A General Framework for Approximating Min Sum Ordering Problems

Felix Happach[*] Lisa Hellerstein[†] Thomas Lidbetter[‡]

July 23, 2020

Abstract

We consider a large family of problems in which an ordering (or, more precisely, a chain of subsets) of a finite set must be chosen to minimize some weighted sum of costs. This family includes variations of Min Sum Set Cover (MSSC), several scheduling and search problems, and problems in Boolean function evaluation. We define a new problem, called the Min Sum Ordering Problem (MSOP) which generalizes all these problems using a cost and a weight function on subsets of a finite set. Assuming a polynomial time α-approximation algorithm for the problem of finding a subset whose ratio of weight to cost is maximal, we show that under very minimal assumptions, there is a polynomial time 4α-approximation algorithm for MSOP. This approximation result generalizes a proof technique used for several distinct problems in the literature. We apply this to obtain a number of new approximation results.

Keywords: scheduling, search theory, Boolean function evaluation, Min Sum Set Cover, approximation algorithms

[*] Department of Mathematics and TUM School of Management, Technische Universitat München, 80333 München, Germany, felix.happach@tum.de

[†] Department of Computer Science and Engineering, New York University Tandon School of Engineering, New York, New York 11201, lisa.hellerstein@nyu.edu

[‡] Department of Management Science and Information Systems, Rutgers Business School, Newark, NJ 07102, tlidbetter@business.rutgers.edu

1 Introduction

Many optimization problems (perhaps the most famous of which being the Traveling Salesman Problem, or TSP) require the elements of a finite set to be arranged in some order to minimize some cost. This work is concerned with ordering problems in which the objective is not to minimize a *single total cost*, as in the TSP, but an *incremental sum of costs* or, equivalently, a weighted average of costs. For example, in many scheduling problems, the objective is to minimize some weighted sum of completion times of a set of jobs. The order in which the jobs are processed may be constrained by some form of precedence constraints. In search problems, one might wish to minimize the expected time or cost incurred in searching for a target or targets that are hidden according to a known probability distribution, and the set of feasible searches may be restricted by some network structure. These problems arise in search and rescue as well as military search operations. Sequential testing problems also come under this framework: a set of tests (for example, medical tests, database queries or quality tests of computer chip components) must be performed in some order to minimize an expected cost (of forming a diagnosis, of determining whether the query is satisfied or of checking whether the component meets certain quality standards).

We unify problems of this type by introducing a new, very general problem formulation, which we call the *Min Sum Ordering Problem* or *MSOP*. Roughly speaking, the problem is to choose an increasing (with respect to set inclusion) sequence of subsets, called a *chain*, of a finite set V so as to minimize some weighted sum of costs, where the weights and costs are given by two non-negative, non-decreasing set functions (a *cost function* and a *weight function*). The elements in the chain must belong to some specified family of subsets of V If this family contains *all* subsets of V, then the problem is equivalent to minimizing over all permutations of V, where the subsets in the chain correspond to the elements picked "so far" by the permutation. By setting the problem up in the more general way, in terms of maximizing over chains rather than permutations, we ensure that the model is general enough to incorporate the intricacies of precedence constraints or restrictions due to a network structure, for example. MSOP is described more precisely in Section 2, where we also discuss in detail some examples of problems that come under its framework.

Similar "min-sum" problems with general cost and/or weight functions have been introduced in Iwata et al. (2012) and Pisaruk (1992) (considered further in Fokkink et al. (2019)). MSOP is more general than both of these problems in two ways. Firstly, we make weaker assumptions about the form of the cost and weight functions: Pisaruk (1992) assumes that the cost function is submodular and the weight function is supermodular, and Iwata et al. (2012) (implicitly) assume the weight function is the cardinality function. Our main results require that either the cost function is subadditive or the weight function is superadditive. Secondly, the aforementioned works take the approach of minimizing over all permutations of V, in contrast to our approach of minimizing over chains.

MSOP is NP-hard, so we consider approximation algorithms in the case that the cost and weight functions take particular forms. An important concept in our analysis is that of the *density* of a subset of V, which is the ratio of its weight to its cost. In forming a chain, we define the *marginal density* of each subset in the chain as the ratio of the increase in weight to the increase in cost. The algorithm relies on a subroutine that solves the subproblem of finding subsets of maximum marginal density, or approximates this subproblem within some factor $\alpha \geq 1$. We consider a greedy algorithm which forms a chain by recursively picking subsets of maximum (or approximately maximum) marginal density. Our main result (Theorem 1) says that the greedy algorithm is a 4α-approximation algorithm for an optimal solution to MSOP for any subadditive cost function and arbitrary weight function (subject to a technical condition on the set of feasible chains). By considering a *dual problem* we also prove an approximation result for the class of problems with a superadditive weight function using a "backward" greedy algorithm (Theorem 13).

The proof of Theorem 1 is inspired by the elegant proof in Feige et al. (2004) of the 4-approximation algorithm for the problem *Min Sum Set Cover* (*MSSC*). This is the problem of ordering a ground set V to minimize the sum of "covering times" of a given collection of subsets of V, where the covering time of a subset is the earliest position in the ordering of any element of that subset. The proof uses the idea of representing the cost of the ordering produced by the greedy algorithm and that of an optimal ordering by two histograms, and showing that when the first histogram is shrunk by a factor of two in the horizontal and vertical directions, it fits in the second histogram. The proof idea is generalized in Streeter and Golovin (2008), who proved a 4-approximation result for a class of problems that includes some special cases of MSOP, including MSSC. A different generalization of MSSC is given in Iwata et al. (2012) to prove the 4-approximation for one case of the *Minimum Linear Ordering Problem*. More recently, a similar proof was used in Hermans et al. (2019) to establish an 8-approximation for the *expanding search* problem, and in Happach and Schulz (2020a) to obtain a 4-approximation for *bipartite OR-scheduling*.

While the last three works cited all use a similar proof method, the proof is somewhat different in each case and none of these results directly implies another. The similarity of the proofs strongly suggests that some deeper result is behind all of these problems. We confirm here that this is indeed the case by showing that MSOP generalizes each of them. In Section 3 we shall show that their respective approximation results are generalized by Theorem 1, which we prove using a variation of the proof originally devised by Feige et al. (2004). The main difference from the original proof stems from the fact that our algorithm does not greedily pick elements of V one by one, but rather greedily picks subsets in the chain. Also, we do not optimize over permutations but over chains.

In Section 4, we will show that Theorem 1 can be applied to additional problems in scheduling theory and in Boolean function evaluation. In particular, we consider scheduling problems with *OR-precedence constraints*, where the set of jobs to be processed are represented by the vertices of a directed acyclic graph (DAG), and a job can only be processed after at least one of its predecessors in the DAG has been processed. We use Theorem 1 to show that there is a polynomial time 4-approximation algorithm for the problem of minimizing the sum of the weighted completion times of a set of jobs that must be scheduled so as to respect some OR-precedence constraints given by a DAG that is in the form of an *inforest* or, more generally, a *multitree* (where inforests and multitrees will be defined in Section 4). We also give a 4-approximation algorithm for a version of MSSC with OR-precedence constraints in the form of an inforest. Finally, we give an 8-approximation algorithm for minimizing the expected cost of non-adaptively evaluating a Boolean read-once formula (AND/OR tree), assuming independent tests. In Section 5 we introduce the dual problem, which leads to further approximation results, and in Section 6 we indicate directions for future work.

2 Problem Definition and Examples

Let V be a finite set of cardinality n and let $f, g : F \to \mathbb{R}$ be a *cost function* and a *weight function*, respectively, defined on some family of subsets $F \subseteq 2^V$ that contains $\emptyset$ and V (where we use the symbol $\subseteq$ to denote "is a subset of or equal to" and $\subset$ to denote strict inclusion). We assume that f and g are given by value oracles. We define an *F-chain* to be a sequence of subsets $S = (S_j)_{j=0}^k$ for some k such that $\emptyset = S_0 \subset S_1 \subset \ldots \subset S_k = V$ and $S_j \in F$ for each j. When there is no ambiguity, we will simply refer to an F-chain as a *chain*.

Then the Min Sum Ordering Problem is to minimize

$$C_{f,g}(S) \equiv \sum_{j=1}^{k} f(S_j)(g(S_j) - g(S_{j-1})), \tag{1}$$

over all chains $S = \{S_j\}_{j=0}^k$. We assume in this paper that f and g are non-decreasing with respect to set inclusion and that $f(\emptyset) = g(\emptyset) = 0$. If S minimizes $C_{f,g}(S)$, we say it is *optimal* and if $C_{f,g}(S)$ is at most a factor $\alpha \geq 1$ times the optimal value of the objective, we say S is an α-approximation. Theorems 1 and 13 are approximation results for MSOP in the cases that f is subadditive and F is closed under union, or g is superadditive and F is closed under intersection, respectively. (The function f is subadditive if and only if $f(S \cup T) \leq f(S) + f(T)$ for all $S, T \in F$

and g is superadditive if and only if $g(S \cup T) \geq g(S) + g(T)$ for all $S, T \in F$. Subadditivity and superadditivity are more general concepts that submodularity and supermodularity, where f is submodular if and only if $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$ for all $S, T \in F$ and f is supermodular if and only if $f(S \cup T) + f(S \cap T) \geq f(S) + f(T)$ for all $S, T \in F$.)

Although it is convenient to define MSOP in terms of minimizing over chains, the applications that we are interested in are problems of minimizing over a set of permutations of V. To be more precise, let $\sigma : V \to V$ be a permutation of V, and for $j = 1, \ldots, n$, let S_j^σ be the union of the first j elements of V under the permutation σ. Then for a set Σ of permutations of V and given functions f and g, we wish to solve

$$\min_{\sigma \in \Sigma} \sum_{j=1}^{n} f(S_j^\sigma)(g(S_j^\sigma) - g(S_{j-1}^\sigma)). \tag{2}$$

We refer to this problem as the *Min Sum Permutation Problem* or *MSPP*. We will show in Section 3 that as long as Σ satisfies some technical condition, MSPP can be regarded as a special case of MSOP. Indeed, let F^Σ consist of all *initial sets* of some $\sigma \in \Sigma$ (that is, all subsets of V that contain the first j elements of some permutation $\sigma \in \Sigma$, for some j). Suppose we have an α-approximate solution S to MSOP with $F = F^\Sigma$. Then any permutation σ such that every subset in S is some initial set of σ will be an α-approximate solution to MSPP. This observation is easily verified and we postpone its justification until Section 3.

