorithm for the classical Set Cover Problem [19, 20].

We use our new Adaptive Dual Greedy algorithm to obtain a 3-approximation algorithm solving the SBFE problem for linear threshold formulas.

The SBFE problem for evaluating linear threshold formulas is closely related to a stochastic version of the Min-Knapsack problem. There are a number of results on the Stochastic (Max) Knapsack problem and on (deterministic) Min-Knapsack, but there has been very little work on stochastic versions of Min-Knapsack.

We define the *Stochastic Min-Knapsack* problem to be the problem of adaptively filling an initially empty knapsack with items from a given set of n items, until the total size of the items in the knapsack is at least a fixed threshold θ , or all items have been used. The size of each item is a random variable and the size of an item is not revealed until it is placed in the knapsack. Each item has a non-negative cost. The goal is to minimize the total cost of the items in the knapsack. We give more details in

¹Golovin and Krause called the problem Stochastic Submodular Coverage, not Stochastic Submodular Set Cover, because the cover is not formed using sets. Our choice of name is for consistency with terminology of Fujito [20].

Section 2.

A minor modification of our algorithm for linear threshold evaluation yields a 3-approximation algorithm for Stochastic Min-Knapsack. We also get a 2-approximation algorithm for the special case in which the distributions on the sizes are such that the threshold θ can always be reached or exceeded. To our knowledge, these are the first constant-factor approximations for Stochastic Min-Knapsack. An $O(\log \theta)$ -approximation algorithm, for the general case, follows from work of Im et al. on the Stochastic Submodular Ranking Problem [31], when the item sizes and threshold θ are non-negative integers.

We give a new bound on the approximation factor achieved by the Adaptive Greedy algorithm of Golovin and Krause. We prove that Adaptive Greedy achieves an $2(\ln P + 1)$ -approximation in the binary case (and $k(\ln P + 1)$ in the k -ary case) where P is the maximum utility that can be contributed by a single item. Our bound generalizes Wolsey's bound for (non-adaptive) submodular set cover [44], except for an additional factor of 2. Wolsey's bound generalized the $(\ln s + 1)$ bound for standard set cover, where s is the maximum size of one of the input subsets (cf. [20]).

We note that our work on SBFE problems suggests many open questions, including approximation algorithms for other classes of Boolean functions, proving hardness results, and determining adaptivity gaps.

Before presenting our results on evaluating linear threshold formulas using our new Adaptive Dual Greedy, we consider solving SBFE problems using an approach exploiting existing techniques. In this approach, we reduce the SBFE problem to an SSSC problem, solve the SSSC problem using the existing Adaptive Greedy algorithm, and apply the bound of Golovin and Krause for that algorithm [22]. We formalize this approach and call it the Q -value approach. We show that the Q -value approach easily yields an $O(\log kd)$ -approximation algorithm for CDNF formulas (or decision trees), where k is the number of clauses in the CNF and d is the number of terms in the DNF. Previously, Kaplan et al. gave an algorithm also achieving an $O(\log kd)$ approximation, but only for *monotone* CDNF formulas, unit costs, and the uniform distribution [33].²

We then show that although the Q -value approach yields a good approximation result for CDNF formula evaluation, it provably cannot yield a sublinear approximation result for evaluating linear threshold formulas. Thus it cannot be used to obtain the 3-approximation algorithm obtained in this paper by using Adaptive Dual Greedy algorithm. We also prove that the Q -value approach cannot yield a sublinear approximation result for

²Although our result solves a more general problem than Kaplan et al., they give their $O(\log kd)$ approximation factor in terms of expected *certificate cost*, which lower bounds the expected cost of the optimal strategy. See Section 2.

evaluating read-once DNF, even though there is an exact algorithm for that problem [33, 25].

At the end of this paper, we include some additional applications of variants of the Q -value approach, using bounds on Adaptive Greedy and Adaptive Dual Greedy. We give two algorithms with approximation factors of $O(\log mD_{avg})$ and D_{max} , respectively, solving the problem of simultaneous evaluation of m linear threshold formulas. Here D_{avg} and D_{max} are the average and maximum, over the m formulas, of the sum of the magnitude of the coefficients. These results generalize results of Liu et al. for *shared filter ordering* [35]. We also improve one of Liu's results for that problem.

We also give an $O(\log(mD_{max}))$ -approximation algorithm for ranking a set of m linear functions $a_1x_1 + \dots + a_nx_n$ (not linear *threshold* functions), defined over $\{0, 1\}^n$, by their output values, in our stochastic setting. This problem arises in Web search and database query processing. For example, we might need to rank a set of documents or tuples by their "scores", where the linear functions compute the scores over a set of unknown properties such as user preferences or data source reputations.

2 Problem Definitions and Related Work

Formally, the input to the SBFE problem is a representation of a Boolean function $f(x_1, \dots, x_n)$ from a fixed class of representations C , a probability vector $p = (p_1, \dots, p_n)$, where $0 < p_i < 1$, and a real-valued cost vector (c_1, \dots, c_n) , where $c_i \geq 0$. An algorithm for this problem must compute and output the value of f on an $x \in \{0, 1\}^n$, drawn randomly from product distribution D_p , such that $p_i = \text{Prob}[x_i = 1]$. However, it is not given access to x . Instead, it can discover the value of any x_i by "testing" it, at a cost of c_i . The algorithm must perform the tests sequentially, each time choosing the next test to perform. The algorithm can be adaptive, so the choice of the next test can depend on the outcomes of the previous tests. The expected cost of the algorithm is the cost it incurs on a random x from D_p . (Since each p_i is strictly between 0 and 1, the algorithm must continue doing tests until it has obtained a 0-certificate or 1-certificate for the function.) The algorithm is optimal if it has minimum expected cost with respect to D_p . The running time of the algorithm is the (worst-case) time it takes to determine the next variable to be tested, or to compute the value of $f(x)$ after the last test. The algorithm corresponds to a Boolean decision tree (strategy) computing f .

If f is given by its truth table, the SBFE Problem can be exactly solved in time polynomial in the size of the truth table, using dynamic programming, as in [26, 39]. The following algorithm solves the SBFE problem with an approximation factor of n for any function f , even under arbitrary distributions: Test the variables in increasing cost order (cf. [33]). We thus consider a factor of n approximation to be trivial.

We define the Stochastic Min-Knapsack problem formally as follows. The input is a set of n items with associated non-negative costs c_1, \dots, c_n . For each item i , we are also given a probability distribution on the size s_i of the item. We assume in this paper that each s_i has only a finite number of possible values. We expect that our results will also apply to a wide-range of continuous distributions for s_i , but defer that to future work.

We assume that the sizes of the items are independent. The problem is to incrementally choose items to place in a subset S , starting from $S = \emptyset$, until either $\sum_{j \in S} s_j \geq \theta$ or S contains all n items. The choice of the next item can depend on the sizes of the previous items chosen. The size s_i of the i th item is not revealed until the item is chosen for inclusion in S , and once an item is added to S , it cannot be removed. The goal is to minimize the expected value of $\sum_{j \in S} c_j$.

We now review related work on the SBFE problem. There is a well-known algorithm that exactly solves the SBFE problem for disjunctions: test the x_i in increasing order of ratio c_i/p_i (see, e.g., [21]). A symmetric algorithm works for conjunctions. There is also a poly-time exact algorithm for evaluating a k -of- n function (i.e., a function that evaluates to 1 iff at least k of the x_i are equal to 1) [40, 4, 41, 11]. There is a poly-time exact algorithm for evaluating a read-once DNF formula f , but the complexity of the problem is open when f is a general read-once formula [9, 25, 24]. The SBFE problem is NP-hard for linear threshold formulas [13], but for the special case of unit costs and uniform distribution, testing the variables in decreasing order of the magnitude of their coefficients is optimal [8, 18]. A survey by Ünlüyurt [43] covers other results on exactly solving the SBFE problem.

There is a *sample* version of the evaluation problem, where the input is a sample of size m of f (i.e., a set of m pairs $(x, f(x))$), and the problem is to build a decision tree that computes f correctly on the x in the sample that minimizes the average cost of evaluation over the sample. Golovin et al. and Bellala et al. developed $O(\log m)$ approximation algorithms for arbitrary f [23, 3], and there is a 4-approximation algorithm when f is a conjunction [2, 17, 38]. Moshkov and Chikalov proved a related bound in terms of a combinatorial measure of the sample [37]. Moshkov gave an $O(\log m)$ -algorithm for a worst-case cost variant of this problem [36].

A number of non-adaptive versions of standard and submodular set cover have been studied. For example, Iwata and Nagano [32] studied the “submodular cost set cover” problem, where the cost of the cover is a submodular function that depends on which subsets are in the cover. Beraldi and Ruszczynski addressed a set cover problem where the set of elements covered by each input subset is a random variable, and full coverage must be achieved with a certain probability [5].

The $O(\log kd)$ approximation factor proved by Ka-

plan et al. for the problem of evaluating monotone CDNF (with unit costs, uniform distribution) was given in terms of the expected certificate cost, rather than in terms of the expected cost of the optimal strategy. The gap between expected certificate cost and expected cost of the optimal strategy can be large: e.g., for disjunction evaluation, with unit costs, where $\text{Prob}[x_i = 1]$ is $1/(i + 1)$, the first measure is constant, while the second is $\Omega(\log n)$.

Kaplan et al. also considered the problem of minimizing the expected cost of evaluating a Boolean function f with respect to a given *arbitrary* probability distribution, where the distribution is given by a conditional probability oracle [33]. In the work of Kaplan et al., the goal of evaluation differs slightly from ours in that they require the evaluation strategy to output an “explanation” of the function value upon termination. They give as an example the case of evaluating a DNF that is identically true; they require testing of the variables in one term of the DNF in order to output that term as a certificate. In contrast, under our definitions, the optimal strategy for evaluating an identically true DNF formula is a zero-cost one that simply outputs “true” and performs no tests.

Charikar et al. [12] considered the problem of minimizing the worst-case ratio between the cost of evaluating f on an input x , and the minimum cost of a certificate contained in x . There are also papers on building *identification* trees of minimum average cost, given $S \subseteq \{0, 1\}^n$, but that problem is fundamentally different than function evaluation because each $x \in S$ must have its own leaf (cf. [1]).

There are several previous combinatorial and non-combinatorial 2-approximation algorithms for (deterministic) Min-Knapsack (cf. [28]), and this problem also has a PTAS [10]. Han and Makino considered an on-line version of Min-Knapsack where the items are given one-by-one over time [28]. There are a number of constant-factor approximation algorithms for Stochastic (Max) Knapsack problems, including a $(2 + \epsilon)$ -approximation in a recent paper of Bhalgat [14, 7, 6]. Derman et al. consider a version of stochastic Min-Knapsack where multiple copies of each item can be added to the cover [15].

