Minimum Latency Submodular Cover

SUNGJIN IM, University of California, Merced
VISWANATH NAGARAJAN, University of Michigan
RUBEN VAN DER ZWAAN, Maastricht University

13

We study the Minimum Latency Submodular Cover (MLSC) problem, which consists of a metric (V, d) with source $r \in V$ and m monotone submodular functions $f_1, f_2, \ldots, f_m : 2^V \to [0, 1]$. The goal is to find a path originating at r that minimizes the total "cover time" of all functions. This generalizes well-studied problems, such as Submodular Ranking [Azar and Gamzu 2011] and the Group Steiner Tree [Garg et al. 2000]. We give a polynomial time $O(\log \frac{1}{\epsilon} \cdot \log^{2+\delta} |V|)$-approximation algorithm for MLSC, where $\epsilon > 0$ is the smallest non-zero marginal increase of any $\{f_i\}_{i=1}^m$ and $\delta > 0$ is any constant.

We also consider the Latency Covering Steiner Tree (LCST) problem, which is the special case of MLSC where the f_is are multi-coverage functions. This is a common generalization of the Latency Group Steiner Tree [Gupta et al. 2010; Chakrabarty and Swamy 2011] and Generalized Min-sum Set Cover [Azar et al. 2009; Bansal et al. 2010] problems. We obtain an $O(\log^2 |V|)$-approximation algorithm for LCST.

Finally, we study a natural stochastic extension of the Submodular Ranking problem and obtain an adaptive algorithm with an $O(\log 1/\epsilon)$-approximation ratio, which is best possible. This result also generalizes some previously studied stochastic optimization problems, such as Stochastic Set Cover [Goemans and Vondrák 2006] and Shared Filter Evaluation [Munagala et al. 2007; Liu et al. 2008].

CCS Concepts: • **Theory of computation → Approximation algorithms analysis**; **Scheduling algorithms**; **Packing and covering problems**

Additional Key Words and Phrases: Approximation, sequencing and scheduling, submodular, stochastic optimization, covering Steiner tree

ACM Reference Format:
Sungjin Im, Viswanath Nagarajan, and Ruben van der Zwaan. 2016. Minimum latency submodular cover. ACM Trans. Algorithms 13, 1, Article 13 (November 2016), 28 pages.
DOI: http://dx.doi.org/10.1145/2987751

1. INTRODUCTION

Ordering a set of elements to be simultaneously good for several valuations is an important issue in web-search ranking and broadcast scheduling. A formal model for Q1 this is the Multiple Intents Re-ranking problem [Azar et al. 2009]; this is also known as Generalized Min-Sum Set Cover [Bansal et al. 2010]. In this problem, a set of elements is to be displayed to m different users, each of whom wants to see some threshold number of elements from its particular subset of interest. The objective is to compute an ordering that minimizes the total overhead of the users, where the overhead corresponds to the position in the ordering when the user is satisfied.

S. Im was supported in part by NSF Grant No. CCF-1409130.
Q2 Authors' addresses: S. Im, Electrical Engineering and Computer Science, University of California, Merced, USA; V. Nagarajan, Department of Industrial and Operations Engineering, University of Michigan, USA; R. van der Zwaan, Maastricht University, The Netherlands.

DOI: http://dx.doi.org/10.1145/2987751

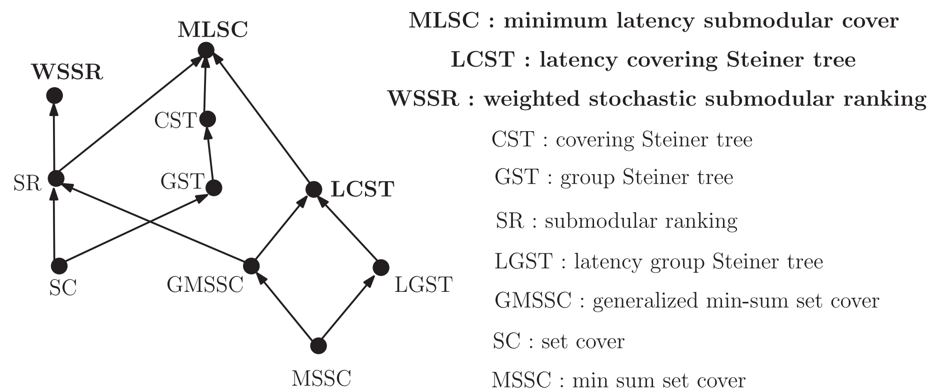

Fig. 1. An arrow from X to Y means X is a special case of Y.

A more general model that has been studied is the Submodular Ranking problem [Azar and Gamzu 2011], where the interests of users are represented by arbitrary (monotone) submodular functions. Again, the objective is to order the elements to minimize the total overhead, where now the overhead of a user is the position when its utility function is "covered."

In this article, we extend both of these models to the setting of metric switching costs between elements. This allows us to handle additional issues such as:

- *Data locality:* It takes $d(i, j)$ time to read or transmit data j after data i.
- *Context switching:* It takes $d(i, j)$ time for a user to parse data j when scheduled after data i.

From a theoretical point of view, these problems generalize a number of previously studied problems, and our results unify/extend techniques used in different settings.

We introduce and study the Minimum Latency Submodular Cover (MLSC) problem, which is the metric version of Submodular Ranking [Azar and Gamzu 2011] and its interesting special case, the Latency Covering Steiner Tree (LCST) problem, which extends Generalized Min-Sum Set Cover [Azar et al. 2009; Bansal et al. 2010]. The formal definitions follow shortly, in the next subsection. We obtain poly-logarithmic approximation guarantees for both problems. We remark that, due to a relation to the well-known Group Steiner Tree problem [Garg et al. 2000], any significant improvement on our results would lead to a similar improvement for the Group Steiner Tree. The MLSC problem is a common generalization of several previously studied problems [Garg et al. 2000; Konjevod et al. 2002; Feige et al. 2004; Gupta et al. 2010; Chakrabarty and Swamy 2011; Azar et al. 2009; Azar and Gamzu 2011]; see also Figure 1.

In a somewhat different direction, we also study the Weighted Stochastic Submodular Ranking problem, where elements are stochastic and the goal is to adaptively schedule elements to minimize the expected total cover time. We obtain an $O(\log \frac{1}{\epsilon})$-approximation algorithm for this problem, which is known to be best possible even in the deterministic setting [Azar and Gamzu 2011]. This result also generalizes many previously studied stochastic optimization problems [Goemans and Vondrák 2006; Munagala et al. 2007; Liu et al. 2008].

1.1. Problem Definitions

We now give formal definitions of the problems considered in this article. The problems followed by * are those for which we obtain the first non-trivial results; these are also shown in bold font in Figure 1. Several other problems are discussed below since those are important special cases of our main problems. The relationships between these problems are also shown pictorially in Figure 1.