The idea of minimizing over chains is novel, but MSPP has been studied previously in some other special cases. In particular, Pisaruk (1992) considered the problem in the case that f is submodular and g is supermodular, giving a 2-approximation algorithm. This special case was also considered more recently in Fokkink et al. (2019), where the problem was introduced as the *submodular search problem*, and previous to that, Fokkink et al. (2017) considered the case of f submodular and g modular.

If g is the cardinality function $g(S) = |S|$ and the sets S_j increase by one element in each step, then the sum in (1) reduces to $\sum_{j=1}^{k} f(S_j)$. This special case of MSPP was considered in Iwata et al. (2012), for various classes of function f. In particular, a 4-approximation algorithm was obtained in the case that f is supermodular. We discuss this in more detail in Subsection 2.2, in particular in reference to MSSC and its generalizations. A more general 4-approximation had already been proved by Streeter and Golovin (2008) in the case of f supermodular and g modular.

We will define a simple greedy algorithm for MSOP, which we now briefly describe (see Section 3 for a more precise description). The algorithm constructs a chain by recursively choosing the $(j+1)$th subset S_{j+1} in the chain in such a way as to maximize the marginal density $(g(S_{j+1}) - g(S_j))/(f(S_{j+1}) - f(S_j))$. We refer to this maximization problem as the *maximum density problem*.

If the maximum density problem cannot be solved in polynomial time, but we can approximate it in polynomial time within a factor $\alpha \geq 1$ then we call a chain produced by such an approximation an *α-greedy* chain. We will prove the following in Section 3.

Theorem 1 Suppose f is subadditive and F is closed under union. Then for any $\alpha \geq 1$, an α-greedy chain is a 4α-approximation for an optimal chain for MSOP.

Later, in Section 5, we will derive an analogous result to Theorem 1 for a "backward" greedy algorithm, using the concept of the *dual problem*.

In the next subsection we present some examples of problems in the fields of search theory, scheduling theory and Boolean function evaluation that fall under the umbrella of MSOP.

2.1 Search Theory and AND-Scheduling

The expanding search problem was introduced in Alpern and Lidbetter (2013), and independently in Averbakh and Pereira (2012) under a different nomenclature. A connected graph $G = (V, E)$ is given, and each edge $e \in E$ has a cost c_e. A target is hidden on one of the vertices of the graph according to a known probability distribution, so that the probability it is on vertex $v \in V$ is p_v. An *expanding search*, starting at some distinguished root r is a sequence of edges $e_1, \cdots, e_{|E|}$ chosen so that r is incident to e_1 and every edge e_i $(i > 1)$ is incident to some previously chosen edge. For a given expanding search, the *expected search cost* of the target is the expected value of the sum of the costs of all the edges chosen up to and including the first edge that contains the target. The problem is to find an expanding search with minimal expected search cost. The problem was shown to be NP-hard in Averbakh and Pereira (2012), and Hermans et al. (2019) recently gave an 8-approximation algorithm.

To express the problem in the form of MSOP, let Σ be the set of expanding searches and take $F = F^{\Sigma}$, as described in the preamble to this section. Then F is closed under union, since its elements consist of connected subgraphs of G containing r. For $S \in F$, let $f(S) = \sum_{e \in S} c_e$ and let $g(S)$ be the sum of p_v over all vertices v contained in some edge of S. Then f is modular, and Theorem 1 along with the results of Subsection 3.2 on MSPP imply that the greedy algorithm is 4α-approximate, where α is the approximation ratio of the maximum density problem, generalizing the analogous Theorem 2 of Hermans et al. (2019).

Alpern and Lidbetter (2013) gave a solution to the expanding search problem in the case that the graph is a tree. In this case, the problem is equivalent to a special case of the single machine, precedence constrained scheduling problem, usually denoted $1|prec|\sum w_j C_j$, of minimizing the sum of the weighted completion times of a set of jobs. A partial order is given on the jobs, and a job becomes available for processing only after all of its predecessors have been processed. We refer to this type of precedence constraints and precedence constrained scheduling as *AND-precedence constraints* and *AND-scheduling*. The jobs have weights w_j and processing times p_j, and for a given ordering, the completion time C_j of a job j is the sum of its processing time and all the processing times of the jobs preceding it. The problem is to find a feasible ordering that minimizes the weighted sum $\sum w_j C_j$ of the completion times. Comparing the weights and the processing times to the probabilities and the edge costs in the expanding search problem, it is easy to see that if the partial order has a tree-like structure, then the scheduling problem and the search problem are equivalent, as pointed out in Fokkink et al. (2019).

A polynomial time algorithm for the scheduling problem $1|prec|\sum w_j C_j$ on trees was given by Horn (1972). Sidney (1975) proved that any optimal schedule (for general precedence constraints) must respect what is now known as a *Sidney decomposition*, obtained by recursively taking subsets of jobs of maximum density. Lawler (1978) gave a polynomial time algorithm for the problem on series-parallel graphs, which was generalized to two-dimensional partial orders in a series of papers of Correa and Schulz (2005) and Ambühl and Mastrolilli (2009). Chekuri and Motwani (1999) and Margot et al. (2003) independently showed that any schedule consistent with a Sidney decomposition is a 2-approximation. Earlier 2-approximation algorithms were derived by Schulz (1996) and Chudak and Hochbaum (1999). Correa and Schulz (2005) showed that all known 2-approximations are consistent with a Sidney decomposition. Sidney's decomposition theorem and the resulting 2-approximation algorithm was generalized to the case of MSOP with f submodular and g supermodular in Fokkink et al. (2019), where further applications to scheduling and search problems were given.

2.2 Min Sum Set Cover and its Generalizations

Min Sum Set Cover was first introduced by Feige et al. (2002). An instance of MSSC consists of a finite ground set V and a collection of subsets (or *hyperedges*) E of V. For a given linear ordering (or permutation) $\pi : V \to [n] := \{1, \dots, n\}$ of the elements of V, the *covering time of set* $e \in E$ is the first point in time that an element contained in e appears in the linear ordering, i.e., $\pi(e) := \min\{\pi(v) \mid v \in e\}$. The objective is to find a linear ordering that minimizes the total sum of covering times, $\sum_{e \in E} \pi(e)$.

MSSC is closely related to Minimum Color Sum (MCS), which was introduced by Kubicka and Schwenk (1989), and can be shown to be a special case of MSSC (though the reduction is not of polynomial size – see Feige et al. (2002)). MSSC and MCS are min sum variants of the well-known Set Cover and Graph Coloring problems, respectively.

Iwata et al. (2012) introduced a generalization of MSSC called the Minimum Linear Ordering Problem, which can be regarded as the specialcase of MSPP where g is the cardinality function. (In fact, Iwata et al. (2012) perform the summation in the opposite order from (1), but of course the objective is the same.) We give here a slightly different reduction of MSSC to MSPP. Taking F to be 2^V, for a subset $S \in F$, we define $g(S)$ to be the number of hyperedges that contain some element of S and $f(S)$ to be the cardinality of S. Then the total sum of covering times is given by (1). The dual of this problem (see Section 5) corresponds to the reduction of Iwata et al. (2012).

Kubicka and Schwenk (1989) observed that MCS can be solved in linear time for trees, and Bar-Noy and Kortsarz (1998) proved that it is APX-hard already for bipartite graphs. For general graphs, Bar-Noy et al. (1998) showed that a greedy algorithm is 4-approximate for MCS. Feige et al. (2002) observed that the greedy algorithm of Bar-Noy et al. (1998) applied to MSSC, which is to choose the element that is contained in the most uncovered sets next, yields a 4-approximation algorithm for MSSC. They simplified the proof by analyzing the performance ratio via a time-indexed linear program instead of comparing the greedy solution directly to the optimum. In the journal version of their paper, Feige et al. (2004) further simplified the proof to an elegant histogram framework, which inspired the results of this paper, and proved that one cannot approximate MSSC strictly better than 4, unless P = NP.

Munagala et al. (2005) generalized MSSC by introducing non-negative costs c_v on the elements of V and non-negative weights w_e on the sets in E. The task is to find a linear ordering π that minimizes the sum of weighted covering costs of the sets, $\sum_{e \in E} w_e C(e)$. The covering cost of $e \in E$ is defined as $C(e) := \min\{\sum_{u \in V : \pi(u) \leq \pi(v)} c_u \mid v \in e \in E\}$ (that is, the sum of all the costs of all the elements of V chosen up to and including the element that covers e). This problem is known as *pipelined set cover*. A natural extension of the greedy algorithm is to pick the element v that maximizes the ratio of the sum of the weights of the sets covered by v and the cost of v. In fact, Munagala et al. (2005) showed that this greedy algorithm is 4-approximate for pipelined set cover.

Pipelined set cover can be expressed in the form of MSOP by taking $f(S) = \sum_{v \in S} c_v$ for a subset $S \subseteq V$ and $g(S)$ to be the sum of the weights of all the subsets in E that contain at least one element of S. In this case, g is submodular and f is modular, and the fact that the greedy algorithm is 4-approximate follows from Theorem 1 of this paper (or more specifically, from Corollary 2).

Yet another generalization of MSSC is precedence-constrained MSSC. Here, the sets are subject to AND-precedence constraints and the task is to find a linear extension of the partial order on the sets. This problem was studied by McClintock et al. (2017), who proposed a $4\sqrt{n}$-approximation algorithm for precedence-constrained MSSC using a similar approach to ours: first, apply a $\sqrt{n}$-

approximation for finding a subset of V of maximum density, and then use a histogram-type argument, which yields an additional factor of 4. This result also follows from Theorem 1 of this paper (assuming the approximation result for the maximum density problem).