3 Preliminaries

Basic notation and definitions. A table with the main notation used in this paper is provided in Appendix A.

A partial assignment is a vector $b \in \{0, 1, *\}^n$. We view b as an assignment to variables x_1, \dots, x_n . We will use $b \in \{0, 1\}^n$ to represent the outcomes of binary tests, where for $l \in \{0, 1\}$, $b_i = l$ indicates that test i was performed and had outcome l , and $b_i = *$ indicates that test i was not performed.

For partial assignments $a, b \in \{0, 1, *\}^n$, a is an *extension* of b , written $a \sim b$, if $a_i = b_i$ for all $b_i \neq *$. We also say that b is *contained* in a . Given Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, a partial assignment

$b \in \{0, 1, *\}^n$ is a 0-certificate (1-certificate) of f if $f(a) = 0$ ($f(a) = 1$) for all a such that $a \sim b$. Given a cost vector $c = (c_1, \dots, c_n)$, the cost of a certificate b is $\sum_{j:b_j \neq *} c_j$.

Let $N = \{1, \dots, n\}$. In what follows, we assume that utility functions are integer-valued. In the context of standard work on submodularity, a *utility function* is a function $g : 2^N \rightarrow \mathbb{Z}_{\geq 0}$. Given $S \subseteq N$ and $j \in N$, $g_S(j)$ denotes the quantity $g(S \cup \{j\}) - g(S)$.

We will also use the term *utility function* to refer to a function $g : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ defined on partial assignments. Let $g : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ be such a utility function, and let $b \in \{0, 1, *\}^n$. We define $g(S, b) = g(b')$ where b' is the partial assignment such that $b'_i = b_i$ for $i \in S$, and $b'_i = *$ otherwise. For $j \in N$, we define $g_{S,b}(j) = g(S \cup \{j\}, b) - g(S, b)$.

For $l \in \{0, 1, *\}$, the quantity $b_{x_i \leftarrow l}$ denotes the partial assignment that is identical to b except that $b_i = l$. We define $g_b(i, l) = g(b_{x_i \leftarrow l}) - g(b)$ if $b_i = *$, and $g_b(i, l) = 0$ otherwise. When b represents test outcomes, and test i has not been performed yet, $g_b(i, l)$ is the change in utility that would result from adding test i with outcome l .

Given probability vector $p = (p_1, \dots, p_n)$, we use $x \sim D_p$ to denote a random x drawn from product distribution D_p . For fixed D_p , $b \in \{0, 1, *\}^n$, and $i \in N$, we use $E[g_b(i)]$ to denote the expected increase in utility that would be obtained by testing i . In the binary case, $E[g_b(i)] = p_i g_b(i, 1) + (1 - p_i) g_b(i, 0)$. Note that $E[g_b(i)] = 0$ if $b_i \neq *$. For $b \in \{0, 1, *\}^n$, let $p(b)$ be the probability that a random x drawn from D_p will have the same values as b for all j such that $b_j \neq *$. That is, $p(b) = (\prod_{j:b_j=1} p_j) (\prod_{j:b_j=0} (1 - p_j))$.

Utility function $g : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ is *monotone* if for $b \in \{0, 1, *\}^n$, $i \in N$ such that $b_i = *$, and $l \in \{0, 1\}$, $g(b_{x_i \leftarrow l}) - g(b) \geq 0$; in other words, additional information can only increase utility. Utility function g is *submodular* if for all $b, b' \in \{0, 1, *\}^n$ and $l \in \{0, 1\}$, $g(b_{x_i \leftarrow l}) - g(b) \geq g(b'_{x_i \leftarrow l}) - g(b')$ whenever $b' \sim b$ and $b_i = b'_i = *$. In the testing context, if the n test outcomes are predetermined, submodularity means that the value of a given test (measured by the increase in utility) will not increase if we delay that test until later.

The Stochastic Submodular Set Cover (SSSC) problem. The SSSC problem is similar to the SBFE problem, except that the goal is to achieve a cover. Let $\mathcal{O} = \{0, 1, \dots, k - 1\}$ be a finite set of k states, where $k \geq 2$ and $*$ $\notin \mathcal{O}$. In what follows, we will assume $k = 2$ unless otherwise noted. However, we will briefly mention extensions to the k -ary case where $k > 2$, and to arbitrary sets \mathcal{O} . To simplify the exposition, we define the SSSC problem in terms of integer valued utility functions.

In the (binary) SSSC problem, the input consists of the set N , a cost vector (c_1, \dots, c_n) , where each $c_j \geq 0$,

a probability vector $p = (p_1, \dots, p_n)$ where $p \in [0, 1]^n$, an integer $Q \geq 0$, and a utility function $g : (\mathcal{O} \cup \{*\})^n \rightarrow \mathbb{Z}_{\geq 0}$. Further, $g(x) = 0$ if x is the vector that is all $*$'s, and $g(x) = Q$ if $x \in \mathcal{O}^n$. We call Q the *goal utility*. We say that $b \in (\mathcal{O} \cup \{*\})^n$ is a *cover* if $g(b) = Q$. The *cost* of cover b is $\sum_{j:b_j \neq *} c_j$.

Each item $j \in N$ has a state $x_j \in \mathcal{O}$. We sequentially choose items from N . When we choose item j , we observe its state x_j (we “test” j). The states of items chosen so far are represented by a partial assignment $b \in (\mathcal{O} \cup \{*\})^n$. When $g(b) = Q$, we have a cover, and we can output it. The goal is to determine the order in which to choose the items, while minimizing the expected testing cost with respect to distribution D_p . We assume that an algorithm for this problem will be executed in an on-line setting, and that it can be adaptive.

SSSC is a generalization of Submodular Set Cover (SSC), which is a generalization of the standard (weighted) Set Cover problem, which we call Classical Set Cover. In Classical Set Cover, the input is a finite ground set X , a set $F = \{S_1, \dots, S_m\}$ where each $S_j \subseteq X$, and a cost vector $c = (c_1, \dots, c_m)$ where each $c_j \geq 0$. The problem is to find a min-cost “cover” $F' \subseteq F$ such that $\bigcup_{S_j \in F'} S_j = X$, and the cost of F' is $\sum_{S_j \in F'} c_j$. In SSC, the input is a cost vector $c = (c_1, \dots, c_n)$, where each $c_j \geq 0$, and a utility function $g : 2^N \rightarrow \mathbb{Z}_{\geq 0}$ such that g is monotone and submodular, $g(\emptyset) = 0$, and $g(N) = Q$. The goal is to find a subset $S \subseteq N$ such that $g(S) = Q$ and $\sum_{j \in S} c_j$ is minimized. SSC can be viewed as a special case of SSSC in which each p_j is equal to 1.

The Adaptive Greedy algorithm for Stochastic Submodular Set Cover. The Classical Set Cover problem has a simple greedy approximation algorithm that chooses the subset with the “best bang for the buck” – i.e., the subset covering the most new elements per unit cost. The generalization of this algorithm to SSC, due to Wolsey, chooses the element that adds the maximum additional utility per unit cost [44]. The Adaptive Greedy algorithm of Golovin and Krause, for the SSSC problem, is a further generalization. It chooses the element with the maximum *expected* increase in utility per unit cost. (Golovin and Krause actually formulated Adaptive Greedy for use in solving a somewhat more general problem than SSSC, but here we describe it only as it applies to SSSC.)

Golovin and Krause proved that Adaptive Greedy is a $(\ln Q + 1)$ -approximation algorithm, where Q is the goal utility. We will make repeated use of this bound.

4 Function Evaluation and the SSSC Problem

4.1 The Q -value approach and CDNF Evaluation.

Definition: Let $f(x_1, \dots, x_n)$ be a Boolean function. Let $g : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ be a utility function. We say that g is *assignment feasible* for f , with goal value Q , if (1) g

is monotone and submodular, (2) $g(*, *, \dots, *) = 0$, and (3) for $b \in \{0, 1, *\}^n$, $g(b) = Q$ iff b is a 0-certificate or a 1-certificate of f .

We will use the following approach to solving SBFE problems, which we call the *Q-value approach*. To evaluate f , we construct an assignment feasible utility function g for f with goal value Q . We then run Adaptive Greedy on the resulting SSSC problem. Because $g(b) = Q$ iff b is either a 0-certificate or a 1-certificate of f , the decision tree that is (implicitly) output by Adaptive Greedy is a solution to the SBFE problem for f . By the bound on Adaptive Greedy, this solution is within a factor of $(\ln Q + 1)$ of optimal.

The challenge in using the above approach is in constructing g . Not only must g be assignment feasible, but Q should be subexponential, to obtain a good approximation bound. We will use the following lemma, from Guillory and Bilmes, in our construction of g .

Lemma 4.1 [27] *Let $g_0 : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$, $g_1 : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$, and $Q_0, Q_1 \in \mathbb{Z}_{\geq 0}$ be such that g_0 and g_1 are monotone, submodular utility functions, $g_0(*, *, \dots, *) = g_1(*, *, \dots, *) = 0$, and $g_0(a) \leq Q_0$ and $g_1(a) \leq Q_1$ for all $a \in \{0, 1, *\}^n$.*

*Let $Q_{\vee} = Q_0 Q_1$ and let $g_{\vee} : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ be such that $g_{\vee}(b) = Q_{\vee} - (Q_0 - g_0(b))(Q_1 - g_1(b))$.*

*Let $Q_{\wedge} = Q_0 + Q_1$ and let $g_{\wedge} : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ be such that $g_{\wedge}(b) = g_0(b) + g_1(b)$.*

Then g_{\vee} and g_{\wedge} are monotone and submodular, and $g_{\vee}(, \dots, *) = g_{\wedge}(*, \dots, *) = 0$. For $b \in \{0, 1, *\}^n$, $g_{\vee}(b) = Q_{\vee}$ iff $g_0(b) = Q_0$ or $g_1(b) = Q_1$, or both. Further, $g_{\wedge}(b) = Q_{\wedge}$ iff $g_0(b) = Q_0$ and $g_1(b) = Q_1$.*

Using the Q -value approach, it is easy to obtain an algorithm for evaluating CDNF formulas. A CDNF formula for Boolean function f is a pair (ϕ_0, ϕ_1) where ϕ_0 and ϕ_1 are CNF and DNF formulas for f , respectively.