A function $f : 2^V \to \mathbb{R}_+$ is *submodular* if, for any $A, B \subseteq V$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, and it is *monotone* if for any $A \subseteq B$, $f(A) \leq f(B)$. We assume some familiarity with submodular functions [Schrijver 2003].

Minimum Latency Submodular Cover*. There is a ground set V of elements/vertices and $d : \binom{V}{2} \to \mathbb{R}_+$ is a distance function. We assume that d is symmetric and satisfies the triangle inequality. In addition, there is a specified root vertex $r \in V$. There are m monotone submodular functions $f_1, \ldots, f_m : 2^V \to \mathbb{R}_+$ representing the valuations of different users. We assume, without loss of generality by truncation, that $f_i(V) = 1$ for all $i \in [m]$.[1] Function f_i is said to be *covered* (or satisfied) by set $S \subseteq V$ if $f_i(S) = 1 = f_i(V)$. The *cover time* of function f_i in a path π is the length of the shortest prefix of π that has f_i value one, that is,

$$\min \{t \;:\; f_i (\{v \in V : v \text{ appears within distance } t \text{ on } \pi\}) = 1\}.$$

The objective in the Minimum Latency Submodular Cover problem is to compute a path originating at r that minimizes the sum of cover times of all functions. A technical parameter that we use to measure performance (which also appears in Azar and Gamzu [2011] and Wolsey [1982]) is ϵ, which is defined to be the smallest non-zero marginal increase of any function $\{f_i\}_{i=1}^m$.

Generalized Min-Sum Set Cover (GMSSC) [Azar et al. 2009; Bansal et al. 2010]. Given a ground set V and m subsets $\{g_i \subseteq V\}_{i=1}^m$ with respective requirements $\{k_i\}_{i=1}^m$, the goal is to find a linear ordering of V that minimizes the sum of cover times. A subset g_i is said to be covered when at least k_i elements from g_i have appeared. Min-Sum Set Cover (MSSC) is the special case when $\max_i k_i = 1$.

Submodular Ranking (SR) [Azar and Gamzu 2011]. Given a ground set V and m monotone submodular functions $f_1, \ldots, f_m : 2^V \to \mathbb{R}_+$, the goal is to compute a linear ordering of V that minimizes the sum of cover times of all functions. The cover time of a function here is the minimum number of elements in a prefix that has function value at least 1. This is a special case of MLSC when metric d is uniform. The set cover problem is a special case of SR when there is a single submodular function (which is also a coverage function). GMSSC is another special case of SR, where each subset g_i corresponds to the submodular function $f_i(S) = \min\{|g_i \cap S|/k_i, 1\}$.

Group Steiner Tree (GST) [Garg et al. 2000]. Given a metric (V, d) with root $r \in V$ and N groups of vertices $\{g_i \subseteq V\}_{i=1}^N$, the goal is to find a minimum length tree containing r and at least one vertex from each of the N groups. Observe that an r-rooted tree can be converted into a path starting from r with at most a factor two loss in the total length and vice versa. Thus GST is a special case of MLSC when there is only a single submodular function,

$$f_1(S) = \frac{1}{N} \sum_{i=1}^{N} \min\{|g_i \cap S|,\ 1\}.$$

Note that $f_1(S') = 1$ if and only if $S' \bigcap g_i$ is nonempty for all $i \in [N]$.

Covering Steiner Tree (CST) [Konjevod et al. 2002; Gupta and Srinivasan 2006]. This is a generalization of GST with the same input as above, where each

[1]Throughout the article, for any integer $\ell \geq 1$, we denote $[\ell] := \{1, 2, \ldots, \ell\}$.

group g_i is also associated with a requirement k_i. The goal here is to find a minimum length tree that contains r and at least k_i vertices from group g_i, for all $i \in [N]$. We recover CST as a special case of MLSC by setting

$$f_1(S) = \frac{1}{N}\sum_{i=1}^{N} \min\left\{\frac{|g_i \cap S|}{k_i},\, 1\right\}.$$

Note that now $f_1(S') = 1$ if and only if $|S' \bigcap g_i| \geq k_i$ for all $i \in [N]$.

Latency Group Steiner Tree (LGST) [Gupta et al. 2010; Chakrabarty and Swamy 2011]. This is a variant of the group Steiner tree problem. Given a metric (V, d) with root r and N groups of vertices $\{g_i \subseteq V\}_{i=1}^{N}$, the goal is to find a path π originating from r that minimizes the sum of cover times of the groups. (A group g_i is covered at the shortest prefix of π that contains at least one vertex from g_i.) Note that MSSC is the special case when the metric is uniform.

Latency Covering Steiner Tree*. The input to this problem is the same as for LGST with additional requirements $\{k_i\}_{i=1}^{N}$ corresponding to each group. The objective is again a path π originating from r that minimizes the sum of cover times, where group g_i is covered at the shortest prefix of π that contains at least k_i vertices from g_i. Clearly, LGST is the special case of LCST where all requirements $k_i = 1$. GMSSC is also a special case when the metric is uniform. We obtain LCST as a special case of MLSC with $m = N$ functions and $f_i(S) = \min\{|g_i \cap S|/k_i,\, 1\}$ for all $i \in [N]$.

Weighted Stochastic Submodular Ranking* (WSSR). This is a stochastic generalization of the Submodular Ranking problem. We are given a set V of stochastic elements (random variables), each having an independent distribution over a certain domain Δ. The submodular functions are also defined on the ground set Δ, that is, $f_1, \ldots, f_m : 2^{\Delta} \to [0, 1]$. In addition, each element $i \in V$ has a deterministic time ℓ_i to be scheduled. The realization (from Δ) of any element is known immediately after scheduling it. The goal is to find an adaptive ordering of V that minimizes the total expected cover time of the m functions. Since elements are stochastic, it is possible that a function is never covered: In such cases, we just fix the cover time to be $\sum_{i \in V} \ell_i$ (which is the total duration of any schedule).

We will be concerned with *adaptive* algorithms. Such an algorithm is allowed to decide the next element to schedule based on the instantiations of the previously scheduled elements. This models the setting where the algorithm can benefit from user feedback.

WSSR generalizes the Stochastic Set Cover studied in Goemans and Vondrák [2006]. Interestingly, it also captures some variants of Stochastic Set Cover that have applications in query processing with probabilistic information [Munagala et al. 2007; Liu et al. 2008]. Various applications of WSSR are discussed in more detail in Section 5.

1.2. Our Results and Techniques

Our first result is on the MLSC problem.