2.3 OR-Scheduling

One can interpret pipelined set cover as a single-machine scheduling problem in the following way. There is a job j_v for every element $v \in V$ with processing time $p_{j_v} = c_v$ and weight $w_{j_v} = 0$, and a job j_e for every $e \in E$ with processing time $p_{j_e} = 0$ and weight $w_{j_e} = w_e$. Further, there are *OR-precedence constraints* between job j_v and all jobs j_e with $v \in e$. That is, job j_e becomes available for processing after *at least one* of its predecessors in $\{j_v \mid v \in e \in E\}$ is completed. Then, finding a linear ordering of V that minimizes the sum of weighted covering costs is equivalent to finding a feasible single-machine schedule that minimizes the sum of weighted completion times.

Formally, OR-scheduling is defined as follows: Let N be a set of jobs that are subject to precedence constraints given by a (DAG) $G = (N, E)$. An arc $(i, j) \in E$ indicates that job i is an OR-predecessor of j. Any job j with $\{i \in N \mid (i, j) \in E\} \neq \emptyset$ requires that at least one of its predecessors is completed before it can start. A job without predecessors may be scheduled at any point in time. The task is to find a feasible schedule, i.e., each job is processed non-preemptively for p_j units of time, and at each point in time at most one job is processed, that minimizes the sum of weighted completion times.

To see that OR-scheduling is indeed a special case of MSOP, let Σ be the set of feasible schedules, and let $\mathcal{F} = \mathcal{F}^{\Sigma}$. In other words, $S \in \mathcal{F}$ if and only if for any job in S with predecessors, at least one of its predecessors is contained in S as well. Clearly, $\mathcal{F}$ is closed under union. Further, we set $f(S) = \sum_{j \in S} p_j$ and $g(S) = \sum_{j \in S} w_j$ for every set of jobs $S \subseteq N$. With these modular functions, it is not hard to see that the sum of weighted completion times of a schedule is equal to (2).

Note that, for the above reduction from pipelined set cover, the set of jobs can be partitioned into $N = A \dot\cup B$ such that all arcs in the precedence graph go from A to B. We call such a precedence graph *bipartite*. In a recent paper, Happach and Schulz (2020a) presented a 4-approximation algorithm for scheduling with bipartite OR-precedence constraints using an approach similar to ours. For bipartite OR-scheduling, the maximum density sets can be computed in polynomial time, so a histogram argument yields a 4-approximation algorithm. In Section 4.1, we will show that yet another class of OR-precedence constraints, namely inforests, admit a 4-approximation algorithm.

Scheduling with OR-precedence constraints was previously considered in the context of AND/OR-networks, see, e.g., Gillies and Liu (1995); Erlebach et al. (2003). In this case, Erlebach et al. (2003)

presented the best-known approximation factor, which is linear in the number of jobs, and showed that obtaining a polynomial time constant-factor approximation algorithm is NP-hard. For the case where the AND/OR-constraints are of a similar bipartite structure as above, and no AND-constraints are within $B \times A$, Happach and Schulz (2020b) obtained a 2Δ-approximation algorithm with Δ being the maximum number of OR-predecessors of any job in B. Johannes (2005) proved that minimizing the sum of weighted completion times with OR-constraints is already NP-hard for unit-processing time jobs. Happach and Schulz (2020a) strengthened this result and showed that the problem remains NP-hard even for bipartite OR-precedence constraints with unit processing times and 0/1 weights, or 0/1 processing times and unit weights.

2.4 Boolean Function Evaluation

In the non-adaptive Stochastic Boolean Function Evaluation problem (non-adaptive SBFE), we are given a Boolean function $\varphi(x_1, \dots, x_n)$ that must be evaluated on an initially unknown random assignment to its input variables. The random assignment is assumed to be drawn from a product distribution, that is, a joint distribution on the x_i's, where the x_i's are independent. Let $p_i := P[x_i = 1]$. We also assume $0 < p_i < 1$.

The value of an x_i in the random assignment can only be ascertained by performing a test, which we call *test i*. Performing test i incurs a positive integer cost c_i, and its outcome is the value of x_i. Tests are performed sequentially until there is enough information to determine the value of φ.

The task in the non-adaptive SBFE problem is as follows: Given φ, the p_i's, and the c_i's, find a linear ordering of the tests (i.e., a *non-adaptive* strategy) that minimizes the expected value of the sum of the testing costs incurred in determining the value of φ.

The non-adaptive SBFE problem can be formulated as an MSOP problem by setting $F = 2^V$, where $V = \{1, \dots, n\}$ is the set of tests, $f(S) = \sum_{i \in S} c_i$, and $g(S)$ is equal to the probability that the value of φ can be determined from the outcomes of the tests in S.

SBFE problems are a subclass of sequential testing problems that have been studied under different names in a variety of application areas, including database query optimization, artificial intelligence, and product quality testing. An excellent survey of exact algorithms for SBFE problems was written by Ünlüyurt (2004). More recent work addresses approximation algorithms (e.g., Deshpande et al. (2016), Allen et al. (2017) and Gkenosis et al. (2018)). Much of the above work considers *adaptive* SBFE problems,where the goal is to find an adaptive testing strategy of minimum expected cost. In an adaptive strategy, the choice of the next test can differ depending on the outcomes of the previous tests.

An easy case of the SBFE problem is where φ is the Boolean OR function, $\varphi(x_1, \ldots, x_n) = x_1 \vee \ldots \vee x_n$. In this case, the same solution is optimal for both the non-adaptive and adaptive problem versions: perform the tests in decreasing order of the ratio p_i/c_i until the value of f can be determined (which occurs as soon as an $x_i = 1$ is found, or after all x_i have been found to equal 0). This optimal solution has been rediscovered many times (cf. Ünlüyurt (2004)).

A more general and challenging case is where φ is given by a (monotone) read-once formula. A read-once formula, also called an AND/OR tree, is a rooted tree with the following properties. Each internal node of the tree is labeled either OR or AND (corresponding to OR or AND gates). The leaves of the tree are labeled with Boolean variables $x_1, \ldots, x_n$, where n is the number of leaves, with each x_i appearing in exactly one leaf. Given a Boolean assignment to the variables in the leaves, the value of the formula on that assignment is defined recursively in the usual way: the value of a leaf labeled x_i is the assignment to x_i, and the value of a tree whose root is labeled OR (respectively, AND) is the Boolean OR (respectively, AND) of the values of the subtrees of that root. Read-once formulas are equivalent to series-parallel systems (cf. Ünlüyurt (2004)).

An example read-once formula, corresponding to the expression $\varphi(x_1, x_2, x_3, x_4, x_5) = x_1 \wedge x_2 \wedge ((x_3 \wedge x_4) \vee x_5)$, is shown in Figure 1. Consider evaluating this formula using the non-adaptive strategy represented by the permutation (3, 4, 5, 2, 1). Suppose, for example, that the first test reveals that $x_3 = 0$, the second that $x_4 = 1$, and the third that $x_5 = 0$. Then testing will stop after that third test, when it can be determined that the value of f is 0. The probability of this happening is $(1 - p_3)p_4(1 - p_5)$, and the incurred cost in this case is $c_3 + c_4 + c_5$. More generally, for each prefix R of (3, 4, 5, 2, 1), let P_R denote the probability that testing stops at the end of that prefix. Using q_i to denote $1 - p_i$, we have e.g., $P_{(3)} = P_{(3,4)} = 0$, $P_{(3,4,5)} = (1 - p_3p_4)q_5$ and $P_{(3,4,5,2)} = (1 - P_{(3,4,5)})q_2$. The expected cost of using the non-adaptive strategy specified by (3, 4, 5, 2, 1) to evaluate f is $c_3 + c_4 + c_5 + c_2(1 - P_{(3,4,5)}) + c_1(1 - P_{(3,4,5,2)})$. Equivalently, the expected cost is equal to $\sum_{j=1}^{5} f(S_j)(g(S_j) - g(S_{j-1}))$, where S_j is the set consisting of the first j elements of the permutation (3, 4, 5, 2, 1), $f(S_j) = \sum_{i \in S_j} c_i$, and $g(S_j)$ is the probability that the value of φ can be determined by performing just the tests in S_j. Thus $g(S_j) - g(S_{j-1}) = P_{R^j}$, where R^j denotes the prefix consisting of the first j elements of the permutation.

The adaptive SBFE problem for read-once formulas has been studied in a number of papers since the 1970's (cf. Ünlüyurt (2004); Greiner et al. (2006)). It is not known whether there is a polynomial-time algorithm for solving the problem or whether it is NP-hard, even in the unit-cost case. It is also unknown whether there is a polynomial-time approximation algorithm with an approximation factor sublinear in n.

The *non-adaptive* SBFE problem for general read-once formulas does not appear to have been

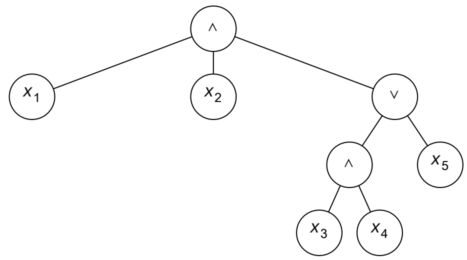

Figure 1: Read-once formula

studied previously. As with the adaptive problem, we do not know whether the non-adaptive problem is NP-hard. It is easy to show that, in contrast to the case of the OR function, an optimal non-adaptive strategy for evaluating a read-once formula will generally incur higher expected cost than an optimal adaptive strategy. This is because, for example, learning that a variable $x_i = 0$ when its parent node is labeled AND allows us to "prune" all other subtrees of that AND node, making it unnecessary to perform tests on any of the variables that were in the pruned subtrees.

In Section 4.3 we show that there is an 8-approximation algorithm for the non-adaptive SBFE problem for read-once formulas. We do this by giving a 2-approximation algorithm for the associated maximum density problem. In the unit-cost case, the algorithm runs in polynomial time. For general costs c, the runtime of the algorithm depends on the costs, and runs in pseudo-polynomial time.

We note that Işık and Ünlüyurt (2013) briefly considered what might be called a "partially-adaptive" version of the SBFE problem for read-once formulas, where the strategy is specified by a permutation, but tests in the permutation are skipped if the results of previous tests have rendered them irrelevant. There does not appear to be a way to formulate this version of the problem as an MSOP.