Theorem 4.1 *There is a polynomial-time $O(\log kd)$ -approximation algorithm solving the SBFE problem for CDNF formulas, where k is the number of clauses in the CNF, and d is the number of terms in the DNF.*

Proof Let ϕ_0 be the CNF and ϕ_1 be the DNF. Let f be the Boolean function defined by these formulas. Let k and d be, respectively, the number of clauses and terms of ϕ_0 and ϕ_1 . Let $g_0 : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ be such that for $a \in \{0, 1, *\}^n$, $g_0(a)$ is the number of terms of ϕ_1 set to 0 by a (i.e. terms with a literal x_i such that $a_i = 0$, or a literal $\neg x_i$ such that $a_i = 1$). Similarly, let $g_1(a)$ be the number of clauses of ϕ_0 set to 1 by a . Clearly, g_0 and g_1 are monotone and submodular. Partial assignment b is a 0-certificate of f iff $g_0(b) = d$ and a 1-certificate of f iff $g_1(b) = k$. Applying the disjunctive construction of Lemma 4.1 to g_1 and g_0 , yields a utility function g that is assignment feasible for f with goal value $Q = kd$. Applying Adaptive Greedy and its $(\ln Q + 1)$ bound yields the theorem. \square

Given a decision tree for a Boolean function f , a CNF (DNF) for f can be easily computed using the paths to the 0-leaves (1-leaves) of the tree. Thus the above theorem gives an $O(\ln t)$ approximation algorithm for evaluating decision trees, where t is the number of leaves.

4.2 Limitations of the Q-value approach The Q -value approach depends on finding an assignment feasible utility function g for f . We first demonstrate that such a generic g exists for all Boolean functions f . For $i \in \{0, 1\}$, let $Q_i = |\{a \in \{0, 1, *\}^n \mid f(a) = i\}|$. For partial assignment b , let $g_i(b) = Q_i - |\{a \in \{0, 1, *\}^n \mid a \sim b, f(a) = i\}|$ with goal value Q_i . Then g_0, Q_0, g_1 and Q_1 obey the properties of Lemma 4.1. Apply the disjunctive construction in that lemma, and let g be the resulting utility function. Then g is assignment feasible for f with goal value $Q = Q_1 Q_0$. In fact, this g is precisely the utility function that would be constructed by the approximation algorithm of Golovin et al. [23], for computing a consistent decision tree of min-expected cost with respect to a sample, if we take the sample to be the set of all 2^n entries $(x, f(x))$ in the truth table of f . The goal value Q of this g is $2^{\theta(n)}$, so in this case the bound for Adaptive Greedy, $(\ln Q + 1)$, is linear in n .

Since we want a sublinear approximation factor, we would instead like to construct an assignment-feasible utility function for f whose Q is sub-exponential in n . However, we now show this is impossible even for some simple Boolean functions f . We begin by introducing the following combinatorial measure of a Boolean function, which we call its Q -value.

Definition: The Q -value of a Boolean function $f : \{0, 1, *\}^n \rightarrow \{0, 1\}$ is the minimum integer Q such that there exists an assignment feasible utility function g for f with goal value Q .

The generic g above shows that the Q -value of every n -variable Boolean function is at most $2^{O(n)}$.

We now prove lower bounds on Q for some particular functions. First, consider the Boolean function $f : \{0, 1, *\}^n \rightarrow \{0, 1\}$, where n is even, represented by the read-once DNF formula $\phi = t_1 \vee t_2 \vee \dots \vee t_{n/2}$ where each $t_i = x_i x_{n/2+i}$. This function has the following simple property: if you set one variable in each of the first m terms of ϕ to be 1, where $1 \leq m \leq n$, this is not sufficient to force the value of the function to be 0. However, if you then set the remaining variable in the m th term to 1, this does force the function value to 1. We can use this property to prove a lower bound of $2^{n/2}$ on the Q -value of f . Rather than showing this directly, however, we prove a slightly more general lemma.

Lemma 4.2 *Let $f(x_1, \dots, x_n)$ be a Boolean function, where n is even. Further, let f be such that for all $n' \leq n/2$, and for all $b \in \{0, 1, *\}^n$, if $b_i = b_{n/2+i} = *$ for all $i \in \{n' + 1, \dots, n/2\}$, the following properties hold: (1) if for all $i \in \{1, \dots, n'\}$, exactly one of b_i and*

$b_{n/2+i}$ is equal to $*$ and the other is equal to 1, then b is not a 0-certificate or a 1-certificate of f and (2) if for all $i \in \{1, \dots, n' - 1\}$, exactly one of b_i and $b_{n/2+i}$ is equal to $*$ and the other is equal to 1, and $b_{n'} = b_{n/2+n'} = 1$, then b contains a 1-certificate of f . Then the Q -value of f is at least $2^{n/2}$.

Proof Let f have the properties specified in the lemma. For bitstrings $r, s \in \{0, 1\}^l$, where $0 \leq l \leq n/2$, let $d_{r,s} \in \{0, 1, *\}^n$ be such that $d_i = r_i$ and $d_{n/2+i} = s_i$ for $i \in \{1, \dots, l\}$, and $d_i = *$ for all other i . Suppose g is an assignment feasible utility function for f with goal value Q . We prove the following claim. Let $0 \leq l \leq n/2$. Then there exists $r, s \in \{0, 1, *\}^l$ such that $0 \leq Q - g(d_{r,s}) \leq Q/2^l$, and for all $i \in \{1, \dots, l\}$, either $r_i = 1$ and $s_i = *$, or $r_i = *$ and $s_i = 1$.

We prove the claim by induction on l . It clearly holds for $l = 0$. For the inductive step, assume it holds for l . We show it holds for $l + 1$. Let $r, s \in \{0, 1, *\}^l$ be as guaranteed by the assumption, so $Q - g(d_{r,s}) \leq n/2^l$.

For $\sigma \in \{0, 1, *\}$, $r\sigma$ denotes the concatenation of bitstring r with σ , and similarly for $s\sigma$. By the conditions on f given in the lemma, $d_{r,s}$ is not a 0 or 1-certificate of f . However, $d_{r1,s1}$ is a 1-certificate of f and so $g(d_{r1,s1}) = Q$. If $Q - g(d_{r1,s*}) \leq Q/2^{l+1}$, then the claim holds for $l + 1$, because $r1, s*$ have the necessary properties. Suppose $Q - g(d_{r1,s*}) > Q/2^{l+1}$. Then, because $g(d_{r1,s1}) = Q$, $g(d_{r1,s1}) - g(d_{r1,s*}) > Q/2^{l+1}$. Note that $d_{r1,s1}$ is the extension of $d_{r1,s*}$ produced by setting $d_{n/2+l+1}$ to 1. Similarly, $d_{r*,s1}$ is the extension of $d_{r*,s*}$ produced by setting $d_{n/2+l+1}$ to 1. Therefore, by the submodularity of g , $g(d_{r*,s1}) - g(d_{r*,s*}) \geq g(d_{r1,s1}) - g(d_{r1,s*})$, and thus $g(d_{r*,s1}) - g(d_{r*,s*}) \geq Q/2^{l+1}$.

Let $A = g(d_{r*,s1}) - g(d_{r*,s*})$ and $B = Q - g(d_{r*,s1})$. Thus $A \geq Q/2^{l+1}$, and $A + B = Q - g(d_{r*,s*}) = Q - g(d_{r,s}) \leq Q/2^l$ where the last inequality is from the original assumptions on r and s . It follows that $B = Q - g(d_{r*,s1}) \leq Q/2^{l+1}$, and the claim holds for $l + 1$, because $r*, s1$ have the necessary properties.

Taking $l = n/2$, the claim says there exists $d_{r,s}$ such that $Q - g(d_{r,s}) \leq Q/2^{n/2}$. Since g is integer-valued, $Q \geq 2^{n/2}$. \square

The above lemma immediately implies the following theorem.

Theorem 4.2 Let n be even. Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be the Boolean function represented by the read-once DNF formula $\phi = t_1 \vee t_2 \vee \dots \vee t_{n/2}$ where each $t_i = x_i x_{n/2+i}$. The Q -value of f is at least $2^{n/2}$.

The above theorem shows that the Q -value approach will not yield a good approximation bound for either read-once DNF formulas or for DNF formulas with terms of length 2.

In the next theorem, we show that there is a particular linear threshold function whose Q -value is at least $2^{n/2}$.

It follows that the Q -value approach will not yield a good approximation bound for linear-threshold formulas either.

We note that the function described in the next theorem has been studied before. As mentioned in [29], there is a lower bound of essentially $2^{n/2}$ on the size of the largest integer coefficients in any representation of the function as a linear threshold formula with integer coefficients.

Theorem 4.3 Let $f(x_1, \dots, x_n)$ be the function defined for even n , whose value is 1 iff the number represented in binary by bits $x_1 \dots x_{n/2}$ is strictly less than the number represented in binary by bits $x_{n/2+1}, \dots, x_n$, and 0 otherwise. The Q -value of f is at least $2^{n/2}$.

Proof We define a new function: $f'(x_1, \dots, x_n) = f(\neg x_1, \dots, \neg x_{n/2}, x_{n/2+1}, \dots, x_n)$. That is, $f'(x_1, \dots, x_n)$ is computed by negating the assignments to the first $n/2$ variables, and then computing the value of f on the resulting assignment. Function f' obeys the conditions of Lemma 4.2, and so has Q -value at least $2^{n/2}$. Then f also has Q -value at least $2^{n/2}$, because the Q -value is not changed by the negation of input variables. \square

Given the limitations of the Q -value approach we can ask whether there are good alternatives. Our new bound on Adaptive Greedy is $O(\log P)$, where P is the maximum amount of utility gained by testing a single variable x_i , so we might hope to use P -value in place of Q -value. However, this does not help much: testing all n variables yields utility Q , so testing one of them alone must yield utility at least Q/n , implying that $P \geq Q/n$. Another possibility might be to exploit the fact that Golovin and Krause's bounds on Adaptive Greedy apply to a more general class of utility functions than the assignment feasible utility functions, but we do not pursue that possibility. Instead, we give a new algorithm for the SSSC problem.

5 Adaptive Dual Greedy

We now present ADG, our new algorithm for the binary version of the SSSC problem. It easily extends to the k -ary version, where $k > 2$, with no change in the approximation bound. Like Fujito's Dual Greedy algorithm for the (non-adaptive) SSC problem, it is based on Wolsey's IP for the (deterministic) Submodular Set Cover Problem. We present Wolsey's IP in Figure 1.

$$\begin{array}{ll} \min & \sum_{j \in N} c_j x_j \\ \text{s.t.} & \sum_{j \in N} (g(S \cup \{j\}) - g(S)) x_j \geq Q - g(S) \quad \forall S \subseteq N \\ & x_j \in \{0, 1\} \quad \forall j \in N \end{array}$$

Figure 1: Wolsey's IP for submodular set cover

Wolsey proved that an assignment $x \in \{0, 1\}^n$ to the variables in this IP is feasible iff $\{j | x_j = 1\}$ is a cover

for the associated Submodular Set Cover instance, i.e., iff $g(\{j|x_j = 1\}) = Q$. We call this Wolsey's property.