THEOREM 1.1. *For any constant $\delta > 0$, there is an $O(\log \frac{1}{\epsilon} \cdot \log^{2+\delta} |V|)$-approximation algorithm for the Minimum Latency Submodular Cover problem. Here $\epsilon > 0$ is a value such that. for any $i \in [m]$ and $S' \subseteq S$, if $f(S) > f(S')$, then $f(S) \geq f(S') + \epsilon$.*

Note that in the special case of the Group Steiner Tree, this result is larger only by a factor of $O(\log^{\delta} |V|)$ than its best-known approximation ratio of $O(\log N \log^2 |V|)$, due to Garg et al. [2000]. Our algorithm uses the framework of Azar and Gamzu [2011] and the Submodular Orienteering problem (SOP) [Chekuri and Pál 2005] as a sub-routine. The input to SOP consists of metric (V, d), root r, monotone submodular function $f : 2^V \to \mathbb{R}_+$, and length bound B. The goal is to find a path originating at r having length at most B that maximizes $f(S)$, where $S \subseteq V$ is the set of vertices visited in the path. Specifically, we show that a (ρ, σ)-bicriteria approximation algorithm[2] for SOP can be used to obtain an $O(\rho\sigma \cdot \log \frac{1}{\epsilon})$-approximation algorithm for MLSC. To obtain Theorem 1.1, we use an $(O(1),\ O(\log^{2+\delta}|V|))$-bicriteria approximation for SOP that follows from Calinescu and Zelikovsky [2005] and Chekuri et al. [2006].

Our algorithm for MLSC is an extension of the elegant "adaptive residual updates scheme" of Azar and Gamzu [2011] for Submodular Ranking (i.e., uniform metric MLSC). As shown in Azar and Gamzu [2011], an interesting aspect of this problem is that the natural greedy algorithm, based on absolute contribution of elements, performs very poorly. Instead, they used a modified greedy algorithm that selects one element at a time according to residual coverage. In the MLSC setting of general metrics, our algorithm uses a similar residual coverage *function* to repeatedly augment the solution. However our augmentations are paths of geometrically increasing lengths, instead of just one element. A crucial point in our algorithm is that the residual coverage functions are always submodular, and hence we can use SOP in the augmentation step.

We note that the approach of covering the maximum number of functions within geometrically increasing lengths fails because the residual coverage function here is non-submodular; in fact, as noted in Bansal et al. [2010], this subproblem contains the difficult dense-*k*-subgraph problem even for the special case of Generalized Min-Sum Set Cover with requirement two. We also note that the choice of our (submodular) residual coverage function ultimately draws on the Submodular Ranking algorithm [Azar and Gamzu 2011].

The analysis in Azar and Gamzu [2011] was based on viewing the optimal and approximate solutions as histograms. This approach was first used in this line of work by Feige et al. [2004] for the Min-Sum Set Cover problem (see also Bar-Noy et al. [1998]). This was also the main framework of analysis in Azar et al. [2009] for Generalized Min-Sum Set Cover and then for Submodular Ranking [Azar and Gamzu 2011]. However, these proofs have been increasingly difficult, as the problem in consideration adds more generality. Instead, we follow a different and more direct approach that is similar to the analysis of the Minimum Latency problem, see, for example, Chaudhuri et al. [2003] and Fakcharoenphol et al. [2007]. In fact, the proof of Theorem 1.1 is enabled by a new simpler analysis of the Submodular Ranking algorithm [Azar and Gamzu 2011].

Our second result is a better approximation ratio for the LCST problem. Note that Theorem 1.1 implies directly an $O(\log k_{max} \cdot \log^{2+\delta}|V|)$-approximation algorithm for LCST, where $k_{max} = \max_{i=1}^{N} k_i$.

THEOREM 1.2. *There is an $O(\log^2|V|)$-approximation algorithm for Latency Covering Steiner Tree.*

The main idea in this result is a new LP relaxation for Covering Steiner Tree (using *Knapsack Cover* type inequalities [Carr et al. 2000]) having a poly-logarithmic integrality gap. This new LP might also be of some independent interest. The previous algorithms [Konjevod et al. 2002; Gupta and Srinivasan 2006] for covering Steiner tree were based on iteratively solving an LP with large integrality gap. However, the previous approach does not seem suitable to the *latency* version we consider. Our new LP relaxation for Covering Steiner Tree (CST) is crucial for obtaining the approximation

[2]Given any instance of SOP, such an algorithm returns a path of length at most $\sigma \cdot B$ and function value at least OPT$/\rho$.

stated in Theorem 1.2. Given this new LP and rounding algorithm for CST, we obtain the LCST algorithm using a time-indexed LP relaxation, which is a direct extension of a similar LP for the LGST in Chakrabarty and Swamy [2011]. Furthermore, as shown in Nagarajan [2009] and Chakrabarty and Swamy [2011], any improvement over Theorem 1.2, even in the $k_{max} = 1$ special case (i.e., LGST), would yield an improved approximation ratio for the Group Steiner Tree, which is a long-standing open question.

Our final result is for the Weighted Stochastic Submodular Ranking problem. As shown in Goemans and Vondrák [2006] and Golovin and Krause [2010], even special cases of this problem have a polynomially large adaptivity gap (ratio between the optimal non-adaptive and adaptive solutions).[3] This motivates adaptive algorithms, and we obtain the following result in Section 5.

THEOREM 1.3. *There is an adaptive $O(\log \frac{1}{\epsilon})$-approximation algorithm for the Weighted Stochastic Submodular Ranking problem.*

In particular, we show that the natural stochastic extension of the algorithm from Azar and Gamzu [2011] achieves this approximation factor. We remark that the analysis in Azar and Gamzu [2011] of deterministic submodular ranking required unit costs, whereas Theorem 1.3 holds for the stochastic setting even with non-uniform costs $\{\ell_i\}$.

As mentioned, our results generalize the results in Goemans and Vondrák [2006], Munagala et al. [2007], and Liu et al. [2008] that study (some variants of) Stochastic Set Cover. Our analysis is arguably simpler and more transparent than that in Liu et al. [2008], which gave the first tight analysis of these problems. We note that Liu et al. [2008] used an intricate charging scheme with "dual prices," and it does not seem directly applicable to general submodular functions.

We note that our techniques do not extend directly to the stochastic MLSC problem (on general metrics), and obtaining a poly-logarithmic approximation here seems to require additional ideas.

1.3. Previous Work

The first poly-logarithmic approximation for the Group Steiner Tree was $O(\log N \log^2 |V|)$, obtained by Garg et al. [2000]. This is still the best-known bound. Chekuri et al. [2006] gave a combinatorial algorithm that achieved a slightly weaker approximation ratio (the algorithm in Garg et al. [2000] was LP based). This combinatorial approach was extended in Calinescu and Zelikovsky [2005] to the problem of covering any submodular function in a metric space. We use this algorithm in the SOP subroutine for our MLSC result. For SOP an $O(\log |V|)$ approximation is known from Chekuri and Pál [2005] but with a *quasi-polynomial* running time. We note that an $\Omega(\log^{2-\delta} |V|)$ hardness of approximation is known for the Group Steiner Tree (even on tree metrics) from Halperin and Krauthgamer [2003].