3 Approximating MSOP

In this section, we first prove our main result in Subsection 3.1. In Subsection 3.2 we then justify the observation made in Section 2 that MSPP is really a special case of MSOP.

3.1 Proof of Main Result

For a set $A \in F$, we write f_A for the function on F given by $f_A(S) = f(S) - f(A)$, and similarly for g_A. For $f_A(S) \neq 0$, let $\rho_A(S) = g_A(S)/f_A(S)$ be the *marginal density* of S (with respect to A). If $A = \varnothing$, we drop the subscript from ρ and simply refer to $\rho(S)$ as the density of S.

We consider a greedy algorithm for MSOP. For $\alpha \geq 1$, we call an F-chain $S = (S_j)_{j=0}^{k}$ an *α-greedy chain* if

$$\rho_{S_j}(S_{j+1}) \geq \frac{1}{\alpha} \max_{\{T \in F : S_j \subseteq T\}} \rho_{S_j}(T),$$

for all $j = 0, \dots, k-1$. If $\alpha = 1$, an α-greedy chain is simply one for which S_{j+1} has maximum marginal density with respect to S_j for each $j = 0, \dots, k-1$.

We now prove Theorem 1, which is a generalization of the result and proof in Feige et al. (2004).

Proof of Theorem 1. Let $T = (T_j)_{j=0}^{\ell}$ be an optimal chain and let $S = (S_i)_{i=0}^{k}$ be an α-greedy chain. We first construct a histogram with ℓ columns, the area under which is equal to $C_{f,g}(T)$. The base of the jth column of the histogram is the interval from $g(T_{j-1})$ to $g(T_j)$ and its height is $f(T_j)$. Thus, the total area under the histogram is equal to $C_{f,g}(T)$.

Next, we construct a second histogram with k columns, the area under which is equal to $C_{f,g}(S)$. Let $\rho_i = \rho_{S_{i-1}}(S_i)$ and let $\phi_i = \rho_i^{-1}(g(V) - g(S_{i-1}))$. The base of the ith column of this histogram is the interval from $g(S_{i-1})$ to $g(S_i)$ and its height is ϕ_i. Thus the total area A under this histogram is

$$\begin{aligned}
A &= \sum_{i=1}^{k} \phi_i (g(S_i) - g(S_{i-1})) \\
&= \sum_{i=1}^{k} (g(V) - g(S_{i-1}))(f(S_i) - f(S_{i-1})) \qquad (3) \\
&= g(V) \sum_{i=1}^{k} (f(S_i) - f(S_{i-1})) - \sum_{i=1}^{k} g(S_{i-1})(f(S_i) - f(S_{i-1}))
\end{aligned}$$

The first sum on the right-hand side above is telescopic and equal to $f(V) - f(\emptyset) = f(V)$. Rearranging the second sum, we obtain

$$\begin{aligned}
A &= g(V)f(V) - g(S_{k-1})f(V) + \sum_{i=1}^{k-1} f(S_i)(g(S_i) - g(S_{i-1})) \\
&= \sum_{i=1}^{k} f(S_i)(g(S_i) - g(S_{i-1})) \\
&= C_{f,g}(S).
\end{aligned}$$

The two histograms are depicted in Figure 2(a). Note that the heights of the columns in the first histogram, from left to right, are non-decreasing.

Now shrink the second histogram by a factor of 2α in the vertical direction, and a factor of 2 in the horizontal direction, and move it to the right so it is flush with the right end $g(V)$, as depicted in Figure 2(b). This results in point (x, y) being mapped to $(\frac{g(V)+x}{2}, \frac{y}{2\alpha})$. The distance of this latter point from the right end is $(g(V) - x)/2$.

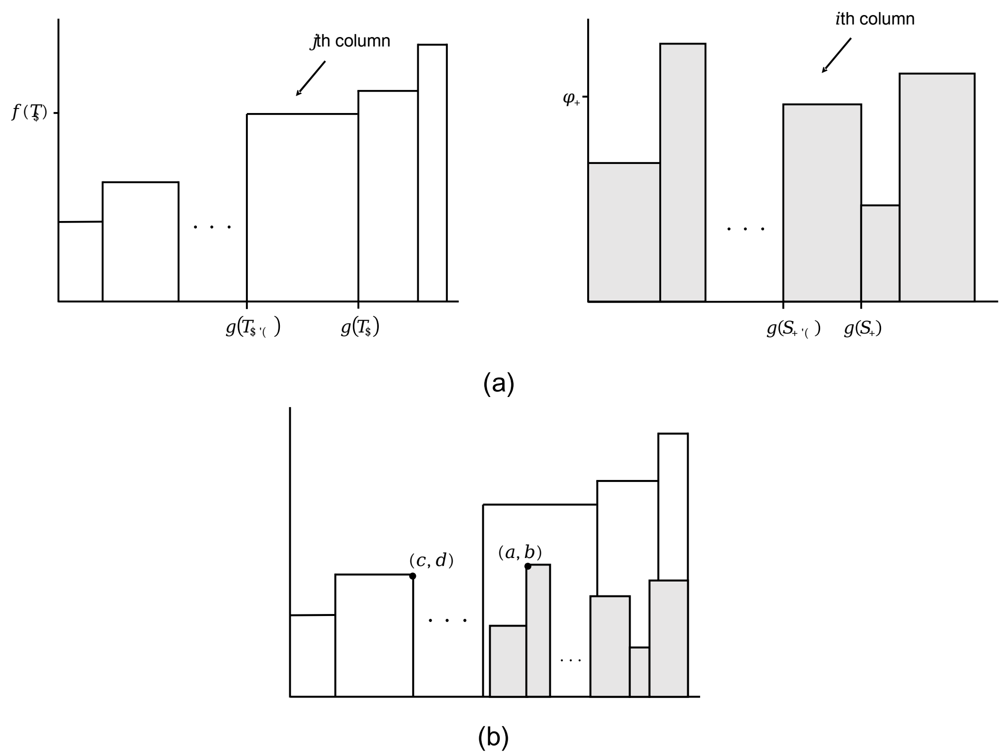

Figure 2: (a) Two histograms with total area $C_{f,g}(T)$ (left) and $C_{f,g}(S)$ (right); (b) the shrunken version of the second histogram in the first histogram.

We now show that the shrunken (and shifted) histogram is contained in the first histogram, from which it follows that $C_{f,g}(S) \leq 4\alpha C_{f,g}(T)$, proving the theorem. To show that the shrunken histogram is contained in the first histogram, it is sufficient to show that if (a, b) is the top left point of some column i in the shrunken histogram, and (c, d) is the top right point of some column j in the first histogram, then $d < b$ implies that $c < a$. Here $(a, b) = \frac{g(V)+g(S_{i-1})}{2}, \frac{\phi_i}{2\alpha}$ and $(c, d) = (g(T_j), f(T_j))$.

So assume $d < b$, or equivalently

$$f(T_j) < \frac{\rho_i^{-1}(g(V) - g(S_{i-1}))}{2\alpha}. \tag{4}$$

Let a^0 be the distance of (a, b) from the right boundary, that is, $a^0 = (g(V) - g(S_{i-1}))/2$ and let c^0 be the distance of (c, d) from the right boundary, that is, $c^0 = g(V) - g(T_j)$. We want to show that $d < b$ implies $c < a$, or equivalently that $a^0 < c^0$. So we want to show that

$$\frac{g(V) - g(S_{i-1})}{2} < g(V) - g(T_j). \tag{5}$$

Since F is closed under union, $S_{i-1} \cup T_j \in F$. We will use the fact that, because S is α-greedy,

$$\rho_i = \rho_{S_{i-1}}(S_i) \geq \frac{1}{\alpha}\rho_{S_{i-1}}(S_{i-1} \cup T_j) = \frac{g(S_{i-1} \cup T_j) - g(S_{i-1})}{\alpha(f(S_{i-1} \cup T_j) - f(S_{i-1}))}. \tag{6}$$

Because f is subadditive, $f(S_{i-1} \cup T_j) - f(S_{i-1}) \leq f(T_j)$. Combining that fact with (6) yields

$$\begin{aligned}\alpha f(T_j) &\geq \rho_i^{-1}(g(S_{i-1} \cup T_j) - g(S_{i-1}))\\ &= \rho_i^{-1}((g(V) - g(S_{i-1})) - (g(V) - g(S_{i-1} \cup T_j)))\end{aligned}$$

Using our assumption in (4), we thus get

$$\begin{aligned}\frac{\rho_i^{-1}(g(V) - g(S_{i-1}))}{2} &> \rho_i^{-1}((g(V) - g(S_{i-1})) - (g(V) - g(S_{i-1} \cup T_j)))\\ &\geq \rho_i^{-1}((g(V) - g(S_{i-1})) - (g(V) - g(T_j))),\end{aligned}$$

where the second inequality follows from the fact g is non-decreasing. Rearranging gives (5). $\square$

Observe that if $F = 2^V$ and f is supermodular and g is submodular, then for $S_j \subseteq T$,

$$\rho_{S_j}(T) \leq \frac{\sum_{v \in T \setminus S_j} g(S_j \cup \{v\}) - g(S_j)}{\sum_{v \in T \setminus S_j} f(S_j \cup \{v\}) - f(S_j)} \leq \max_{v \in T \setminus S_j} \rho_{S_j}(S_j \cup \{v\}).$$

Hence, a 1-greedy chain can be obtained in polynomial time by adding singletons one-by-one. Suppose f is not just supermodular but also modular. Then f is also subadditive so, by Theorem 1, there is a polynomial time 4-approximation algorithm. We summarize this observation below.

Corollary 2 Suppose $F = 2^V$. If f is modular and g is submodular then a 1-greedy chain can be constructed in polynomial time and there exists a polynomial-time 4-approximation algorithm for MSOP.

As discussed in Subsection 2.2, the problem MSSC and, more generally, pipelined set cover, are special cases of MSOP where g is submodular and f is modular, so the 4-approximation algorithms for these problems follow from Corollary 2.