In Figure 2, we present a new LP, based on Wolsey's IP, which we call LP1. There are $n2^{n-1}$ variables x_w in this LP, one for each partial assignment w such that $w_j = *$ for exactly one value of j . Let W be the set of such assignments, so $W = \{w \in \{0, 1, *\}^n \mid w_j = * \text{ for exactly one value of } j\}$. For $w \in W$, let $j(w)$ denote the $j \in N$ where $w_j = *$. Further, let $w^{(0)}$ and $w^{(1)}$ denote the extensions of w obtained from w by setting $w_{j(w)}$ to 0 and 1, respectively. Finally, for $a \in \{0, 1\}^n$, $j \in N$, let a^j denote the partial assignment obtained from a by setting a_j to $*$. Note that, for any fixed $a \in \{0, 1\}^n$, the subset of constraints involving a , one for each $S \subseteq N$, is precisely the set of constraints of Wolsey's IP, if we take the utility function to be g_a such that $g_a(S) = g(S, a)$.

We will rely on the following observation, which we call the Neighbor Property: Let T be a decision tree solving the SSSC problem. Given two assignments $a, a' \in \{0, 1\}^n$ differing only in bit j , either T tests j on both input a and input a' , or on neither.

$$\begin{array}{ll} \min & \sum_{w \in W} c_{j(w)} p(w) x_w \\ \text{s.t.} & \\ & \sum_{j \in N} g_{S,a}(j) x_{a^j} \geq Q - g(S, a) \\ & \qquad \qquad \qquad \forall a \in \{0, 1\}^n, S \subseteq N \\ & x_w \geq 0 \qquad \qquad \qquad \forall w \in W \end{array}$$

Figure 2: LP1: the Linear Program for Lower Bounding Adaptive Dual Greedy

Lemma 5.1 *The optimal value of LP1 lower bounds the expected cost of an optimal decision tree T for the SSSC instance on g, p , and c .*

Proof Let X be the assignment to the variables x_w in the LP such that $x_w = 1$ if T tests j on both assignments extending w , and $x_w = 0$ otherwise. With respect to X , the expected cost of T equals $\sum_{a \in \{0, 1\}^n} \sum_j c_j x_{a^j} p(a)$. This equals the value of the objective function, because for $a, a' \in \{0, 1\}^n$ differing only in bit j , $p(a) + p(a') = p(a^j)$. Finally, as noted earlier, for any fixed $a \in \{0, 1\}^n$, the subset of constraints involving a is precisely the set of constraints of Wolsey's IP, if we take the utility function to be g_a such that $g_a(S) = g(S, a)$. Since T produces a cover for every a , by Wolsey's property, the constraints of LP1 involving a are satisfied. Thus X is a feasible solution to LP1, and the optimal value of the LP is at most the expected cost of the optimal tree. \square

We present the pseudocode for ADG in Algorithm 1. (In the line assigning a value to j_l , assume that if $E[g_b(x)] = 0$, the expression evaluates to ∞ .) Its main loop is analogous to the main loop in Fujito's Dual Greedy algorithm, except that ADG uses expected increases in utility, instead of known, deterministic increases in utility.

We now analyze Adaptive Dual Greedy. In the LP in Figure 2, there is a constraint for each a, S pair.

$$\begin{array}{ll} \max & \sum_{a \in \{0, 1\}^n} \sum_{S \subseteq N} p(a) (Q - g(S, a)) y_{S,a} \\ \text{s.t.} & \\ & \sum_{S \subseteq N} \sum_{i \in \mathcal{O}} Pr[w_j = i] g_{S, w^{(i)}}(j) y_{S, w^{(i)}} \leq c_j \\ & \qquad \qquad \qquad \forall w, j \text{ s.t. } w \in W, w_j = * \\ & y_{S,a} \geq 0 \qquad \qquad \qquad \forall S \subseteq N, a \in \{0, 1\}^n \end{array}$$

Figure 3: LP2: the Linear Program for Adaptive Dual Greedy

$$\begin{array}{l} b \leftarrow (*, * \dots, *) , y_S \leftarrow 0 \text{ for all } S \subseteq N \\ F^0 = \emptyset, l \leftarrow 0 \\ \text{while } b \text{ is not a solution to SSSC } (g(b) < Q) \text{ do} \\ \quad l \leftarrow l + 1 \\ \quad j_l \leftarrow \arg \min_{j \notin F^{l-1}} \frac{c_j - \sum_{S: y_S \neq 0} (E[g_{S,b}(j)]) y_S}{E[g_b(j)]} \\ \quad y_{F^{l-1}} \leftarrow \frac{c_{j_l} - \sum_{S: y_S \neq 0} (E[g_{S,b}(j_l)]) y_S}{E[g_b(j_l)]} \\ \quad k \leftarrow \text{the state of } j_l \quad // \text{ "test" } j_l \\ \quad F^l \leftarrow F^{l-1} \cup \{j_l\} \quad // F^l \text{ is set of } j \text{ tested so far} \\ \quad b_{j_l} \leftarrow k \\ \text{end while} \\ \text{return } b \end{array}$$

Algorithm 1: Adaptive Dual Greedy

Multiply both sides of such constraints by $p(a)$, to form an equivalent LP. Note that for all $w \in W$, variable x_w appears only in the constraints for pairs a, S where $a = w^{(0)}$ or $a = w^{(1)}$. Since we are assuming the binary case where $\mathcal{O} = \{0, 1\}$, $P[w_j = i]$ will equal either p_j or $1 - p_j$. Take the dual of the result, and divide both sides of each constraint by $p(w)$. Using the fact that $p(w^{(1)})/p(w) = p_j$ and $p(w^{(0)})/p(w) = 1 - p_j$, we get the LP that we call LP2, shown in Figure 3. In LP2 there is one constraint for each w , because $w_j = *$ for exactly one j . The variables in LP2 are $y_{S,a}$, where $S \subseteq N$ and $a \in \{0, 1\}^n$. Let $P[w_j = i]$ denote the probability that the j th bit of w is in state i .

Consider running ADG on an input $a \in \{0, 1\}^n$. Because $g(a) = Q$, ADG is guaranteed to terminate with an output b such that $g(b) = Q$. Let $C(a)$ be the set of items that ADG tests and inserts into the cover it constructs for a . We will sometimes treat $C(a)$ as a sequence of items, ordered by their insertion order. ADG constructs an assignment to the variables y_S (one for each $S \subseteq N$) when it is run on input a .

Consider the assignment Y to the variables $y_{S,a}$ of LP2 such that each $y_{S,a}$ is given the value that ADG assigns to variable y_S when it is run on input a . Let $Y_{S,a}$ denote the assignment Y makes to variable $y_{S,a}$.

We now show that Y is a feasible solution to LP2 and that for each a , and each $j \in C(a)$, Y makes the

constraint for $w = a^j$ tight. For $w \in W$, let $h'_w(y)$ denote the function of the variables $y_{S,a}$ computed in the left hand side of the constraint for w in LP2.

Lemma 5.2 For every $a \in \{0, 1\}^n$, $j \in N$,

- (1) $h'_{a^j}(Y) = c_j$ if $j \in C(a)$, and
- (2) $h'_{a^j}(Y) \leq c_j$ if $j \notin C(a)$.

Proof Assignment Y assigns non-zero values only to variables $y_{S,a}$ where S is a prefix of sequence $C(a)$.

For $t \in N$, let Y^t denote the assignment to the $y_{S,a}$ variables such that $y_{S,a}$ equals the value of variable y_S at the end of iteration t of the loop in ADG, when ADG is run on input a . (If ADG terminates before iteration t , $y_{S,a}$ equals the final value of y_S). Let Y^0 be the all 0's assignment.

We first prove the following claim: For all $t \in N$ and $w \in W$, $h'_w(Y^t) = \sum_{S \subseteq N} E[g_{S,w}(j)]Y_{S,w^{(1)}}^t$.

Let $w \in W$. Consider running ADG on $w^{(0)}$ and $w^{(1)}$. Since ADG corresponds to a decision tree, the Neighbor Property holds. We consider two cases. For the first case, suppose j is never tested on $w^{(0)}$. Then it is never tested on $w^{(1)}$. and $Y_{S,w^{(0)}}^t = Y_{S,w^{(1)}}^t$ for all S, t . Thus $h'_w(Y^t) = \sum_{S \subseteq N} ((p_j g_{S,w^{(1)}}(j)) + (1 - p_j) g_{S,w^{(0)}}(j)) Y_{S,w^{(1)}}^t = \sum_{S \subseteq N} E[g_{S,w}(j)] Y_{S,w^{(1)}}^t$ for all t .

For the second case, suppose that j is tested in iteration \hat{t} on input $w^{(1)}$, and hence on input $w^{(0)}$. For $t \leq \hat{t}$, $Y_{S,w^{(1)}}^t = Y_{S,w^{(0)}}^t$ for all S . This is not true for $t > \hat{t}$. However, in iterations $t > \hat{t}$, j is already part of the cover, so ADG assigns values only to variables y_S where $j \in S$. For such S , $g_{S,w^{(1)}}(j) = 0$. Thus in this case also, $h'_w(Y^t) = \sum_{S \subseteq N} E[g_{S,w}(j)] Y_{S,w^{(1)}}^t$ for all t . This completes the proof of the claim.

It is now easy to show by induction on t that the two properties of the lemma hold for every Y^t , and hence for Y . They hold for Y^0 . Assume they hold for Y^t . Again consider assignments $w^{(1)}$ and $w^{(0)}$. If j was tested on $w^{(1)}$ and $w^{(0)}$ in some iteration $\hat{t} < t + 1$, then $h'_{a^j}(Y^t) = h'_{a^j}(Y^{\hat{t}+1})$ by the arguments above. If j is tested in iteration $t + 1$ on both inputs, then the value assigned to $y_{F^{t-1}}$ by ADG on $w^{(1)}$ (and $w^{(0)}$) equals $(c_j - h'_w(Y^t)) / E[g_{F^{t-1},w}(j)]$, and thus $h'_w(Y^{t+1}) = c_j$. If j is not tested in iteration $t + 1$, and was not tested earlier, the inductive assumption and the greedy choice criterion ensure that $h'_w(Y^{t+1}) \leq c_j$. \square

The expected cost of the cover produced by ADG on a random input a is $\sum_{a \in \{0,1\}^n} \sum_{j \in C(a)} p(a) c_j$. The next lemma relates this value to an expression involving the $Y_{S,a}$.