The Covering Steiner Tree problem was introduced by Konjevod et al. [2002], which can be viewed as the multicover version of the Group Steiner Tree. They gave an $O(\log(Nk_{max}) \log^2 |V|)$ approximation using an LP relaxation. However, the LP used in Konjevod et al. [2002] has a large $\Omega(k_{max})$ integrality gap; they got around this issue by iteratively solving a suitable sequence of LPs. They also extended the randomized rounding analysis from Garg et al. [2000] to this context. Later, Gupta and Srinivasan [2006] improved the approximation bound to $O(\log N \log^2 |V|)$, removing the dependence on the covering requirements. This algorithm was also based on solving

[3]A non-adaptive solution is just a fixed linear ordering of the elements, whereas an adaptive solution can select the next element based on previous instantiations.

a similar sequence of LPs; the improvement was due to a combination of threshold rounding and randomized rounding. In this article, we give a stronger LP relaxation for the Covering Steiner Tree based on so-called Knapsack-Covering inequalities (abbreviated to KC inequalities), which has an $O(\log N \log^2 |V|)$ integrality gap.

The Stochastic Set Cover problem (which is a special case of Weighted Stochastic Submodular Ranking) was introduced by Goemans and Vondrák [2006]. Here each set covers a random subset of items, and the goal is to minimize the expected cost of a set cover. Goemans and Vondrák [2006] showed a large adaptivity gap for Stochastic Set Cover, and gave a logarithmic approximation for a relaxed version where each stochastic set can be added multiple times. A related problem in the context of fast query evaluation was studied in Munagala et al. [2007], where the authors gave a triple logarithmic approximation. This bound was improved to the best-possible logarithmic ratio by Liu et al. [2008]; this result was also applicable to stochastic set cover (where each set can be added at most once). Another related article is Golovin and Krause [2010], where the authors defined a general property "adaptive submodularity" and showed nearly optimal approximation guarantees for several objectives (max coverage, min-cost cover, and min-sum cover). The most relevant result in Golovin and Krause [2010] to WSSR is the 4-approximation for Stochastic Min Sum Set Cover. This approach required a *fixed* submodular function f such that the objective is to minimize $\mathbb{E}[\sum_{t\geq 0} f(\overline{V}) - f(\overline{\pi}_t)]$, where $\overline{\pi}_t$ is the realization of elements scheduled within time t and $\overline{V}$ denotes the realization of all elements. However, this assumption is not satisfied even for the special case of Generalized Min-Sum Set Cover with requirements 2. So an extension of Golovin and Krause [2010] to our setting is unclear. Recently, Guillory and Bilmes [2011] studied the Submodular Ranking problem in an online regret setting, which differs from the adaptive model we consider.

1.4. Organization

In Section 2 we revisit the Submodular Ranking problem and give an easier and perhaps more intuitive analysis of the algorithm from Azar and Gamzu [2011]. This simpler analysis is then used in the algorithms for Minimum Latency Submodular Cover (Theorem 1.1) and Weighted Stochastic Submodular Ranking (Theorem 1.3), which appear in Sections 3 and 5, respectively. Section 4 contains the improved approximation algorithm for Latency Covering Steiner Tree (Theorem 1.2), which makes use of a new linear programming relaxation for Covering Steiner Tree. The section on LCST can be read independently of the other three sections.

2. SIMPLER ANALYSIS OF THE SUBMODULAR RANKING ALGORITHM

In this section, we revisit the Submodular Ranking problem [Azar and Gamzu 2011]. Recall that the input consists of a ground set $V := [n]$ of elements and monotone submodular functions $f_1, f_2, \ldots, f_m : 2^{[n]} \to [0, 1]$ with $f_i(V) = 1$, $\forall i \in [m]$. The goal is to find a complete linear ordering of the elements that minimizes the total cover time of all functions. The cover time $\mathtt{cov}(f_i)$ of f_i is defined as the smallest index t such that the function f_i has value 1 for the first t elements in the ordering. We also say that an element e is scheduled at time t if it is the tth element in the ordering. It is assumed that each function f_i satisfies the following property: For any $S \supseteq S'$, if $f_i(S) - f_i(S') > 0$, then it must be the case that $f_i(S) - f_i(S') \geq \epsilon$, where $\epsilon > 0$ is a value that is uniform for all functions f_i. This is a useful parameter in describing the performance guarantee.

Azar and Gamzu [2011] gave a modified greedy-style algorithm with an approximation factor of $O(\log \frac{1}{\epsilon})$ for Submodular Ranking. Their analysis was fairly involved. In this section, we give an alternate shorter proof of their result. Our analysis also extends to the more general MLSC problem which we study in the next section. The algorithm ALG-AG from Azar and Gamzu [2011] is given below. In the output, $\pi(t)$ denotes the element that appears in the tth time slot.

ALGORITHM 1: Algorithm for Submodular Ranking (ALG-AG).

INPUT: Ground set $[n]$; monotone submodular functions $f_i : 2^{[n]} \rightarrow [0,1], i \in [m]$

```
S ← ∅
for t = 1 to n do
    Let f^S(e) := Σ_{i∈[m], f_i(S)<1} (f_i(S∪{e}) − f_i(S)) / (1 − f_i(S))
    e = arg max_{e∈[n]\S} f^S(e)
    S ← S ⋃ {e}
    π(t) ← e
end for
```

OUTPUT: A linear ordering $\langle \pi(1), \pi(2), \ldots, \pi(n) \rangle$ of $[n]$.

Theorem 2.1 (Azar and Gamzu [2011]). *ALG-AG is an $O(\ln(\frac{1}{\epsilon}))$-approximation algorithm for Submodular Ranking.*

Let $\alpha := 1+\ln(\frac{1}{\epsilon})$. To simplify notation, without loss of generality, we assume that α is an integer. Let $R(t)$ denote the set of functions that are *not satisfied* by ALG-AG earlier than time t; $R(t)$ includes the functions that are satisfied exactly at time t. For notational convenience, we use $i \in R(t)$ interchangeably with $f_i \in R(t)$. Analogously, $R^*(t)$ is the set of functions that are not satisfied in the optimal solution before time t. Note that algorithm's objective $\mathsf{ALG} = \sum_{t\geq 1} |R(t)|$ and the optimal value $\mathsf{OPT} = \sum_{t\geq 1} |R^*(t)|$. We will be interested in the number of unsatisfied functions at times $\{8\alpha 2^j : j \in \mathbb{Z}_+\}$ by ALG-AG and the number of unsatisfied functions at times $\{2^j : j \in \mathbb{Z}_+\}$ by the optimal solution. Let $R_j := R(8\alpha 2^j)$ and $R^*_j = R^*(2^j)$ for all integer $j \geq 0$. It is important to note that R_j and R^*_j are concerned with different times. For notational simplicity, we let $R_{-1} := \emptyset$.