3.2 Minimizing Over Permutations

We now turn to the problem MSPP, given in (2), where we wish to minimize a weighted sum over a set of permutations Σ. We will show that provided Σ satisfies a certain technical condition, solving MSPP for Σ is equivalent to solving the corresponding MSOP problem (with the same f and g) for $F = F^{\Sigma}$.

Suppose F is some family of subsets of V, and suppose $S \equiv (S_j)_{j=0}^k$ is an F-chain. Then if $1 = j_1 \leq \ldots \leq j_\ell = k$, we say $S' \equiv (S_{j_i})_{i=0}^{\ell}$ is a *subchain* of S.

Lemma 3 Suppose $S \equiv (S_{j_i})_{i=0}^{\ell}$ is a subchain of the F-chain $S' \equiv (S_j)_{j=0}^{k}$. Then

(i) $C_{f,g}(S) \geq C_{f,g}(S')$ and

(ii) if S approximates MSOP by a factor of $\alpha \geq 1$ then so does S'.

Proof. We perform the following calculation.

$$\begin{aligned}
C_{f,g}(S) &\equiv \sum_{i=1}^{\ell} f(S_{j_i})(g(S_{j_i}) - g(S_{j_{i-1}})) \\
&= \sum_{i=1}^{\ell} f(S_{j_i}) \sum_{j=j_{i-1}+1}^{j_i} (g(S_j) - g(S_{j-1})) \\
&\geq \sum_{i=1}^{\ell} \sum_{j=j_{i-1}+1}^{j_i} f(S_j)(g(S_j) - g(S_{j-1})) \\
&= \sum_{j=1}^{k} f(S_j)(g(S_j) - g(S_{j-1})) \equiv C_{f,g}(S'),
\end{aligned}$$

where the inequality above follows from the monotonicity of f and g. Part (ii) of the lemma follows immediately. □

Suppose now that Σ is a set of permutations of V. Recall that F^{Σ} consists of all subsets of V that are initial sets of some permutation $\sigma \in \Sigma$. If for some F^{Σ}-chain S, there exists a permutation $\sigma \in \Sigma$ such that each element of S is an initial set of σ, then we say σ is *consistent* with S. If every F^{Σ}-chain is consistent with some permutation in Σ then we say Σ is *well-founded*.

Lemma 4 Suppose Σ is a set of permutations of V and that Σ is well-founded. If there exists a polynomial time α-approximation algorithm for MSOP with $F = F^{\Sigma}$ for some $\alpha \geq 1$ then there exists a polynomial time α-approximation algorithm for MSPP.

Proof. This follows immediately from Lemma 3, part (ii). Indeed, suppose that S is an α-approximate F^{Σ}-chain and that σ is consistent with S. Let S' be the chain consisting of all the initial sets of σ. Then S is a subchain of S', so S' is an α-approximation for MSOP. Equivalently, σ is an α-approximation for MSPP. □

It is easy to think of examples of Σ that are not well-founded. For example, if $V = \{1, 2, 3\}$ and Σ contains only the permutations (1, 2, 3) and (3, 1, 2), then the F^{Σ}-chain $\{\{1\}, \{1, 3\}, \{1, 2, 3\}\}$ is not consistent with either of the two permutations, so Σ is not well-founded.

However, for all the examples we consider in this paper, the set of permutations is well-founded. This is easy to check by using the following sufficient condition.

For two permutations σ and τ of V, let $\pi_j(\sigma, \tau)$ be the permutation that follows σ for the first j elements then chooses the remaining elements of V in the order specified by τ, for each $1 \leq j \leq n$. For example, if $V = \{1, 2, 3, 4, 5\}$, and σ and τ are given by (3, 1, 5, 2, 4) and (4, 5, 1, 2, 3), respectively then $\pi_2(\sigma, \tau)$ is given by (3, 1, 4, 5, 2) and $\pi_3(\sigma, \tau)$ is given by (3, 1, 5, 4, 2).

If Σ is a set of permutations for which $\pi_j(\sigma, \tau) \in \Sigma$ for any $\sigma, \tau \in \Sigma$ and $1 \leq j \leq n$, then we say Σ is *closed*.

Lemma 5 Let Σ be a set of permutations of V. If Σ is closed then it is well-founded and F^{Σ} is closed under union.

Proof. Suppose Σ is closed. Let $S = (S_j)_{j=1}^{k}$ be a F^{Σ}-chain. We will show that there is some permutation contained in Σ that is consistent with S. Let σ_j be a permutation in Σ that is consistent with S_j for $j = 1, \ldots, k$. We set $\tau_1 = \sigma_1$ and for $j = 2, 3, \ldots, k$, we recursively define $\tau_j = \pi_{|S_{j-1}|}(\tau_{j-1}, \sigma_j)$, which is contained in Σ, by induction on j and since Σ is closed. Also, S_1 is an initial set of τ_1, and, by definition of $\pi_{|S_{j-1}|}(\tau_{j-1}, \sigma_j)$ and by induction on j, each of $S_1, \ldots, S_j$ are initial sets of τ_j for $j \geq 2$. Therefore, τ_k is consistent with S, so Σ is closed.

To see that F^{Σ} is closed under union, let S and T be elements of F^{Σ}. Then they are initial sets of some permutations σ and τ in Σ, so $S \cup T$ is an initial set of $\pi_{|S|}(\sigma, \tau)$, which lies in Σ, since Σ is closed. Therefore $S \cup T \in F^{\Sigma}$. 2

We observe that for the expanding search problem, if we take Σ to be the set of expanding searches,then it is easy to check that Σ is closed and therefore well-founded, by Lemma 5. Furthermore, F^{Σ} is closed under union. Similarly, for both AND-precedence constraints and OR-precedence constraints, the set of feasible orderings is closed and therefore well founded. If Σ consists of all permutations of V, as in Boolean function evaluation, then Σ is trivially well-founded. It follows from Lemma 4 that for these problems, if we can find a solution (or approximate solution) S to MSOP with $F = F^{\Sigma}$, then we can recover a solution (or approximate solution) to the original problem by taking any permutation that is consistent with S. Since in each case F^{Σ} is closed under union, we only need f to be subadditive to apply Theorem 1.

4 Applications

We now describe some special cases of MSOP for which our results imply the existence of approximation algorithms.

We will begin in Subsection 4.1 by first providing 4-approximation algorithms for OR-scheduling (as defined in Subsection 2.3) on inforests and, more generally, on *multitrees*.We then consider a new OR-precedence constrained version of MSSC in Subsection 4.2 and show that there is also a 4-approximation algorithm for this. Lastly, in Subsection 4.3, we give an 8-approximation algorithm for a problem in Boolean function evaluation.

4.1 OR-Scheduling

We consider a special case of MSOP that can be stated in terms of MSPP. Suppose the elements of V are vertices of a DAG $G = (V, E)$ which represents some OR-precedence constraints. That is, a permutation σ of V is feasible if each element v with a non-empty set of predecessors appears in σ later than at least one of its direct predecessors $P(v)$ (where $P(v)$ is the set of u such that $(u, v) \in E$). Recall that a DAG $G = (V, E)$ is an *intree* if every vertex has at most one successor. A DAG whose connected components are intrees is an inforest.

Theorem 6 Consider an instance of MSPP for which the set of feasible permutations is derived from some OR-precedence constraints given by an inforest. Then if f is modular and g is submodular, there is a polynomial time 4-approximation algorithm for the problem.

Proof. Note f is modular and hence subadditive. Also, as pointed out in Subsection 3.2, the set of feasible permutations Σ is closed and therefore, by Lemma 5, the set Σ is consistent and F^{Σ} is closed under union. Hence, by Theorem 1 and Lemma 4, it suffices to construct in polynomial time a 1-greedy F^{Σ}-chain.

We characterize inclusion-minimal sets of maximum density, using the concept of a *stem*. We define a stem in G to be a sequence of vertices $v_1, \ldots, v_k$ in V such that v_1 has no predecessors and $v_i \in P(v_{i+1})$ for all $i = 1, \ldots, k-1$. We show that any inclusion-minimal subset S_{j+1} of V that maximizes the density $\rho_{S_j}(S_{j+1})$ is a stem and that we can enumerate all stems in polynomial time. Observe that, if we remove a stem S_j from the instance along with all edges incident to vertices in the stem, the graph decomposes into intrees again; also f_{S_j} is modular and g_{S_j} is submodular. So it suffices to consider only $S_j = \emptyset$.

Since G is an inforest, the number of paths starting at any vertex is bounded by the total number of vertices. Therefore, the total number of stems is $O(n^2)$. So we can enumerate all stems S that start at a job without a predecessor, and pick the one of maximum density $\rho(S) = g(S)/f(S)$. It remains to show that a stem of maximum density is indeed an OR-initial set of maximum density.

Let $S \in F$ be an inclusion-minimal set of maximum density and suppose that S is not a stem. Since G is an inforest, S must be an inforest that contains at least two vertices without a predecessor. Since every vertex has at most one successor, any vertex without a predecessor induces a unique stem to the root of its connected component in S (the root of a component being the

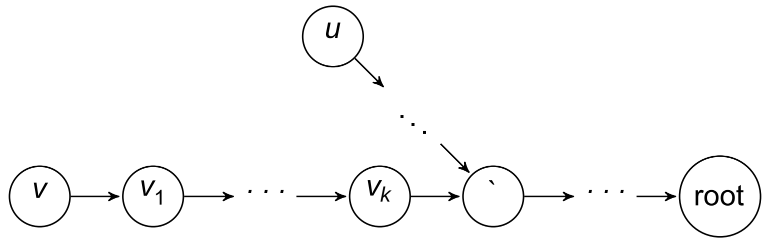

Figure 3: Paths starting at v and u to the root, and their intersection point `.

unique vertex contained in that component that has no successors). For two such vertices, v and u, let ` be the vertex where the unique stems starting at $v$ and $u$ meet, see Figure 3. (Note that ` does not exist if v and u are in different connected components.) Now, let $T_v := \{v, v_1, \ldots, v_k\}$ be the set of vertices on the stem from v to ` such that $v_k \in P(`)$, i.e., ` $= v_{k+1}$. If ` does not exist, then let T_v be the stem starting at v to its root in S. Clearly, $T_v \in F$, and also $\overline{T} := S \setminus T_v$ is an OR-initial set.