Lemma 5.3 $\sum_{a \in \{0,1\}^n} \sum_{j \in C(a)} p(a) c_j$
 $\sum_{a \in \{0,1\}^n} \sum_{S \subseteq N} \sum_{j: j \in C(a)} p(a) g_{S,a}(j) Y_{S,a}$

Proof For $j \in N$, let $W^j = \{w \in W | w_j = *\}$. Then

$$\begin{aligned} & \sum_a \sum_{j: j \in C(a)} p(a) c_j \\ &= \sum_j \sum_{a: j \in C(a)} p(a) c_j \\ & \quad \text{switching the order of summation} \\ &= \sum_j \sum_{i \in \mathcal{O}} \sum_{w \in W_j: j \in C(w^{(i)})} p(w^{(i)}) c_j \\ & \quad \text{grouping assignments by the value of bit } j \\ &= \sum_j \sum_{i \in \mathcal{O}} \sum_{w \in W_j: j \in C(w^{(0)})} (p(w^{(i)}) c_j) \\ & \quad \text{by the Neighbor Property, } j \in C(w^{(1)}) \text{ iff } j \in C(w^{(0)}) \\ &= \sum_j \sum_{w \in W_j: j \in C(w^{(0)})} p(w) c_j \\ &= \sum_j \sum_{w \in W_j: j \in C(w^{(0)})} p(w) h'_w(Y) \\ & \quad \text{by Lemma 5.2} \\ &= \sum_j \sum_{i \in \mathcal{O}} \sum_{w \in W_j: j \in C(w^{(i)})} \sum_S p(w^{(i)}) g_{S,w^{(i)}}(j) Y_{S,w^{(i)}} \\ & \quad \text{by the definition of } h'_w \\ &= \sum_j \sum_{i \in \mathcal{O}} \sum_{w \in W_j: j \in C(w^{(i)})} \sum_S p(w^{(i)}) g_{S,w^{(i)}}(j) Y_{S,w^{(i)}} \\ & \quad \text{because } j \in C(w^{(1)}) \text{ iff } j \in C(w^{(0)}) \\ &= \sum_j \sum_{a: j \in C(a)} \sum_S p(a) g_{S,a}(j) Y_{S,a} \\ &= \sum_a \sum_S \sum_{j: j \in C(a)} p(a) g_{S,a}(j) Y_{S,a} \end{aligned}$$

\square

We now give our approximation bound for ADG.

Theorem 5.1 Given an instance of SSSC with utility function g and goal value Q , ADG constructs a cover whose expected cost is no more than a factor of α larger than the expected cost of the cover produced by the optimal strategy, where $\alpha = \max_{\frac{\sum_{j \in C(a)} g_{S,a}(j)}{Q - g(S,a)}}$, with the max taken over all pairs S, a where $a \in \{0, 1\}^n$, S is a prefix of the cover $C(a)$ that ADG constructs on input a , and $Q - g(S, a) \neq 0$.

Proof By Lemma 5.3, the expected cost of the cover constructed by ADG is $\sum_a \sum_S \sum_{j: j \in C(a)} p(a) g_{S,a}(j) Y_{S,a}$. The value of the objective function of LP2 on Y is $\sum_a \sum_S p(a) (Q - g(S, a)) Y_{S,a}$. For any a , $Y_{S,a}$ is non-zero if S is a prefix of the cover that ADG constructs on input a . Comparing the coefficients of $Y_{S,a}$ in these two expressions implies that the expected cost of the cover is at most $\max_{\frac{\sum_{j \in C(a)} g_{S,a}(j)}{Q - g(S,a)}}$ times the value of the objective function on Y . The theorem follows by Lemma 5.1 and weak duality. \square

6 Algorithms for linear threshold evaluation and Stochastic Min-Knapsack

A linear threshold formula with integer coefficients has the form $\sum_{i=1}^n a_i x_i \geq \theta$ where the a_i and θ are integers. It represents the function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ such that $f(x) = 1$ if $\sum_{i=1}^n a_i x_i \geq \theta$, and $f(x) = 0$ otherwise. We show how to use the Q -value approach to obtain an algorithm solving the SBFE problem for linear threshold formulas with integer coefficients.

Our results on Adaptive Dual Greedy yield a 3-approximation for linear threshold evaluation.

Theorem 6.1 *There is a polynomial-time 3-approximation algorithm solving the SBFE problem for linear threshold formulas with integer coefficients.*

Proof Let $\sum_{i=1}^n a_i x_i \geq \theta$ be the linear threshold formula. Let $h(x) = (\sum_{i=1}^n a_i x_i) - \theta$. For $b \in \{0, 1, *\}$, let $\min(b)$ be the minimum value of $h(b')$ for any extension b' of b . Similarly, let $\max(b)$ be the maximum value of $h(b')$ for any extension b' of b . Thus $\min(b) = (\sum_{j:b_j \neq * a_j b_j}) + (\sum_{i:a_i < 0, b_i = * a_i}) - \theta$, $\max(b) = (\sum_{j:b_j \neq * a_j b_j}) + (\sum_{i:a_i > 0, b_i = * a_i}) - \theta$, and each can be calculated in linear time. Let $R_{\min} = \min(*, \dots, *)$ and $R_{\max} = \max(*, \dots, *)$. If $R_{\min} \geq 0$ or $R_{\max} < 0$, f is constant and no testing is needed. Suppose this is not the case.

Let $Q_1 = -R_{\min}$ and let submodular utility function g_1 be such that $g_1(b) = \min\{-R_{\min}, \min(b) - R_{\min}\}$. Intuitively, $Q_1 - g_1(b)$ is length of the interval containing all values of h that can be induced by extensions b' of b such that $f(b') = 0$. Similarly, define $g_0(b) = \min\{R_{\max} + 1, R_{\max} - \max(b)\}$ and $Q_0 = R_{\max} + 1$. Thus b is a 1-certificate of f iff $g_1(b) = Q_1$, and a 0-certificate iff $g_0(b) = Q_0$.

We apply the disjunctive construction of Lemma 4.1 to construct $g(b) = Q - (Q_1 - g_1(b))(Q_0 - g_0(b))$, which is an assignment feasible utility function for f with goal value $Q = Q_1 Q_0$. We then run ADG using the utility functions and goal value defined above. By Theorem 5.1, the resulting algorithm is within a factor of α of optimal. We now show that $\alpha \leq 3$ in this case.

Fix x and consider the run of ADG on x . Let T be the number of loop iterations. So $C(x) = j_1, \dots, j_T$ is the sequence of tested items, and $F^t = \{j_1, \dots, j_t\}$. Assume first that $f(x) = 1$. Let $F = F^0 = \emptyset$, and consider the ratio $\frac{\sum_{j \in C(x)} g_{F,x}(j)}{Q - g(F,x)}$.

Assume without loss of generality that neither g_0 nor g_1 is identically 0.

Let $A = -R_{\min}$ and let $B = R_{\max} + 1$. Thus $Q - g(\emptyset, x) = AB$. Let C_1 be the set of items j_i in $C(x)$ such that either $x_{j_i} = 1$ and $a_{j_i} \geq 0$ or $x_{j_i} = 0$ and $a_{j_i} < 0$. Similarly, let C_0 be the set of items j_i in $C(x)$, such that either $x_{j_i} = 0$ and $a_{j_i} \geq 0$ or $x_{j_i} = 1$ and $a_{j_i} < 0$.

Testing stops as soon as the goal utility is reached. Since $f(x) = 1$, this means testing on x stops when b satisfies $g_1(b) = Q_1$, or equivalently, b is a 1-certificate of f . Thus the last tested item, j_T , is in C_1 . Further, the sum of the $a_{j_i} x_{j_i}$ over all $j_i \in C_1(x)$, excluding j_T , is less than $-R_{\min}$, while the sum including j_T is greater than or equal to $-R_{\min}$. By the definition of utility function g , $\sum_{j_i \in C_1: j_i \neq j_T} g_{\emptyset, x}(j_i) < AB$. The maximum possible value of $g_{\emptyset, x}(j_T)$ is AB . Therefore, $\sum_{j_i \in C_1} g_{\emptyset, x}(j_i) < 2AB$.

Since x does not contain both a 0-certificate and a 1-certificate of f , the sum of the $a_{j_i} x_{j_i}$ over all $j_i \in C_0(x)$ is strictly less than R_{\max} . Thus by the definition of g , $\sum_{j_i \in C_0} g_{\emptyset, x}(j_i) < AB$. Summing over all $j_i \in C(x)$, we get that $\sum_{j_i \in C(x)} g_{\emptyset, x}(j_i) < 3AB$. Therefore, $\frac{\sum_{j \in C(x)} g_{F,x}(j)}{Q - g(F,x)} < 3$, because for $F = \emptyset$, $Q = AB$ and $g(\emptyset, x) = 0$. A symmetric argument holds when $f(x) = 0$.

It remains to show that the same bound holds when $F \neq \emptyset$. We reduce this to the case $F = \emptyset$. Once we have tested the variables in F^t , we have an induced linear threshold evaluation problem on the remaining variables (replacing the tested variables by their values). Let g' and Q' be the utility function and goal value for the induced problem. The ratio $\frac{\sum_{j \in C(x)} g_{F,x}(j)}{Q - g(F,x)}$ is equal to $\frac{\sum_{j \in C(x)-F} g'_{\emptyset, x'}(j)}{Q' - g'(F, x')}$, where x' is x restricted to the elements not in F . By the argument above, this ratio is bounded by 3. \square

We can use essentially the same approach to obtain results for Stochastic Min-Knapsack.

Theorem 6.2 *There is a polynomial-time 3-approximation algorithm solving the Stochastic Min-Knapsack problem. For the special case of Stochastic Min-Knapsack where threshold θ can always be reached or exceeded, there is a polynomial-time 2-approximation.*

Proof Given a Stochastic Min-Knapsack instance, we can associate with it the linear threshold formula $x_1 + x_2 + \dots + x_n \geq \theta$, over items $\{1, \dots, n\}$, where each item i has cost c_i , and x_i is s_i , the size of item i . Without loss of generality, we can assume that the size s_i of every item is at most θ .

Placing items in the knapsack until the sum of the sizes of the chosen items is at least θ is like testing the values of the variables in this linear threshold formula until the formula evaluates to θ . We would therefore like to apply the proof of the 3-approximation bound for linear threshold formulas to Stochastic Min-Knapsack. However, we need to modify the proof to account for two main differences between the threshold evaluation problem and Stochastic Min-Knapsack.

The first is how the two problems deal with the situation where the threshold value cannot be attained. In

linear threshold evaluation, we can stop testing as soon as we have a 0-certificate, but in Min-Knapsack, we must pay to put all the items in the knapsack if the threshold is not reached. To account for this difference, we modify the utility function. Let $g_1(b) = \min(\sum_{i:b_i \neq *})_{b_i, \theta}$ with goal $Q_1 = \theta$, and $g_0(b) = |\{i : b_i \neq *\}|$ with goal $Q_0 = n$. Combining these with the disjunctive construction to get utility function g with goal value $Q = n\theta$. We can solve the Stochastic Min-Knapsack problem by solving the SSSC problem for g and Q .