We show the following key lemma. Roughly speaking, it says that the number of unsatisfied functions by ALG-AG diminishes quickly unless it is comparable to the number of unsatisfied functions in OPT.

Lemma 2.2. *For any $j \geq 0$, we have $|R_j| \leq \frac{1}{4}|R_{j-1}| + |R^*_j|$.*

Proof. When $j = 0$ the lemma holds trivially. Now consider any integer $j \geq 1$ and time step $t \in [8\alpha 2^{j-1}, 8\alpha 2^j)$. Let S_{t-1} denote the set of elements that ALG-AG schedules before time t and let e_t denote the element that ALG-AG schedules exactly at time t. Let E_j denote the set of elements that ALG-AG schedules until time $8\alpha 2^j$. Let E^*_j denote the set of elements that OPT schedules until time 2^j. Recall that ALG-AG picks e_t as an element e that maximizes

$$f^{S_{t-1}}(e) := \sum_{i\in[m]: f_i(S_{t-1})<1} \frac{f_i(S_{t-1}\cup\{e\}) - f_i(S_{t-1})}{1 - f_i(S_{t-1})}.$$

This leads us to the following proposition.

Proposition 2.3. *For any $j \geq 1$, time step $t \in [8\alpha 2^{j-1}, 8\alpha 2^j)$ and $e \in E^*_j$, we have $f^{S_{t-1}}(e_t) \geq f^{S_{t-1}}(e)$.*

Proof. Since ALG-AG has chosen to schedule element e_t over all elements $e \in E_j^* \backslash S_{t-1}$, we know that the claimed inequality holds for any $e \in E_j^* \backslash S_{t-1}$. Further, the inequality holds for any element e in S_{t-1}, since $f^{S_{t-1}}(e) = 0$ for such an element e. □

By taking an average over all elements in E_j^*, we derive

$$f^{S_{t-1}}(e_t) \geq \frac{1}{|E_j^*|} \sum_{e \in E_j^*} f^{S_{t-1}}(e)$$

$$\geq \frac{1}{|E_j^*|} \sum_{e \in E_j^*} \sum_{i \in R_j \backslash R_j^*} \frac{f_i(S_{t-1} \cup \{e\}) - f_i(S_{t-1})}{1 - f_i(S_{t-1})}. \quad (1)$$

Observe that in Equation (1), the inner summation only involves functions f_i for which $f_i(S_{t-1}) < 1$. This is because for any $i \in R_j$, function f_i is not covered before time $8\alpha 2^j$ and $t < 8\alpha 2^j$. Due to submodularity of each function f_i, we have that

$$(1) \;\geq\; \frac{1}{|E_j^*|} \sum_{i \in R_j \backslash R_j^*} \frac{f_i(S_{t-1} \cup E_j^*) - f_i(S_{t-1})}{1 - f_i(S_{t-1})} \;=\; \frac{1}{|E_j^*|} \sum_{i \in R_j \backslash R_j^*} 1 \;\geq\; \frac{|R_j| - |R_j^*|}{|E_j^*|}.$$

The equality is due to the fact that for any $i \notin R_j^*$, $f_i(E_j^*) = 1$ and each function f_i is monotone. Hence:

$$\sum_{8\alpha \cdot 2^{j-1} \leq t < 8\alpha \cdot 2^j} f^{S_{t-1}}(e_t) \;\geq\; \frac{8\alpha(2^j - 2^{j-1})}{|E_j^*|}(|R_j| - |R_j^*|) \;=\; 4\alpha(|R_j| - |R_j^*|), \quad (2)$$

where we used $|E_j^*| = 2^j$. We now upper bound the left-hand side of Equation (2). To this end, we need the following claim from Azar and Gamzu [2011].

Claim 2.4 (Claim 2.3 in Azar and Gamzu [2011]). *Given a monotone function* $f : 2^{[n]} \rightarrow [0, 1]$ *with* $f([n]) = 1$ *and sets* $\emptyset = S_0 \subseteq S_1 \subseteq \cdots \subseteq S_\ell \subseteq [n]$, *we have (using the convention* $0/0 = 0$*)*

$$\sum_{k=1}^{\ell} \frac{f(S_k) - f(S_{k-1})}{1 - f(S_{k-1})} \;\leq\; 1 + \ln \frac{1}{\delta}.$$

Here $\delta > 0$ *is such that for any* $A \subseteq B$, *if* $f(B) - f(A) > 0$, *then* $f(B) - f(A) \geq \delta$.

Proof. We give a proof for completeness. We can assume, without loss of generality, that $S_\ell = [n]$. Order the values in the set $\{f(S_k) \mid 0 \leq k \leq \ell\} \backslash \{1\}$ in increasing order to obtain $\beta_0 < \beta_1 < \ldots < \beta_H$. By the assumption, we have $\beta_0 \geq 0$ and $\beta_H \leq 1 - \delta$ (moreover, $\beta_h - \beta_{h-1} \geq \delta$, $\forall h \in [H]$). We will show that

$$\sum_{h=1}^{H} \frac{\beta_h - \beta_{h-1}}{1 - \beta_{h-1}} \;\leq\; \ln \frac{1}{\delta}.$$

Since $f(S_\ell) = 1$, the summation we want to bound has an additional term of $\frac{1-\beta_H}{1-\beta_H} = 1$.

Knowing that the function $u(x) = \frac{1}{1-x}$ is increasing for $x \in [0, 1)$, we derive

$$\sum_{h=1}^{H} \frac{\beta_h - \beta_{h-1}}{1 - \beta_{h-1}} \;=\; \sum_{h=1}^{H} \int_{x=\beta_{h-1}}^{\beta_h} \frac{1}{1 - \beta_{h-1}} \,\mathrm{d}x \;\leq\; \sum_{h=1}^{H} \int_{x=\beta_{h-1}}^{\beta_h} \frac{1}{1 - x} \,\mathrm{d}x \;=\; \int_{x=0}^{\beta_H} \frac{1}{1 - x} \,\mathrm{d}x$$

$$= \ln\left(\frac{1 - \beta_0}{1 - \beta_H}\right) \;\leq\; \ln \frac{1}{\delta}.$$

This proves the claim. □

Note that any function f_i not in R_{j-1} does not contribute to the left-hand side of Equation (2) since any such function f_i was already covered before time $8\alpha\, 2^{j-1} \leq t$. Further, knowing by Claim 2.4 that each function $f_i \in R_{j-1}$ can add at most $\alpha := 1 + \ln \frac{1}{\epsilon}$, we can upper bound the left-hand side of Equation (2) by $\alpha |R_{j-1}|$. Formally,

$$\begin{aligned}\sum_{8\alpha\cdot 2^{j-1} < t \leq 8\alpha\cdot 2^j} f^{S_{t-1}}(e_t) &= \sum_{8\alpha\cdot 2^{j-1} < t \leq 8\alpha\cdot 2^j} \; \sum_{i\in R_{j-1}: f_i(S_{t-1})<1} \frac{f_i(S_{t-1}\cup\{e_t\}) - f_i(S_{t-1})}{1 - f_i(S_{t-1})} \\ &\leq \sum_{i\in R_{j-1}} \; \sum_{t\geq 1: f_i(S_{t-1})<1} \frac{f_i(S_{t-1}\cup\{e_t\}) - f_i(S_{t-1})}{1 - f_i(S_{t-1})} \\ &\leq \alpha |R_{j-1}|. \end{aligned} \tag{3}$$

From Equations (2) and (3) we obtain $4\alpha(|R_j| - |R_j^*|) \leq \alpha |R_{j-1}|$, which completes the proof of Lemma 2.2.