By the submodularity of g, the density $\rho(S)$ satisfies

$$\rho(S) \leq \frac{g(T_v) + g(\overline{T})}{f(T_v) + f(\overline{T})} = \theta\rho(T_v) + (1-\theta)\rho(\overline{T}), \tag{7}$$

where $\theta = f(T_v)/(f(T_v) + f(\overline{T})) \in [0, 1]$. Hence, by the maximality of $\rho(S)$, both T_v and $\overline{T}$ must have maximum density, contradicting the assumption that S was an inclusion-minimal subset of maximum density.

2

Recall from Subsection 2.3 the problem of minimizing the sum of weighted completion times of a set of jobs, where the order that the jobs are processed must respect some OR-precedence constraints given by some DAG G. Suppose that G is an inforest. Since the cost function and the weight function are both modular, the next theorem follows immediately from Theorem 6.

Theorem 7 There is a polynomial time 4-approximation algorithm for OR-scheduling of inforests.

In fact, we derive a more general result for OR-scheduling of *multitrees*, introduced in Furnas and Zacks (1994). A DAG is called a multitree if, for every vertex, its successors form an outtree (where an outtree is a DAG such that each vertex has at most one predecessor). Equivalently, there is at most one directed path between any two vertices. Inforests are examples of multitrees.

Theorem 8 There is a polynomial time 4-approximation algorithm for OR-scheduling of multi-trees.

Proof. By Theorem 13 and Lemma 4, it is sufficient to find a 1-greedy F-chain, where $F = F^{\Sigma}$ and Σ is the set of feasible schedules.

We will show that the inclusion-minimal sets of maximum density are outtrees. This means that we can find a maximum density subset of F by considering each vertex v with no predecessors and finding a maximum density subtree $T_v \in F$ of the outtree formed by v and its sucessors. This can be done in polynomial time using the dynamic programming algorithm of Horn (1972), for example. We then choose a subtree T_v with maximum density over all vertices v with no predecessors.

The proof that the inclusion-minimal sets of maximum density are outtrees is similar to the proof that the inclusion-minimal subsets of Theorem 6 were stems, so we do not go into detail. It can be shown that if S is an inclusion-minimal set of maximum density that is not an outtree, then it can be expressed as the disjoint union of an outtree T and another set in F. By an identical calculation as in (7), the set T must have maximum density, contradicting S being inclusion-minimal. This completes the proof. *2*.

It is worth pointing out that approximating the problem of minimizing the sum of weighted completion times for OR-scheduling appears to be harder than the analogous problem for AND-scheduling in the following sense. As discussed in Subsection 2.1, for AND-scheduling there is a polynomial time algorithm for series-parallel DAGs and polynomial time 2-approximation algorithms for arbitrary DAGs, whereas for OR-scheduling, we have given polynomial time 4-approximation algorithms for inforests and, more generally, multitrees. It is not possible that better approximations exist for OR-scheduling on multitrees (or even for bipartite graphs) unless P = NP, since the same is true of MSSC, which is a special case of OR-scheduling on bipartite graphs. Of course, for outtrees, OR-scheduling and AND-scheduling are equivalent, but there are no polynomial time algorithms known for OR-scheduling on any other classes of DAGS.

4.2 OR-Precedence Constrained MSSC and Pipelined Set Cover

Consider a variation of MSSC in which the order that the elements of V are chosen must be consistent with some OR-precedence constraints given by a DAG G. As in pipelined set cover, we additionally assume that there is a non-negative cost c_v for each vertex v and a non-negative weight w_e for each hyperedge $e \in E$, and the objective is to minimize the weighted sum of covering times of the edges. As for OR-scheduling, we take F to be the collection of OR-initial sets of G and as for pipelined set cover, we take $f(S) = \sum_{v \in S} c_v$ and $g(S)$ to be the sum of the weights of all hyperedges in E that contain at least one element of S, for $S \in F$. Then f is modular and g is submodular, so we can again apply Theorem 6.

Theorem 9 There is a polynomial time 4-approximation algorithm for pipelined set cover with OR-precedence constraints that take the form of an inforest.

To see that Theorem 9 is a generalization Theorem 7, simply observe that if the set of hyperedges E consists of all the singletons of V, then pipelined set cover with OR-precedence constraints is equivalent to OR-scheduling.

4.3 Evaluation of Read-Once Formulas

Recall the inputs to the non-adaptive SBFE problem for read-once formulas: (1) a read-once formula $\varphi(x_1, \ldots, x_n)$, (2) for each x_i, the value $p_i := P[x_i = 1]$ where $0 < p_i < 1$, and (3) for each x_i, the associated test cost c_i, which is greater than 0.

We assume without loss of generality that each AND and OR gate of φ has fan-in 2. We consider each input x_i in φ to also be a gate (an *input gate*) of φ.

We use partial assignments to represent the outcomes of a subset of the tests. In a partial assignment $b \in \{0, 1, *\}^n$, $b_i = *$ means that test i has not been performed and the value of x_i is unknown, otherwise b_i is the outcome of test i. For (full) assignment $a \in \{0, 1\}^n$ and S a subset of $V = \{1, \ldots, n\}$, $a|_S$ is the partial assignment $b \in \{0, 1, *\}^n$ where $b_i = a_i$ for $i \in S$, and $b_i = *$ otherwise.

Given partial assignment $b \in \{0, 1, *\}^n$, an *extension* of b is a (full) assignment $a \in \{0, 1\}^n$ where $a_i = b_i$ for all i such that $b_i \neq *$. If $\varphi(a)$ has the same value ℓ for all extensions a of b, the value of φ *is determined by* b and we write $\varphi(b) = \ell$. Otherwise, we write $\varphi(b) = *$.

Let $A_1, \ldots, A_n$ be independent Bernoulli random variables, where $P[A_i = 1] = p_i$. Let $A = [A_1, \ldots, A_n]$.

In the MSOP formulation of the non-adaptive SBFE problem for read-once formulas, $F = 2^V$ where $V = \{1, \ldots, n\}$ is the set of tests, $f(S) = \sum_{i \in S} c_i$, and $g(S) = P[\varphi(A|_S) \neq *]$.

The set F is closed under union and f is modular. We will obtain an 8-approximate solution to this MSOP by constructing a 2-greedy chain.

For $S \subset T \subseteq V$,

$$\rho_S(T) = \frac{P[\varphi(A|_T) \neq *] - P[\varphi(A|_S) \neq *]}{\sum_{i \in T \setminus S} c_i} \tag{8}$$

By the independence of the A_i, for any partial assignment $b \in \{0, 1, *\}^n$,

$$P[A|_S = b] = \prod_{i \in S: b_i = 1} p_i \prod_{i \in S: b_i = 0} (1 - p_i).$$

For $S \subset V$ and $\alpha > 0$, call $R \subseteq V \setminus S$ an α-approximate max-density *supplement* for S if

$$\rho_S(S \cup R) \geq \frac{1}{\alpha} \max_{\{T \subseteq V : S \subset T\}} \rho_S(T).$$

For gate G of φ, define $tests(G)$ to be the set of $i \in V$ such that x_i is a descendant of G in the tree φ. We consider a gate to be its own descendant, so if G is an input gate x_i, then $i \in tests(G)$.

Each gate G of φ is the root of a subtree of φ. Define φ_G to be the subformula corresponding to the subtree of φ that is rooted at G. Thus φ_G is a read-once formula over the variable set $\{x_i \mid i \in tests(G)\}$. We treat φ_G as computing a function over $\{0, 1\}^n$, whose output depends only on the values of the variables in $\{x_i \mid i \in tests(G)\}$. For $b \in \{0, 1, *\}^n$, we refer to $\varphi_G(b)$ as the output of gate G on partial assignment b, which may be either 0, 1, or $*$.

We note that given a subset $S \subseteq V$, and $\ell \in \{0, 1\}$, the value of $P[\varphi_G(A|_S) = \ell]$ for each gate G of φ can be computed in time linear in n by processing the gates of φ in bottom-up order. Consider the case where $\ell = 1$ and let $p_G = P[\varphi_G(A|_S) = 1]$. If G is an input gate x_i, then $p_G = p_i$ if $x_i \in S$, otherwise $p_G = 0$. If G is an AND gate with children G' and G'', then because φ is read-once and the A_i are independent, $p_G = p_{G'} \cdot p_{G''}$. If G is an OR gate, then $p_G = p_{G'} + p_{G''} - p_{G'} \cdot p_{G''}$.

Dually, consider the case where $\ell = 0$ and let $q_G = P[\varphi_G(A_S) = 0]$. If G is an input gate x_i, then $q_G = 1 - p_i$ if $x_i \in S$, otherwise $q_G = 0$. If G is an OR gate with children G' and G'', then $q_G = q_{G'} q_{G''}$. If G is an AND gate, then $q_G = q_{G'} + q_{G''} - q_{G'} q_{G''}$.

4.3.1 The 8-Approximation Algorithm

The 8-approximation algorithm, for the non-adaptive SBFE problem for read-once formulas, relies on the following lemma.

Lemma 10 Given $S \subset V$, a 2-approximate max-density supplement R for S can be computed in time polynomial in n and $\sum_{i=1}^{n} c_i$.

Proof. We prove the lemma for the case of unit costs, where all the c_i's are equal to 1. We then explain how to extend the proof to handle arbitrary costs.

Assume the c_i's are all equal to 1. We describe an algorithm that we call FindSupp that finds a max-density subset R for a given input subset S.

Fix S. For $R \subseteq V \setminus S$, let $\sigma(R) = \rho_S(S \cup R)$. Since we assumed the c_i's are equal to 1,

$$\sigma(R) = \frac{P[\varphi(A|_{S \cup R}) \neq *] - P[\varphi(A|_S) \neq *]}{|R|}.$$

Clearly, $P[\varphi(A|_S) \neq *] = P[\varphi(A|_S) = 1] + P[\varphi(A|_S) = 0]$ and similarly for $A|_{S \cup R}$.