The other main difference between the two problems is that for linear threshold evaluation, we considered only Bernoulli distributions on the variables x_i (with integer coefficients on the x_i). In Stochastic Min-Knapsack, the s_i may have arbitrary values in the interval $[0, \theta]$. Since we have assumed that each s_i has only a finite number of possible values, we just have a k -ary SSSC problem for some k , rather than a binary one, and the α bound on ADG holds.

We show that $\alpha \leq 3$ when ADG is run on this g . As in the proof for linear threshold, it suffices to show that $\sum_{j=1}^k g_{\theta, s}(j)/(Q - g(s, \emptyset)) \leq 3$ for all possible size vectors s .

We first consider the case where the threshold is achieved by s , that is, $\sum_{j=1}^n s_j \geq \theta$. Suppose we run ADG on s , and without loss of generality we test items $1, \dots, k$, in that order. For $j \in \{1, \dots, k\}$, the contribution to g_0 for the j th item (if j is added by itself to \emptyset) is 1, and the contribution to g_1 is s_j . Thus for each $j \in \{1, \dots, k\}$, $g_{\theta, s}(j) = 1 + s_j(n - 1)$. Further, $\sum_{j=1}^{k-1} s_j < \theta$, and since each $s_i \leq \theta$ by assumption, $\sum_{j=1}^k g_{\theta, s}(j) \leq \theta(k + 2(n - 1)) < 3n\theta$, and therefore $\sum_{j=1}^k g_{\theta, s}(j)/(Q - g(s, \emptyset)) \leq 3$.

We now consider the case where the threshold is not achieved on s . In this case, $\sum_{j=1}^k s_j < \theta$, and $\sum_{j=1}^k g_{\theta, s}(j) \leq (k + (n - 1))\theta < 2n\theta$. Therefore, $\sum_{j=1}^k g_{\theta, s}(j)/(Q - g(s, \emptyset)) \leq 2$. This inequality also holds if we replace \emptyset by any prefix of $\{1, \dots, k\}$. Thus $\alpha < 3$, and we have a 3-approximation algorithm.

In the special case of the Stochastic Min-Knapsack where the distributions on the sizes are such that we are guaranteed to reach the threshold of θ , we can take utility function g to be just the g_1 defined above. It is easy to show that $\alpha \leq 2$ for this g , and we have a 2-approximation algorithm. \square

7 A new bound for Adaptive Greedy

We give a new analysis of the Adaptive Greedy algorithm of Golovin and Krause, whose pseudocode we present in Algorithm 2.

Some of the variables used in the pseudocode are not necessary for the running of the algorithm, but are useful in its analysis. (In the line assigning a value to j_l , assume

that if $E[g_b(j)] = 0$, the expression evaluates to ∞ .)

```

b  $\leftarrow$   $(*, *, \dots, *)$ 
l  $\leftarrow$  0,  $F^0 \leftarrow \emptyset$ 
while  $b$  is not a solution to SSSC ( $f(b) < Q$ ) do
    l  $\leftarrow$   $l + 1$ 
     $j_l \leftarrow \arg \min_{j \notin F^{l-1}} \frac{c_j}{E[g_b(j)]}$ 
     $k \leftarrow$  the state of  $j_l$  //“test”  $j_l$ 
     $F^l \leftarrow F^{l-1} \cup \{j_l\}$  // $F^l$  is set of  $j$  tested so far
     $b_{j_l} \leftarrow k$ 
end while
return  $b$ 

```

Algorithm 2: Adaptive Greedy

Throughout this section, we let $g(j) = \max_{l \in \{0, 1\}} g_r(j, l)$ where $r = (*, \dots, *)$. Thus $g(j)$ is the maximum increase in utility that can be obtained as a result of testing j (since g is submodular). We show that the expected cost of the solution computed by Adaptive Greedy is within a factor of $2(\ln(\max_{i \in N} g(i) + 1))$ of optimal in the binary case.

In the k -ary case, the 2 in the bound is replaced by k . Note that $\max_j g(j)$ is clearly upper bounded by Q , and in some instances may be much less than Q . However, because of the factor of k at the front of our bound, we cannot say that it is strictly better than the $(\ln Q + 1)$ bound of Golovin and Krause. (The bound that is analogous to ours in the non-adaptive case, proved by Wolsey, does not have a factor of 2.)

Adaptive Greedy is a natural extension of the Greedy algorithm for (deterministic) submodular set cover of Wolsey. We will extend Wolsey’s analysis [44], as it was presented by Fujito [20]. In our analysis, we will refer to LP2 defined in Section 5, along with the associated notation for the constraints $h'_{aj}(y) \leq c_j$.

For $x \in \{0, 1\}^n$, let T^x be the number of iterations of the Adaptive Greedy while loop on input x . Let b_x^t denote the value of b at the end of iteration t of the while loop on input x , and let F_x^t denote the value of F^t , where F^t is the set of j s tested by the end of the $t + 1$ st iteration. (The x in the notation may be dropped when it is understood implicitly.) Set $\theta_x^t = \min_{j \notin F^{t-1}} \frac{c_j}{E[g_{b^{t-1}}(j)]}$.

For $j \in N$, let k_j be the value of t that maximizes $(\theta_x^t)(g_{b_x^t}(j, 1))$. Similarly, let l_j be the value of t that maximizes $(\theta_{x'}^t)(g_{b_{x'}^{t-1}}(j, 0))$, where x' is the assignment obtained from x by complementing x_j . Again, let r denote the assignment $\{*, \dots, *\}$, and let $H_j^1 = H(g_r(j, 1))$ and $H_j^0 = H(g_r(j, 0))$, where $H(n)$ denotes the n th harmonic number, which is at most $(\ln n + 1)$. Let $q_j = 1 - p_j$.

To analyze Adaptive Greedy, we define Y to be the assignment to the LP2 variables $y_{S,x}$ setting $y_{F^0,x} = \theta_x^1$, $y_{F^t,x} = (\theta_x^{t+1} - \theta_x^t)$ for $t \in \{1 \dots T^x - 1\}$ and $y_{S,x} = 0$ for all other S . We define Y^x to be the restriction of that assignment to the variables $y_{S,x}$ for that x . Let $q^x(y) = \sum_{S \subseteq N} (Q - g(S,x))y_{S,x}$.

Lemma 7.1 *The expected cost of the cover constructed by Adaptive Greedy is at most $E[q^x(Y^x)]$, where the expectation is with respect to $x \sim D_p$.*

Proof By the definition of Y^x , the proof follows directly from the analysis of (non-adaptive) Greedy in Theorem 1 of [20], by linearity of expectation. \square

We need to bound the value of $h'_w(Y)$ for each $w \in W$. We will use the following lemma from Wolsey's analysis.

Lemma 7.2 [44] *Given two sequences $(\alpha^{(t)})_{t=1}^T$ and $(\beta^{(t)})_{t=0}^{T-1}$, such that both are nonnegative, the former is monotonically nondecreasing and the latter, monotonically non-increasing, and $\beta^{(t)}$ is a nonnegative integer for any value of t , then*

$$\alpha^{(1)}\beta^{(0)} + (\alpha^{(2)} - \alpha^{(1)})\beta^{(1)} + \dots + (\alpha^{(T)} - \alpha^{(T-1)})\beta^{(T-1)} \leq \left(\max_{1 \leq t \leq T} \alpha^{(t)} \beta^{(t-1)} H(\beta^{(0)}) \right).$$

Lemma 7.3 *For every $x \in \{0, 1\}^n$ and $j \in \{1, \dots, N\}$, $h'_{x,j}(Y) \leq c_j 2H(\max_{i \in N} g(i))$.*

Proof By the submodularity of g , and the greedy choice criterion used by Adaptive Greedy, $\theta_x^1 \leq \theta_x^2 \dots \leq \theta_x^{T^x}$. By the submodularity of g , $g_{b_x^0}(j, 0) \geq g_{b_x^1}(j, 0) \dots \geq g_{b_x^{T^x}}(j, 0)$. Thus Lemma 7.2 applies to the non-decreasing sequence $\theta_x^1, \theta_x^2, \dots, \theta_x^{T^x}$ and the non-increasing sequence $g_{b_x^0}(j, 0), \dots, g_{b_x^1}(j, 0), \dots, g_{b_x^{T^x-1}}(j, 0)$. This also holds if we substitute $(j, 1)$ for $(j, 0)$ in the second sequence.

Let x' be the assignment differing from x only in bit j . In the following displayed equations, we write k and l in place of k_j and l_j to simplify the notation.

$$\begin{aligned} h'_{x,j}(Y) &= \sum_{S \subseteq N} (p_j g_{S,x}(j) + (1 - p_j) g_{S,x'}(j)) Y_{S,x} \\ &\quad \text{by the Neighbor Property} \\ &= p_j [\theta_x^1 g_{b_x^0}(j, 1) + \sum_{i=2}^{T^x} (\theta_x^i - \theta_x^{i-1}) g_{b_x^{i-1}}(j, 1)] + \\ &\quad q_j [\theta_{x'}^1 g_{b_{x'}^0}(j, 0) + \sum_{i=2}^{T^{x'}} (\theta_{x'}^i - \theta_{x'}^{i-1}) g_{b_{x'}^{i-1}}(j, 0)] \end{aligned}$$

$$\begin{aligned} &\leq p_j [\theta_x^k g_{b_x^{k-1}}(j, 1) H_j^1] + q_j [\theta_{x'}^l g_{b_{x'}^{l-1}}(j, 0) H_j^0] \\ &\quad \text{by Lemma 7.2 as indicated above} \\ &\leq p_j [\theta_x^k g_{b_x^{k-1}}(j, 1) H_j^1] + q_j [\theta_x^k g_{b_x^{k-1}}(j, 0) H_j^0] \\ &\quad + p_j [\theta_{x'}^l g_{b_{x'}^{l-1}}(j, 1) H_j^1] + q_j [\theta_{x'}^l g_{b_{x'}^{l-1}}(j, 0) H_j^0] \\ &\quad \text{since this just adds extra non-negative terms} \\ &= \theta_x^k H_j^1 [p_j g_{b_x^{k-1}}(j, 1) + q_j g_{b_x^{k-1}}(j, 0)] \\ &\quad + \theta_{x'}^l H_j^0 [q_j g_{b_{x'}^{l-1}}(j, 0) + p_j g_{b_{x'}^{l-1}}(j, 1)] \\ &\leq c_j H_j^1 + c_j H_j^0 \quad \text{due to the greedy choices of Algorithm 2} \\ &\leq c_j 2H(g(j)) \\ &\leq c_j 2H(\max_i g(i)) \end{aligned}$$

\square

Theorem 7.1 *Given an instance of SSSC with utility function g , Adaptive Greedy constructs a decision tree whose expected cost is no more than a factor of $2(\max_{i \in N} (\ln g(i)) + 1)$ larger than the expected cost of the cover produced by the optimal strategy.*

Proof Let OPT be the expected cost of the cover produced by the optimal strategy. let $AGCOST$ be the expected cost of the cover produced by Adaptive Greedy, and let $q(y)$ denote the objective function of LP2. By Lemma 5.1, the optimal value of LP1 is a lower bound on OPT . By Lemma 7.3, $Z = Y/(2H(\max_i g(i)))$ is a feasible solution to LP2. Thus by weak duality, $q(Z) \leq OPT$. By Lemma 7.1, $AGCOST \leq E[q^x(Y^x)]$, and it is easy to see that $E[q^x(Y^x)] = q(Y)$. Since $q(Y) = q(Z)(2H(\max_i g(i)))$, $AGCOST \leq OPT(2H(\max_i g(i)))$. \square

8 Simultaneous Evaluation and Ranking

Let f_1, \dots, f_m be (representations of) Boolean functions from a class C , such that each $f_i : \{0, 1\}^n \rightarrow \{0, 1\}$. We consider the generalization of the SBFE problem where instead of determining the value of a single function f on an input x , we need to determine the value of all m functions f_i on the same input x .