Now we can prove Theorem 2.1 using Lemma 2.2.

PROOF OF THEOREM 2.1.

$$\begin{aligned}\mathsf{ALG} &= \sum_{j\geq 0} \; \sum_{8\alpha 2^j \leq t < 8\alpha 2^{j+1}} |R(t)| \quad + \sum_{1\leq t<8\alpha} |R(t)| \\ &\leq \sum_{j\geq 0} 8\alpha(2^{j+1} - 2^j)|R_j| \,+\, 8\alpha\mathsf{OPT} \text{ [Since } |R(t)| \text{ is non-increasing, and } |R(1)| \leq m \leq \mathsf{OPT}] \\ &= 8\alpha \sum_{j\geq 0} 2^{j+1}\left(|R_j| - \frac{1}{4}|R_{j-1}|\right) \quad + \quad 8\alpha\mathsf{OPT} \qquad [\text{Using } R_{-1} = \emptyset] \\ &\leq 8\alpha \sum_{j\geq 0} 2^{j+1}|R_j^*| \quad + \quad 8\alpha\mathsf{OPT} \qquad [\text{By Lemma 2.2}] \\ &\leq 8\alpha \sum_{j\geq 1} 4 \sum_{2^{j-1}\leq t<2^j} |R^*(t)| \,+\, 16\alpha|R_0^*| + 8\alpha\mathsf{OPT} \quad [\text{Since } |R^*(t)| \text{ is non-increasing}] \\ &\leq 32\alpha\mathsf{OPT} + 24\alpha\mathsf{OPT}.\end{aligned}$$

Thus we obtain $\mathsf{ALG} \leq 56\alpha\, \mathsf{OPT}$, which proves Theorem 2.1. □

3. MINIMUM LATENCY SUBMODULAR COVER

Recall that in the MLSC problem, we are given a metric (V, d) with root $r \in V$ and m monotone submodular functions $f_1, f_2, \ldots, f_m : 2^V \to [0, 1]$. Without loss of generality, by scaling, we assume that all distances $d(\cdot,\cdot)$ are integers. The objective in MLSC is to find a path starting at r that minimizes the total cover time of all functions.

As mentioned earlier, our algorithm for MLSC uses as a subroutine an algorithm for the SOP. In this problem, given metric (V, d), root r, monotone submodular function $g : 2^V \to \mathbb{R}_+$ and bound B, the goal is to compute a path P originating at r that has length at most B and maximizes $g(V(P))$ where $V(P)$ is the set of vertices covered by P. We assume that we have a (ρ, σ)-bicriteria approximation algorithm ALG-SOP for SOP. That is, on any SOP instance, ALG-SOP returns a path P of length at most $\sigma \cdot B$ and $g(V(P)) \geq \mathsf{OPT}/\rho$, where OPT is the optimal value obtained by any length B path. We recall the following known results on SOP.

THEOREM 3.1 (CALINESCU AND ZELIKOVSKY [2005]). *For any constant $\delta > 0$ there is a polynomial time $(O(1), O(\log^{2+\delta}|V|))$ bicriteria approximation algorithm for the Submodular Orienteering problem.*

THEOREM 3.2 (CHEKURI AND PÁL [2005]). *There is a quasi-polynomial time* $O(\log |V|)$ *approximation algorithm for the Submodular Orienteering problem.*

We note that Theorem 3.1 is implicit in Calinescu and Zelikovsky [2005]; for completeness, we provide additional detail in Appendix A.

We describe below our algorithm ALG-MLSC for MLSC. Here $\alpha = 1 + \ln \frac{1}{\epsilon}$. Note the difference from the Submodular Ranking algorithm [Azar and Gamzu 2011]: Here each augmentation is a path possibly covering several vertices. Despite the similarity of ALG-MLSC to the min-latency TSP-type algorithms [Chaudhuri et al. 2003; Fakcharoenphol et al. 2007], an important difference is that we *do not* try to directly maximize the number of covered functions in each augmentation: As noted, this subproblem is at least as hard as dense-k-subgraph, for which the best approximation ratio known is only polynomial [Bhaskara et al. 2010]. Instead, we maximize in each step some proxy residual coverage function f^S that suffices to eventually cover all functions quickly. This function is a natural extension of the single-element coverage values used in ALG-AG [Azar and Gamzu 2011]. It is important to note that in Line (4), $f^S(\cdot)$ is defined adaptively based on the current set S of visited vertices in each iteration. Moreover, since each function f_i is monotone and submodular, so is f^S for any $S \subseteq V$. In Step 6, $\pi \cdot P$ denotes the concatenation of paths π and P.

ALGORITHM 2: Algorithm for Min-Latency Submodular Cover (ALG-MLSC).

INPUT: $(V, d), r \in V; \{f_i : 2^V \to [0, 1]\}_{i=1}^m$.

1: $S \leftarrow \emptyset, \pi \leftarrow \emptyset$.
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: **for** $u = 1, 2, \ldots, 4\alpha\rho$ **do**
4: Define the submodular function

$$f^S(T) := \sum_{i \in [m],\, f_i(S) < 1} \frac{f_i(S \cup T) - f_i(S)}{1 - f_i(S)}, \quad \text{for all } T \subseteq V.$$

5: Use ALG-SOP to approximately solve the SOP instance on metric (V, d) with root r, submodular function f^S and length bound 2^k. Let P denote the solution path; note $d(P) \leq \sigma \cdot 2^k$.
6: $S \leftarrow S \cup V(P)$ and $\pi \leftarrow \pi \cdot P$.
7: **end for**
8: **end for**

OUTPUT: Output solution π.

We prove the following theorem, which implies Theorem 1.1.

THEOREM 3.3. *ALG-MLSC is an* $O(\alpha\rho\sigma)$*-approximation algorithm for Minimum Latency Submodular Cover.*

We now analyze ALG-MLSC. We say that the algorithm is in the jth phase when the variable k of the for loop in Step 2 has value j. Note that there are $4\alpha\rho$ iterations of Steps 4–6 in each phase.