For $\ell \in \{0, 1\}$, define

$$\sigma_\ell(R) = \frac{P[\varphi(A|_{S \cup R}) = \ell] - P[\varphi(A|_S) = \ell]}{|R|}.$$

Thus

$$\sigma(R) = \sigma_1(R) + \sigma_0(R) \tag{9}$$

The idea behind FindSupp is to compute two subsets R^1 and R^0, maximizing σ_1 and σ_0 respectively. By (9), the R^ℓ with the larger value of $\sigma_\ell(R^\ell)$ is a 2-approximate max-density supplement for S.

For gate G of φ, let $T(G) = tests(G) \setminus S$. For $t \in \{0, \ldots, |T(G)|\}$, and $\ell \in \{0, 1\}$, let $R_{G,t,\ell}$ be a subset R that maximizes the value of $P[\varphi_G(A|_{S \cup R}) = \ell]$ subject to the constraints that $R \subseteq T(G)$ and $|R| = t$.

Let $p_{G,t,\ell}$ be the value of $P[\varphi_G(A|_{S \cup R}) = \ell]$ for $R = R_{G,t,\ell}$. Let $\tilde{G}$ denote the root gate of φ. Thus for $t \in \{1, \ldots, |V \setminus S|\}$ and $\ell \in \{0, 1\}$, setting $R = R_{\tilde{G},t,\ell}$ maximizes the value of $\sigma_\ell(R)$, over all $R \subseteq V \setminus S$ of size t.

Algorithm FindSupp**:**

FindSupp first runs a procedure ComputeRp that computes $R_{\tilde{G},t,\ell}$ and $p_{\tilde{G},t,\ell}$ for all $t \in \{1, \ldots, |V \setminus S|\}$ and $\ell \in \{0, 1\}$. We describe the details of ComputeRp below.

After running ComputeRp, the algorithm uses it to obtain the two subsets, R^1 and R^0, maximizing σ_1 and σ_0 respectively. It does this as follows. First, using the linear-time procedure described above, it computes the value of $P[\varphi(A|_S) = \ell]$, for $\ell \in \{0, 1\}$.

For each $t \in \{1, \ldots, |V \setminus S|\}$, for $\ell \in \{0, 1\}$, FindSupp computes the value of

$$\sigma_\ell(R_{\tilde{G},t,\ell}) = \frac{p_{\tilde{G},t,\ell} - P[\varphi(A|_S) = \ell]}{t}$$

For each $\ell \in \{0, 1\}$, the algorithm then finds the value of t which yielded the highest value for $\sigma_\ell(R_{\tilde{G},t,\ell})$. Let t^ℓ denote that value. Let R^ℓ be the value of $R_{\tilde{G},t,\ell}$ for $t = t^\ell$.

Because $R_{\tilde{G},t,\ell}$ maximizes σ among candidate subsets of size t, setting $R = R^\ell$ maximizes $\sigma(R)$ among candidate subsets of all possible sizes. The algorithm returns R^0 if $\sigma_0(R^0) > \sigma_1(R^1)$, and returns R^1 otherwise.

Procedure ComputeRp**:**

For all gates G of φ, ComputeRp computes the values of $p_{G,t,\ell}$ and $R_{G,t,\ell}$ for all $t \in \{0, \ldots, |T(G)|\}$, and $\ell \in \{0, 1\}$. It processes the gates G of φ in bottom-up order, from the leaves to the root.

We begin by describing how ComputeRp computes $p_{G,t,\ell}$ and $R_{G,t,\ell}$ when G is an AND gate, $t \in \{0, \ldots, |T(G)|\}$, and $\ell = 1$. Suppose that G' and G'' are the children of AND gate G, and that $R_{G',t',1}$, $p_{G',t',1}$, $R_{G'',t'',1}$, and $p_{G'',t'',1}$ have already been computed, for all $t' \in \{0, \ldots, |T(G')|\}$ and $t'' \in \{0, \ldots, |T(G'')|\}$. ComputeRp first computes the product $p_{G',j,1} \cdot p_{G'',t-j,1}$ for all j such that

$j \in \{0, \ldots, |T(G')|\}$ and $t - j \in \{0, \ldots, |T(G'')|\}$.) It then sets j^* to be the value of j maximizing that product, and sets $R_{G,t,1} = R_{G',j^*,1} \cup R_{G'',t-j^*,1}$ and $p_{G,t,1} = p_{G',j^*,1} \cdot p_{G'',t-j^*,1}$.

The correctness of these settings follows from the fact that that $R_{G,t,1}$ must consist of a subset R' of $T(G')$ of some size j^*, and a subset R'' of $T(G'')$ of size $t-j^*$. $R_{G,t,1}$ maximizes the probability that G outputs 1 (among subsets of $T(G)$ of size t, when added to S). Since φ is a read-once formula, $tests(G')$ and $tests(G'')$ are disjoint. R' and R'' are thus sets that maximize the probability that G' and G'' output 1 (when added to S, among subsets of $T(G')$ and $T(G'')$ of sizes j^* and $t - j^*$ respectively). Thus, given j^*, $R_{G,t,1}$ can be set to $R' \cup R''$ where $R' = R_{G',j^*,1}$, $R'' = R_{G'',t-j^*,1}$, and $p_{G,t,1}$ can be set to $p_{G',j^*,1} \cdot p_{G'',t-j^*,1}$. Since ComputeRp is not given the value of j^*, it must try all possible j.

Similarly, suppose G is an AND gate, $t \in \{0, \ldots, |T(G)|\}$, and $\ell = 0$. In this case, ComputeRp computes $p_{G',j,0} + p_{G'',t-j,0} - p_{G',j,0} \cdot p_{G'',t-j,0}$ for all possible j, and then sets j^* to be the value that maximized the expression. It then sets $R_{G,t,0} = R_{G',j^*,0} \cup R_{G'',t-j^*,0}$ and $p_{G,t,0} = p_{G',j^*,0} + p_{G'',t-j^*,0} - p_{G',j^*,0} \cdot p_{G'',t-j^*,0}$. The correctness in this case follows from the fact that the output of G is 0 if either of its child gates outputs 0, and to maximize the probability that AND gate G outputs 0, one needs to maximize the probability that each of its children outputs 0.

The case where G is an OR gate is dual and we omit the details.

The remaining case is where G is an input gate x_i, $t \in \{0, \ldots, T(G)\}$, and $\ell \in \{0, 1\}$. Note that since G is an input gate, if $i \in S$, then $|T(G)| = 0$. If $i \notin S$, then $|T(G)| = 1$. If $t = 0$, then for $\ell \in \{0, 1\}$, ComputeRp sets $R_{G,t,\ell} = \emptyset$. Then, if $i \in S$ it sets $p_{G,t,1} = p_i$ and $p_{G,t,0} = 1 - p_i$. If $i \notin S$ it sets both $p_{G,t,0} = 0$ and $p_{G,t,1} = 0$. If $t = 1$ (and therefore $i \notin S$), it sets $R_{G,t,\ell} = x_i$, $p_{G,t,1} = p_i$ and $p_{G,t,0} = 1 - p_i$. The correctness of these settings is straightforward.

Generalization to arbitrary costs:

The algorithm FindSupp can easily be modified to handle arbitrary non-negative integer costs. The main difference is that t is used to represent the total cost of a set R of tests, rather than just the size of the set.

Consider a gate G of φ. If there is at least one subset $R \subseteq T(G)$ such that $t = \sum_{i \in R} c_i$, call t a feasible value for G.

For t a feasible value for G, let $R_{G,t,\ell}$ be the subset maximizing $\rho_R(S)$ (whose denominator is now $\sum_{i \in R} c_i$) subject to the constraints that $R \subseteq T(G)$ and $\sum_{i \in R} c_i = t$.

ComputeRp computes $R_{G,t,\ell}$ for all feasible values t for G. When ComputeRp computes $R_{G,t,\ell}$ for an AND or OR gate G, instead of trying all $j \in \{0, \ldots, |T(G')|\}$ where $t - j \in \{0, \ldots, |T(G'')|\}$, as it did in the unit cost case, ComputeRp tries all j such that j is feasible for G' and $t - j$ is feasible for G''. The other modifications in the algorithm are straightforward.

The running time of FindSupp is $O(n(\sum_{i\in V} c_i)^2)$. 2

The algorithm described in Lemma 10 can be used to form a 2-greedy chain $S_0 \subset S_1 \subset \ldots \subset S_k$, where each S_{j+1} is generated from S_j by running FindSupp with $S = S_j$ to produce R, and then setting $S_{j+1} = S_j \cup R$.

The theorem now follows immediately from Lemma 10 and Theorem 1.

Theorem 11 There is an 8-approximation algorithm solving the non-adaptive SBFE problem for read-once formulas. The running time of the algorithm depends on the costs c_i. It runs in polynomial-time in the unit cost case, and in pseudo-polynomial time when the costs are arbitrary positive integers.

5 Backward Greedy Algorithms

The greedy algorithm we presented for MSOP works by starting with the empty set of elements, and greedily adding subsets in each greedy step. An alternative greedy approach starts with the set V, and greedily removes subsets from V in each greedy step. A backward greedy approach was used by Iwata et al. (2012) in their work on some special cases of MSPP.

In this section, we describe a backward analog to our greedy algorithm for MSOP and its relationship to a *dual* MSOP problem, analogous to the dual problem introduced in Fokkink et al. (2019).

We call an F-chain $S = (S_j)_{j=0}^k$ a *backward α-greedy chain* if

$$\rho_{S_j}(S_{j-1}) \leq \alpha \min_{\{T \in F : T \subseteq S_j\}} \rho_{S_j}(T),$$

for all $j = 1, \ldots, k$.

A backward α-greedy chain can be seen to be equivalent to an α-greedy chain for the dual problem. To describe the dual problem, we write the cost function $C(f, g)$ in another, equivalent way. Fixing $F \subseteq 2^V$, we first define $F^\#$ as the family of complements of sets in F. That is, $F^\# = \{S \subseteq V : V \setminus S \in F\}$. Note that F is closed under intersection if and only if $F^\#$ is closed under union. There is a one-to-one correspondence between F-chains and $F^\#$-chains, obtained by mapping an element S of an F-chain to $V \setminus S$ and reversing the order. We refer to the corresponding $F^\#$-chain of a F-chain S as its *dual chain*, which we denote by $S^\#$. We also denote the *dual function* of f by $f^\# : F^\# \to \mathbb{R}$, given by $f^\#(S) = f(V) - f(V \setminus S)$, and similarly for g. Note that a set function is the dual of its dual, as is an F-chain, and that $f^\#$ and $g^\#$ are non-decreasing. Also, f is submodular if and only if $f^\#$ is supermodular and f is subadditive if and only if $f^\#$ is superadditive.