The Q -value approach can be easily extended to this problem by constructing utility functions for each of the f_i , and combining them using the conjunctive construction in Lemma 4.1. The algorithm of Golovin and Krause for simultaneous evaluation of OR formulas follows this approach [22] (Liu et al. presented a similar algorithm earlier, using a different analysis [35].) We can also modify the approach by calculating a bound based on P -value, or using ADG instead of Adaptive Greedy. We thus obtain the following theorem, where $\sum_{i=1}^n a_{k,i} x_i \leq \theta_k$ is the k th threshold formula.

Theorem 8.1 *There is a polynomial-time algorithm for solving the simultaneous evaluation of linear threshold*

formulas problem which produces a solution that is within a factor of $O(\log m D_{avg})$ of optimal where D_{avg} is the average, over $k \in \{1, \dots, m\}$, of $\sum_{i=1}^n |a_{ki}|$. In the special case of OR formulas, where each variable appears in at most r of them, the algorithm achieves an approximation factor of $2(\ln(\beta_{max} r) + 1)$, where β_{max} is the maximum number of variables in any of the OR formulas.

There is also a polynomial-time algorithm for solving the simultaneous evaluation of threshold formulas problem which produces a solution that is within a factor of D_{max} of optimal, where $D_{max} = \max_{k \in \{1, \dots, m\}} \sum_{i=1}^n |a_{ki}|$.

Proof We use the utility function construction for linear threshold functions from the beginning of the proof of Lemma 6.1. Let $g^{(1)}, \dots, g^{(m)}$ be the m utility functions that result from applying this construction to the m linear threshold formulas that need to be evaluated. Let $Q^{(1)}, \dots, Q^{(m)}$ be the associated goal values.

Using the conjunctive construction, we construct utility function g such that $g(b) = \sum_{k=1}^m g^{(k)}(b)$, and $Q = \sum_{k=1}^m Q^{(k)}$.

To obtain the first algorithm, we evaluate all the threshold formulas by running Adaptive Greedy with g , goal value Q , and the given p , and c , until it outputs a cover b . Given cover b , it is easy to determine the value of $f_k(x)$ for each f_k .

For each f_k , the associated Q_k is $O(D_k)$, where D_k is the sum of the absolute values of the coefficients in f_k . Since $Q = \sum_k Q_k$, the $O(\log(m D_{avg}))$ bound follows from the $(\ln Q + 1)$ bound for Adaptive Greedy.

Suppose each threshold formula is an OR formula. For $b \in \{0, 1, *\}^n$, $\max_{l \in \{0, 1\}} g_b^{(k)}(i, l) = 0$ if x_j does not appear in the k th OR formula, otherwise it is equal to the number of variables in that formula. The $2(\ln(\beta_{max} r) + 1)$ approximation factor then follows by our bound on Adaptive Greedy in Theorem 7.1.

For the second algorithm, we just use ADG instead of Adaptive Greedy with the same utility function g . By Theorem 5.1, the approximation factor achieved by ADG is $\max \frac{\sum_j g_{S,x}^{(j)}}{Q - g(S,x)}$.

We bound this ratio for g . Let $D_j = \sum_{i=1}^n |a_{ji}|$. Let $d \in \{0, 1\}^n$ and $S \in F(x)$. Without loss of generality, assume $S = \{n' + 1, \dots, n\}$. In the k th threshold formula, for $i \geq n'$, replace x_i with d_i . This induces a new threshold formula on $n - n'$ variables with threshold $\theta_{k,d} = \theta_k - D_{k,d}$ whose coefficients sum to $D_{k,b} = D_k - \sum_{i=n'+1}^n a_i d_i$. Let b be the partial assignment such that $b_i = d_i$ for $i \geq n'$, and $b_i = *$ otherwise. If b contains either a 0-certificate or a 1-certificate for f_k , then $Q_k - g^k(S, d) = 0$.

Otherwise, $Q_k - g(S, d) = (\theta_{k,b})(D_{k,b} - \theta_{k,b} + 1)$, and $\sum_j g_{S,d}^k(j) \leq D_{k,b} \max\{\theta, D_{k,b} - \theta_{k,b} + 1\}$. It

follows that $\frac{\sum_j g_{S,d}^k(j)}{Q_k - g^k(S, d)} \leq D_{k,b} \leq D_{max}$.

Since this holds for each k , $\max \frac{\sum_j g_{S,d}^{(j)}}{Q - g(S, d)} \leq D_{max}$. \square

For the special case of simultaneous evaluation of OR formulas, the theorem implies a β -approximation algorithm, where β is the length of the largest OR formula. This improves the 2β -approximation achieved by the randomized algorithm of Liu et al. [35].

We use a similar approach to solve the *Linear Function Ranking* problem. In this problem, you are given a system of linear functions f_1, \dots, f_m , where for $j \in \{1, \dots, m\}$, f_j is $a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n$, and the coefficients a_{ji} are integers. You would like to determine the sorted order of the values $f_1(x), \dots, f_m(x)$, for an initially unknown $x \in \{0, 1\}^n$. (Note that the values of the $f_j(x)$ are not Boolean.) We consider the problem of finding an optimal testing strategy for this problem, where as usual, $x \sim D_p$, for some probability vector p , and there is a cost vector c specifying the cost of testing each variable x_i .

Note that there may be more than one correct output for this problem if there are ties. So, strictly speaking, this is not a function evaluation problem. Nevertheless, we can still exploit our previous techniques. For each system of linear equations f_1, \dots, f_m over x_1, \dots, x_n , and each $x \in \{0, 1\}^n$, let $f(x)$ denote the set of permutations $\{f_{j_1}, f_{j_2}, \dots, f_{j_m}\}$ of f_1, \dots, f_m such that $f_{j_1}(x) \leq f_{j_2}(x) \leq \dots \leq f_{j_m}(x)$. The goal of sorting the f_j is to output *some* permutation that we know definitively to be in $f(x)$. Note that in particular, if e.g., $f_i(x) < f_j(x)$, it may be enough for us to determine that $f_i(x) \leq f_j(x)$.

Theorem 8.2 *There is an algorithm that solves the Linear Function Ranking problem that runs in time polynomial in m, n , and D_{max} , and achieves an approximation factor that is within $O(\log(m D_{max}))$ of optimal, where D_{max} is the maximum value of $\sum_{i=1}^n |a_{ji}|$ over all the functions f_j .*

Proof For each pair of linear equations f_i and f_j in the system, where $i < j$, let f_{ij} denote the linear function $f_i - f_j$. We construct a utility function $g^{(ij)}$ with goal value $Q^{(ij)}$. Intuitively, the goal value of $g^{(ij)}$ is reached when there is enough information to determine that $f_{ij}(x) \geq 0$, or when there is enough information to determine that $f_{ij}(x) \leq 0$.

For each i, j pair, let $\min_{ij}(b)$ be the minimum value of $f_{ij}(b')$ on any assignment $b' \in \{0, 1\}^n$ such that $b' \sim b$, and let $\max_{ij}(b)$ be the maximum value. Let $R_{max}(ij) = \max_{ij}(*, \dots, *)$ and let $R_{min}(ij) = \min_{ij}(*, \dots, *)$.

Let $g_{<}^{(ij)} : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$, be defined as follows. If $R_{max}(ij) \leq 0$, then $g_{<}^{(ij)}(b) = 0$ for all $b \in \{0, 1, *\}^n$ and $Q_{<}^{(ij)} = 0$. Otherwise, for $b \in \{0, 1, *\}^n$, let $g_{<}^{(ij)}(b) = \min\{R_{max}(ij), R_{max}(ij) - \max_{ij}(b)\}$ and $Q_{<}^{(ij)} =$

$R_{max(ij)}$. It follows that for $b \in \{0, 1, *\}^n$, $f_i(b') \leq f_j(b')$ for all extensions $b' \sim b$ iff $g_{<}^{(ij)}(b) = Q_{<}^{(ij)}$.

We define $g_{>}^{(ij)}$ and $Q_{<}^{(ij)}$ symmetrically, so that $f_i(b') \geq f_j(b')$ for all extensions $b' \sim b$ iff $g_{>}^{(ij)}(b) = Q_{>}^{(ij)}$.

We apply the disjunctive construction of Lemma 4.1 to combine $g_{>}^{(ij)}$ and $g_{<}^{(ij)}$ and their associated goal values. Let the resulting new utility function be $g^{(ij)}$ and let its goal value be $Q^{(ij)}$. As in the analysis of the algorithm in Section 6, we can show that $Q^{(ij)}$ is $O(D^2)$, where D is the sum of the magnitudes of the coefficients in f_{ij} .

Using the AND construction of Lemma 4.1 to combine the $g^{(ij)}$ we get our final utility function $g = \sum_{i < j} g^{(ij)}$ with goal value $Q = \sum_{i < j} Q^{(ij)}$.

We now show that achieving the goal utility Q is equivalent to having enough information to do the ranking. Until the goal value is reached, there is still a pair i, j such that it remains possible that $f_i(x) > f_j(x)$ (under one setting of the untested variables), and it remains possible that $f_j(x) < f_i(x)$ (under another setting). In this situation, we do not have enough information to output a ranking we know to be valid.