PROPOSITION 3.4. *The length of* π *at the end of phase* j *is at most* $16\alpha\rho\sigma \cdot 2^j$. *Hence any vertex added to* S *in the* j*th phase is visited by* π *within* $16\alpha\rho\sigma \cdot 2^j$.

PROOF. The final solution is a concatenation of the paths that were found in Step 6. Since all these paths are stitched at the root r, the length of π at the end of phase j is at most $\sum_{k=0}^{j} 2 \cdot 4\alpha\rho \cdot \sigma 2^k \leq 16\alpha\rho\sigma \cdot 2^j$. □

For the sake of analysis, we now make the following modification. We artificially *increase* the length of path π at certain points so

$$\text{For each phase } j \geq 0\text{, the length of } \pi \text{ at the end of phase-}j \text{ is exactly } 16\alpha\rho\sigma \cdot 2^j. \quad (4)$$

This modification is valid due to Proposition 3.4.

Let $R(t)$ denote the set of (indices of) the functions that are not covered by ALG-MLSC earlier than time t; $R(t)$ includes the functions that are covered exactly at time t as well. We interchangeably use $i \in R(t)$ and $f_i \in R(t)$. For any $j \geq 0$, let $R_j := R(16\alpha\rho\sigma\, 2^j)$ be the set of uncovered functions at the end of phase j. Similarly, we let $R^*(t)$ denote the set of functions that are not covered by OPT earlier than time t and let $R^*_j = R^*(2^j)$. Let $R_{-1} := \emptyset$.

We show the following key lemma. It shows that the number of uncovered functions by ALG-MLSC must decrease fast as j grows, unless the corresponding number in the optimal solution is comparable.

Lemma 3.5. *For any $j \geq 0$, we have $|R_j| \leq \frac{1}{4}|R_{j-1}| + |R^*_j|$.*

Proof. The lemma holds trivially when $j = 0$. Now consider any fixed phase $j \geq 1$. Let S_0 denote the set of vertices that were added to S up to the end of phase $j-1$. Let $H = 4\alpha\rho$ and $T_1, T_2, \ldots, T_H$ be the sets of vertices that were added in Line (6) in the jth phase. Let $S_h = S_0 \cup T_1 \cup T_2 \cup \ldots \cup T_h$, $\forall 1 \leq h \leq H$. We prove Lemma 3.5 by lower and upper bounding the quantity

$$\Delta_j \;:=\; \sum_{h=1}^{H} f^{S_{h-1}}(T_h) \;=\; \sum_{h=1}^{H} \sum_{i\in[m]:\, f_i(S_{h-1})<1} \frac{f_i(S_h) - f_i(S_{h-1})}{1 - f_i(S_{h-1})},$$

which is intuitively the total amount of "residual requirement" that is covered by the algorithm in phase j.

We first lower bound Δ_j. Let T^* denote the set of vertices that OPT visited within time 2^j. Observe that a feasible solution to the SOP instance in Step 5 is OPT's prefix of length 2^j that covers vertices T^*. So by the approximation guarantee of ALG-SOP, we obtain

Proposition 3.6. *For any $h \in [H]$, we have $f^{S_{h-1}}(T_h) \geq \frac{1}{\rho} \cdot f^{S_{h-1}}(T^*)$.*

We restrict our concern to the functions in $R_j \backslash R^*_j$. Observe that for any $i \in R_j$ and $h \in [H]$, $f_i(S_{h-1}) < 1$ and that for any $i \notin R^*_j$, $f_i(T^*) = 1$. Hence by summing the inequality in the above proposition over all functions f_i in $R_j \backslash R^*_j$, we have

$$\begin{aligned}
\Delta_j \;&\geq\; \frac{1}{\rho}\sum_{h=1}^{H} f^{S_{h-1}}(T^*) \;\geq\; \frac{1}{\rho}\sum_{h=1}^{H}\sum_{i\in R_j\backslash R^*_j} \frac{f_i(T^*\cup S_{h-1}) - f_i(S_{h-1})}{1 - f_i(S_{h-1})} \\
&\geq\; \frac{1}{\rho}\sum_{h=1}^{H}\sum_{i\in R_j\backslash R^*_j} 1 \;\geq\; \frac{H}{\rho}(|R_j| - |R^*_j|) \\
&=\; 4\alpha(|R_j| - |R^*_j|). \qquad (5)
\end{aligned}$$

We now upper bound Δ_j. Note that for any $i \notin R_{j-1}$, $f_i(S_0) = 1$, and therefore f_i does not contribute to Δ_j. For any $i \in R_{j-1}$, the total contribution of f_i to Δ_j is at most α by Claim 2.4. Hence,

$$\Delta_j \leq \alpha|R_{j-1}|. \qquad (6)$$

Combining Equations (5) and (6) completes the proof of Lemma 3.5. □

Finally, we can use Lemma 3.5 to prove Theorem 3.3 exactly as we proved Theorem 2.1 in the previous section using Lemma 2.2. We omit repeating the calculations here.

4. LATENCY COVERING STEINER TREE

In this section, we consider the LCST problem, which is an interesting special case of MLSC. Recall that the input to LCST consists of a symmetric metric (V, d), root $r \in V$, and a collection $\mathcal{G}$ of groups, where each group $g \in \mathcal{G}$ is a subset of vertices with an associated requirement k_g. The goal is to find a path starting from r that minimizes the total cover time of all groups. We say that group g is covered at the earliest time t when the path within distance t visits at least k_g vertices in g. We give an $O(\log g_{max} \cdot \log |V|)$-approximation algorithm for this problem where $g_{max} := \max_{g \in \mathcal{G}} |g|$ is the maximum group size. This would prove Theorem 1.2.

Simplifying Assumptions. Following Konjevod et al. [2002] and Gupta and Srinivasan [2006], without loss of generality, we assume that:

(1) The metric is induced by a tree $T = (V, E)$ with root r and weight w_e on each edge $e \in E$.
(2) Every vertex in a group is a leaf, that is, has degree one in T.
(3) The groups in $\mathcal{G}$ are disjoint.
(4) Every vertex of degree one lies in some group.

The only non-trivial assumption is the first one, which uses tree embedding [Fakcharoenphol et al. 2004] to reduce general metrics to trees at the loss of an $O(\log |V|)$ approximation factor. In the rest of this section, we work with such instances of LCST and obtain an $O(\log g_{max})$-approximation algorithm.

We first discuss a new LP relaxation for the Covering Steiner Tree problem in Section 4.1, which is shown to have a poly-logarithmic integrality gap in Section 4.2. Next, in Section 4.3, we provide an LP relaxation for Latency Covering Steiner Tree: Given our new LP relaxation for CST, the LCST LP is a natural extension of a previously known LP for a special case [Chakrabarty and Swamy 2011]. Finally, in Section 4.4, we present the rounding algorithm for Latency Covering Steiner Tree.