Given an MSOP with inputs f, g and F, the dual problem is an MSOP with inputs $g^{\#}$, $f^{\#}$ and $F^{\#}$. In other words, the dual problem is to minimize

$$C_{g^{\#},f^{\#}}(T) = \sum_{j=1}^{k} g^{\#}(T_j)(f^{\#}(T_j) - f^{\#}(T_{j-1})),$$

over all $F^{\#}$-chains $T = (T_j)_{j=0}^{k}$. Observe that an MSOP is the dual of its dual.

It is now easy to see that an F-chain $S = (S_j)_{j=0}^{n}$ is a backward α-greedy chain if and only if its dual chain is an α-greedy $F^{\#}$-chain for the dual problem.

The following is immediate and generalizes a similar observation from Fokkink et al. (2019).

Lemma 12 If S is an F-chain, then $C_{f,g}(S) = C_{g^{\#},f^{\#}}(S^{\#})$ and S is an α-approximation for an instance of MSOP if and only if $S^{\#}$ is an α-approximation for its dual.

Proof. To prove the first statement, we point out that the area A under the second histogram in the proof of Theorem 1, given by the sum in (3), is equal to $C_{g^{\#},f^{\#}}(S^{\#})$. This area is also shown to be equal to $C_{f,g}(S)$ later in the same proof. The second statement in the lemma follows directly from the first. 2

Applying Theorem 1 to the dual problem, we obtain the following theorem as a corollary.

Theorem 13 Suppose g is superadditive and F is closed under intersectionThen for any $\alpha \geq 1$, a backward α-greedy chain is a 4α-approximation for an optimal chain for MSOP.

We may also apply Corollary 2 to the dual problem to obtain an additional corollary.

Corollary 14 Suppose $F = 2^V$. If f is supermodular and g is modular then a backward 1-greedy chain can be found in polynomial time and there exists a polynomial-time 4-approximation algorithm for MSOP.

6 Future Work

We have created a general framework for min-sum ordering problems, and while Theorem 1 relies on very modest assumptions, it is only useful if the maximum density problem can be efficiently approximated. More work is needed in this area in order to further exploit our approximation result.

Particular problems of interest include Generalized Min Sum Set Cover (GMSSC), introduced in Azar et al. (2009). Unlike MSSC, where a hyperedge is “covered” the first time any of its vertices are chosen,in GMSSC, each hyperedge has its own “covering requirement”, which specifies how many of its vertices must be chosen before it is "covered". The objective is to minimize the sum of covering times, as in MSSC. The best known approximation algorithm for GMSSC is 12.4, due to Im et al. (2014). This could be improved if the associated maximum density problem could be approximated within a factor less than 3.13. Of course, by adding costs to vertices and weights to hyperedges, one could further generalize GMSSC, giving rise to a more general maximum density problem of interest.

Another problem that comes under our framework is the Unreliable Job Scheduling Problem (UJP), introduced in Agnetis et al. (2009). In the basic setting, a set of jobs with given rewards must be scheduled by a single machine to maximize the total expected reward. There is a probability of failure associated with each job when it is scheduled, and if failure occurs the machine cannot schedule any further jobs. The problem has a neat "index" solution. Natural generalizations of the problem would consider the possibility of AND- or OR-precedence constraints, and therefore their associated maximum density problems.

Acknowledgements

The authors would like to thank Andreas S. Schulz for helpful comments in improving the presentation of the paper. This material is based upon work supported by the National Science Foundation under Grant Numbers IIS-1909446 and IIS-1909335 and by the Alexander von Humboldt Foundation with funds from the German Federal Ministry of Education and Research (BMBF).

References

Agnetis, A., Detti, P., Pranzo, M., and Sodhi, M. S. (2009). Sequencing unreliable jobs on parallel machines. *Journal of Scheduling*, 12(1):45.

Allen, S. R., Hellerstein, L., Kletenik, D., and Ünlüyurt, T. (2017). Evaluation of monotone DNF formulas. *Algorithmica*, 77(3):661–685.

Alpern, S. and Lidbetter, T. (2013). Mining coal or finding terrorists: The expanding search paradigm. *Operations Research*, 61(2):265–279.

Ambühl, C. and Mastrolilli, M. (2009). Single machine precedence constrained scheduling is a vertex cover problem. *Algorithmica*, 53(4):488–503.

Averbakh, I. and Pereira, J. (2012). The flowtime network construction problem. *IIE Transactions*, 44(8):681–694.

Azar, Y., Gamzu, I., and Yin, X. (2009). Multiple intents re-ranking. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 669–678.

Bar-Noy, A., Bellare, M., Halldórsson, M. M., Shachnai, H., and Tamir, T. (1998). On chromatic sums and distributed resource allocation. *Information and Computation*, 140(2):183–202.

Bar-Noy, A. and Kortsarz, G. (1998). Minimum color sum of bipartite graphs. *Journal of Algorithms*, 28(2):339–365.

Chekuri, C. and Motwani, R. (1999). Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. *Disc. Appl. Math.*, 98(1):29–38.

Chudak, F. A. and Hochbaum, D. S. (1999). A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters*, 25(5):199–204.

Correa, J. R. and Schulz, A. S. (2005). Single-machine scheduling with precedence constraints. *Mathematics of Operations Research*, 30(4):1005–1021.

Deshpande, A., Hellerstein, L., and Kletenik, D. (2016). 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.

Erlebach, T., Kääb, V., and Möhring, R. H. (2003). Scheduling AND/OR-networks on identical parallel machines. In *International Workshop on Approximation and Online Algorithms*, volume 2909 of *LNCS*, pages 123–136. Springer.

Feige, U., Lovász, L., and Tetali, P. (2002). Approximating min-sum set cover. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, volume 2462 of *LNCS*, pages 94–107. Springer.

Feige, U., Lovász, L., and Tetali, P. (2004). Approximating min sum set cover. *Algorithmica*, 40(4):219–234.

Fokkink, R., Kikuta, K., and Ramsey, D. (2017). The search value of a set. *Annals of Oper. Res.*, 256(1):63–73.

Fokkink, R., Lidbetter, T., and Végh, L. (2019). On submodular search and machine scheduling. *Mathematics of Operations Research*, 44(4):1431–1449.

Furnas, G. W. and Zacks, J. (1994). Multitrees: enriching and reusing hierarchical structure. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 330–336.

Gillies, D. W. and Liu, J. W.-S. (1995). Scheduling tasks with and/or precedence constraints. *SIAM Journal on Computing*, 24(4):797–810.

Gkenosis, D., Grammel, N., Hellerstein, L., and Kletenik, D. (2018). The stochastic score classification problem. In *Proceedings of the 26th Annual European Symposium on Algorithms, ESA*, pages 36:1–36:14.

Greiner, R., Hayward, R., Jankowska, M., and Molloy, M. (2006). Finding optimal satisficing strategies for and-or trees. *Artificial Intelligence*, 170(1):19–58.

Happach, F. and Schulz, A. S. (2020a). Approximation algorithms and LP relaxations for scheduling problems related to min-sum set cover. *arXiv:2001.07011*.

Happach, F. and Schulz, A. S. (2020b). Precedence-constrained scheduling and min-sum set cover. In *Approximation and Online Algorithms*, volume 11926 of *LNCS*, pages 170–187. Springer.

Hermans, B., Leus, R., and Matuschke, J. (2019). Exact and approximation algorithms for the expanding search problem. *arXiv:1911.08959*.

Horn, W. (1972). Single-machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM Journal on Applied Mathematics*, 23(2):189–202.

Im, S., Sviridenko, M., and Van Der Zwaan, R. (2014). Preemptive and non-preemptive generalized min sum set cover. *Mathematical Programming*, 145(1-2):377–401.

Işık, G. and Ünlüyurt, T. (2013). Sequential testing of 3-level deep series-parallel systems. In *IEEE International Conference on Industrial Engineering and Engineering Management*.

Iwata, S., Tetali, P., and Tripathi, P. (2012). Approximating minimum linear ordering problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 206–217. Springer.

Johannes, B. (2005). On the complexity of scheduling unit-time jobs with OR-precedence constraints. *Operations Research Letters*, 33(6):587–596.

Kubicka, E. and Schwenk, A. J. (1989). An introduction to chromatic sums. In *Proceedings of the 17th conference on ACM Annual Computer Science Conference*, pages 39–45. ACM.

Lawler, E. L. (1978). Sequencing jobs to minimize total weighted completion time subject to precedence constraints. In *Annals of Discrete Mathematics*, volume 2, pages 75–90. Elsevier.

Margot, F., Queyranne, M., and Wang, Y. (2003). Decompositions, network flows and a precedence constrained single machine scheduling problem. *Oper. Res.*, 51(6):981–992.

McClintock, J., Mestre, J., and Wirth, A. (2017). Precedence-constrained min sum set cover. In *28th International Symposium on Algorithms and Computation*, volume 92 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:12.

Munagala, K., Babu, S., Motwani, R., and Widom, J. (2005). The pipelined set cover problem. In *International Conference on Database Theory*, volume 3363 of *LNCS*, pages 83–98. Springer.

Pisaruk, N. (1992). The boundaries of submodular functions. *Comp. Math. Math. Phys.*, 32(12):1769–1783.

Schulz, A. S. (1996). Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds. In *Proceedings of the 5th International Conference on Integer Programming and Combinatorial Optimization*, volume 1084 of *LNCS*, pages 301–315. Springer.

Sidney, J. (1975). Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Oper. Res.*, 23(2):283–298.

Streeter, M. J. and Golovin, D. (2008). An online algorithm for maximizing submodular functions. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1577–1584. Curran Associates, Inc.

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