Once $g(b) = Q$, the situation changes. For each i, j such that $f_i(x) < f_j(x)$, we know that $f_i(x) \leq f_j(x)$. Similarly, if $f_i(x) > f_j(x)$, then at goal utility Q , we know that $f_i(x) \geq f_j(x)$. If $f_i(x) = f_j(x)$ at goal utility Q , we may only know that $f_i(x) \geq f_j(x)$ or that $f_i(x) \leq f_j(x)$. We build a valid ranking from this knowledge as follows. If there exists an i such that we know that $f_i(x) \leq f_j(x)$ for all $j \neq i$, then we place $f_i(x)$ first in our ranking, and recursively rank the other elements. Otherwise, we can easily find a “directed cycle,” i.e. a sequence i_1, \dots, i_m , $m \geq 2$, such that we know that $f_{i_1}(x) \leq f_{i_2}(x) \leq \dots \leq f_{i_m}(x)$ and $f_{i_m}(x) \leq f_{i_1}(x)$. It follows that $f_{i_1}(x) = \dots = f_{i_m}(x)$. In this case, we can delete f_{i_2}, \dots, f_{i_m} , recursively rank f_{i_1} and the remaining f_i , and then insert f_{i_2}, \dots, f_{i_m} into the ranking next to f_{i_1} .

Applying Adaptive Greedy to solve the SSSC problem for g , the theorem follows from the $(\ln Q + 1)$ approximation bound for Adaptive Greedy, and the fact that $Q = O(D_{max}^2 m^2)$. \square

9 Acknowledgments

Lisa Hellerstein was partially supported by NSF Grants 1217968 and 0917153. Devorah Kletenik was partially supported by NSF Grant 0917153 and by US Department of Education GAANN Grant P200A090157. Amol Deshpande was partially supported by NSF Grants 0916736 and 1218367. We thank Tonguç Ünlüyurt and Sarah Allen for helpful feedback and Sarah Allen for a notation summary. We thank anonymous referees for helpful suggestions, such as applying our techniques to Stochastic Min-Knapsack, and suggesting a way to simplify our

original analysis of ADG. Alina Ene told us about the work of Im et al. and its connection to Stochastic Min-Knapsack. Lisa Hellerstein would like to thank Endre Boros, Kazuhisa Makino, and Vladimir Gurvich for a stimulating discussion at RUTCOR.

References

- [1] M. Adler and B. Heeringa. Approximating optimal binary decision trees. *Algorithmica*, 62(3-4):1112–1121, 2012.
- [2] A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, and T. Tamir. On chromatic sums and distributed resource allocation. *Inf. Comput.*, 140(2):183–202, February 1998.
- [3] G. Bellala, S. Bhavnani, and C. Scott. Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Trans. on Information Theory*, 2012.
- [4] Y. Ben-Dov. Optimal testing procedure for special structures of coherent systems. *Management Science*, 1981.
- [5] P. Beraldi and A. Ruszczyński. The probabilistic set-covering problem. *Operations Research*, 50(6):956–967, 2002.
- [6] A. Bhalgat. A $(2 + \epsilon)$ -approximation algorithm for the stochastic knapsack problem. 2011. Unpublished Manuscript.
- [7] A. Bhalgat, A. Goel, and S. Khanna. Improved approximation results for stochastic knapsack problems. In *SODA*, 2011.
- [8] E. Boros and T. Ünlüyurt. Diagnosing double regular systems. *Annals of Mathematics and Artificial Intelligence*, 26(1-4):171–191, September 1999.
- [9] E. Boros and T. Ünlüyurt. Sequential testing of series-parallel systems of small depth. *Computing Tools for Modeling, Optimization and Simulation*, pages 39–74, 2000.
- [10] R. Carr, L. Fleischer, V. Leung, and C. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *SODA*, 2000.
- [11] M.-F. Chang, W. Shi, and W. K. Fuchs. Optimal diagnosis procedures for k-out-of-n structures. *IEEE Trans. Comput.*, 39(4), April 1990.
- [12] M. Charikar, R. Fagin, V. Guruswami, J. M. Kleinberg, P. Raghavan, and A. Sahai. Query strategies for priced information. *J. Comput. Syst. Sci.*, 64(4):785–819, 2002.
- [13] L. Cox, Y. Qiu, and W. Kuehner. Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations Research*, 21:1–29, 1989.
- [14] B. Dean, M. Goemans, and J. Vondrak. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *FOCS*, 2004.
- [15] C. Derman, C. Lieberman, and S. Ross. Stochastic optimization is (almost) as easy as deterministic optimization. *Management Science*, 24(5):554–561, 1978.
- [16] A. Deshpande and L. Hellerstein. Flow algorithms for parallel query optimization. In *ICDE*, 2008.
- [17] U. Feige, L. Lovász, and P. Tetali. Approximating minimum set cover. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, APPROX '02, pages 94–107, London, UK, 2002. Springer-Verlag.

- [18] A. Fiat and D. Pechyony. Decision trees: More theoretical justification for practical algorithms. In *ALT*, 2004.
- [19] T. Fujito. On approximation of the submodular set cover problem. *Operations Research Letters*, 1999.
- [20] T. Fujito. Approximation algorithms for submodular set cover with applications. *IEICE Trans. Inf. Syst.*, 83, 2000.
- [21] M. Garey. Optimal task scheduling with precedence constraints. *Discrete Mathematics*, 4:37–56, 1973.
- [22] D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res.*, 2011.
- [23] D. Golovin, A. Krause, and D. Ray. Near-optimal Bayesian active learning with noisy observations. In *NIPS*, pages 766–774, 2010.
- [24] R. Greiner, R. Hayward, M. Jankowska, and M. Molloy. Finding optimal satisficing strategies for and-or trees. *Artif. Intell.*, 170(1):19–58, 2006.
- [25] R. Greiner, R. Hayward, and M. Molloy. Optimal depth-first strategies for and-or trees. In *AAAI/IAAI*, pages 725–730, 2002.
- [26] D. Guijarro, V. Lavín, and V. Raghavan. Exact learning when irrelevant variables abound. In *EuroCOLT*, 1999.
- [27] A. Guillory and J. Bilmes. Simultaneous learning and covering with adversarial noise. In *ICML*, 2011.
- [28] X. Han and K. Makino. Online minimization knapsack problem. In E. Bampis and K. Jansen, editors, *Approximation and Online Algorithms, 7th International Workshop, WAOA 2009, Copenhagen, Denmark, September 10–11, 2009. Revised Papers*, volume 5893 of *Lecture Notes in Computer Science*, pages 182–193. Springer, 2010.
- [29] J. Hastad. On the size of weights for threshold gates. *SIAM Journal on Discrete Mathematics*, 1994.
- [30] T. Ibaraki and T. Kameda. On the optimal nesting order for computing n-relational joins. *ACM Trans. Database Syst.*, 9(3):482–502, 1984.
- [31] S. Im, V. Nagarajan, and R. van der Zwaan. Minimum latency submodular cover. In *ICALP (1)*, pages 485–497, 2012.
- [32] S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In *FOCS*, 2009.
- [33] H. Kaplan, E. Kushilevitz, and Y. Mansour. Learning with attribute costs. In *STOC*, pages 356–365, 2005.
- [34] R. Krishnamurthy, H. Boral, and C. Zaniolo. Optimization of nonrecursive queries. In *VLDB*, 1986.
- [35] Z. Liu, S. Parthasarathy, A. Ranganathan, and H. Yang. Near-optimal algorithms for shared filter evaluation in data stream systems. In *SIGMOD*, 2008.
- [36] M. Moshkov. Approximate algorithm for minimization of decision tree depth. In G. Wang, Q. Liu, Y. Yao, and A. Skowron, editors, *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, volume 2639 of *Lecture Notes in Computer Science*, pages 579–579. Springer Berlin / Heidelberg, 2003.
- [37] M. Moshkov and I. Chikalov. Bounds on average weighted depth of decision trees. *Fundam. Inform.*, 31(2):145–156, 1997.
- [38] K. Munagala, S. Babu, R. Motwani, and J. Widom. The pipelined set cover problem. In *ICDT*, 2005.
- [39] S. Nijssen and E. Fromont. Mining optimal decision trees from itemset lattices. In *KDD*, 2007.
- [40] S. Salloum. *Optimal testing algorithms for symmetric coherent systems*. PhD thesis, University of Southern California, 1979.
- [41] S. Salloum and M. Breuer. An optimum testing algorithm for some symmetric coherent systems. *Journal of Mathematical Analysis and Applications*, 101(1):170 – 194, 1984.
- [42] U. Srivastava, K. Munagala, J. Widom, and R. Motwani. Query optimization over web services. In *VLDB*, 2006.
- [43] T. Ünlüyurt. Sequential testing of complex systems: a review. *Discrete Applied Mathematics*, 142(1-3):189–205, 2004.
- [44] L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982. 10.1007/BF02579435.

A Table of notation

x_i	the i th variable
p_i	probability that variable x_i is 1
c_i	cost of testing x_i
p	the probability product vector (p_1, p_2, \dots, p_n)
c	the cost vector (c_1, c_2, \dots, c_n)
b	a partial assignment, an element of $\{0, 1, *\}^n$
$a \sim b$	a extends b (is identical to b for all variables i such that $b_i \neq *$)
D_p	product distribution, defined by p
$x \sim D_p$	a random x drawn from distribution D_p
Q	goal utility
P	maximum utility that testing a single variable x_i can contribute
g	utility function defined on partial assignments with a value in $\{0, \dots, Q\}$
N	the set $\{1, \dots, n\}$
S	a subset of N
$g(S, b)$	utility of testing only the items in S , with outcomes specified by b
$g_{S,b}(j)$	$g(S \cup \{j\}, b) - g(S, b)$
$b_{x_i \leftarrow l}$	b extended by testing variable i with outcome l
W	the set of partial assignments that contain exactly one $*$
$w^{(0)}, w^{(1)}$	for $w \in W$, the extensions obtained from w by setting the $*$ to 0 and 1, respectively
$j(w)$	for $w \in W$, the j for which $w_j = *$
a^j	the partial assignment produced from a by setting the j th bit to $*$ for assignment a
a'	the assignment produced from a by complementing the j th bit
$g_a(S)$	$g(S, a)$
$y_{S,a}$	the variable in LP2 for SSSC associated with subset S and assignment a
$C(a)$	the sequence of items tested by ADG on assignment a , in order of testing
$Y_{S,a}$	the value of ADG variable y_S after running ADG on input a
$h'_w(y)$	the left hand side of the constraint in LP2 for w (a function of the $y_{S,a}$ variables)
Y^t	assignment to the $y_{S,a}$ variables s.t. $y_{S,a}$ is the value of ADG variable y_S at the end of iteration t of its while loop, when ADG is run on input a
F^t	variable of ADG, the set containing the first t variables it tests
T^x	the number of iterations of the Adaptive Greedy (AG) while loop on input x
b_x^t	the value of b on input x after the t th iteration of the loop of AG on x
Y^x	the assignment to the LP2 variables used in the analysis of the new bound for AG
$q^x(y)$	$\sum_{S \subset N} (Q - g(S, x)) y_{S,x}$
$g_b(i, l)$	equals increase in utility of test on i with outcome l if i was not yet tested (else equals 0), used in analysis of AG
$g(j)$	equals $\max_{l \in \{0,1\}} g_r(j, l)$ where $r = (*, \dots, *)$, used in analysis of AG