4.1. New LP Relaxation for CST

Recall that the input to Covering Steiner Tree consists of a metric (V, d) with root r and a collection of m groups $\mathcal{G} \subseteq 2^V$ where each group $g \in \mathcal{G}$ is associated with a requirement k_g. The goal is to find a minimum cost r-rooted tree that includes r and at least k_g vertices from each group g. Although an $O(\log m \cdot \log g_{max} \cdot \log n)$ approximation is known for CST [Gupta and Srinivasan 2006], there was no (single) linear program known to have a poly-logarithmic integrality gap. Previous results on CST relied on an LP with a large $\Omega(k_{max})$ integrality gap [Konjevod et al. 2002].

We introduce stronger constraints that yield an LP for CST with the integrality gap $O(\log m \cdot \log g_{max} \cdot \log n)$. This new LP is an important ingredient in our algorithm for LCST and might also be useful in other contexts.

Let L denote the set of leaves in V. Because of the above simplifying assumptions, we can label each vertex v in a group with a unique leaf edge incident on it and vice versa. We abuse notation by allowing $j \in L$ to denote both the leaf vertex and its unique incident edge. For any edge $e \in E$, let $pe(e)$ denote its unique parent edge; if e is incident to the root, then $pe(e) = \text{NIL}$. For any subset of leaves $L' \subseteq L$, let $\texttt{cut}(r, L')$ denote the family of all edge subsets whose removal separates the root r from all vertices in L'.

We formulate the following linear programming relaxation for CST on tree instances:

$$\min \sum_{e\in E} w_e x_e \qquad \mathsf{LP}_{\mathsf{CST}}$$

$$\text{s.t.} \quad x_{pe(e)} \geq x_e \qquad \forall e \in E, \qquad (7)$$

$$(k_g - |A|)\sum_{j\in B\setminus L} x_j + \sum_{j\in B\cap(L\setminus A)} x_j \geq k_g - |A| \qquad \forall g \in \mathcal{G}, \forall A \subseteq g, \forall B \in \mathtt{cut}(r, g\setminus A) \qquad (8)$$

$$x_e \in [0, 1]. \qquad \forall e \in E$$

To reduce notation, we use the convention $x_{\mathsf{NIL}} = 1$, so constraint (7) is always trivially satisfied for edges e incident to the root. We note that constraint (8) can be seen as an extension of knapsack covering inequalities, which was first introduced in Carr et al. [2000]. In the analysis, in each iteration, A will be set to the leaf edges in each group g that are already covered. Then, no subtree induced by edge j can contribute to covering the group g by more than the "residual" demand $k_g-|A|$, conditioned on the edge j being chosen. This constraint will be used to show that the KRS properties (see Definition 4.4) are satisfied, which play a crucial role in the analysis of the iterative rounding.

Validity of $\mathsf{LP}_{\mathsf{CST}}$. We first argue that this is a valid relaxation. Consider any instance of CST on trees and a fixed feasible solution (tree) τ^*, which gives a natural integral solution: $x_e = 1$ if and only if $e \in \tau^*$. We focus on constraints (8), since the other constraints are obviously satisfied. Consider any $g \in \mathcal{G}$, $A \subseteq g$, and $B \in \mathtt{cut}(r, g\setminus A)$. Let $\tau^*(E\setminus A)$ denote the subtree induced by the edges in $\tau^* \bigcap (E\setminus A)$. Note that $\tau^*(E\setminus A)$ is connected, since A consists only of leaf edges. Let $C = \tau^*(E\setminus A) \cap g$ denote the leaf edges of group g in $\tau^*(E\setminus A)$. Since τ^* has at least k_g edges from g (it is a feasible CST solution), we must have $|C| \geq k_g-|A|$. Note also that the edge set B separates all leaves C from r.

—Suppose that there exists $\bar{j} \in \tau^*(E\setminus A) \cap B$ such that $\bar{j} \notin L$. Then, since $\bar{j} \in B\setminus L$, it follows that $(k_g - |A|)\sum_{j\in B\setminus L} x_j \geq k_g - |A|$, and hence the constraint is satisfied.

—The remaining case has $\tau^*(E\setminus A) \cap B \subseteq L$, that is, B separates C from r using only leaf edges. So $B \supseteq C$ and $\sum_{j\in B\cap(L\setminus A)} x_j \geq |C| \geq k_g - |A|$.

In both the above cases, constraint (8) is satisfied.

Solving $\mathsf{LP}_{\mathsf{CST}}$. Since $\mathsf{LP}_{\mathsf{CST}}$ has exponentially many constraints, in order to solve it in polynomial time, we need a separation oracle. Again we focus on constraints (8), since other constraints are only polynomially many. We observe that this separation oracle reduces to the following problem.

Problem MinCutWithExceptions: Given as input a tree T rooted at r with leaves L and cost $\ell(e)$ on each edge e and an integer $D \geq 0$, the goal is to find a minimum cost cut that separates r from any D leaves.

We first show why this suffices to separate constraints (9). Consider any fixed $g \in \mathcal{G}$ and all $A \subseteq g$ with $|A| = \eta$ (for some fixed value $0 \leq \eta \leq n$). We will show that the constraints in Equation (8) corresponding to group g and $A \subseteq g$ with $|A| = \eta$ (and *any* cut B) can be verified by solving one instance of MinCutWithExceptions. Note that all such constraints have the same right-hand side $k_g-\eta$. In order to verify these constraints, we would like to find $A \subseteq g$ with $|A| = \eta$ and $B \in \mathtt{cut}(r, g\setminus A)$ that minimizes the left-hand side of Equation (8) and check if this value is smaller than $k_g - \eta$. This test can be cast as the following MinCutWithExceptions instance:

Remove all edges from E that are not on any path from the root r to a vertex in g, and let T' be the resulting tree. T' is the input tree to the MinCutWithExceptions instance. Note that leaves of T' are precisely g. For all leaf edges $j \in g$, let $\ell(j) := x_j$; and for all non-leaf $e \in T' \backslash g$, $\ell(e) := (k_g - \eta) \cdot x_e$. Also set bound $D := |g| - \eta$.

Finally, we iterate over all $g \in \mathcal{G}$ and $0 \le \eta \le n$ in order to verify all constraints in Equation (8).

We next show that MinCutWithExceptions can be solved via a dynamic programming.

LEMMA 4.1. *The problem MinCutWithExceptions can be solved in polynomial time.*

PROOF. To formally describe our dynamic program, we make some simplifying assumptions. By introducing dummy edges of infinite cost, we assume without loss of generality that the tree T is binary and the root r is incident to exactly one edge e_r. Hence every non-leaf edge e has exactly two child edges, e_1 and e_2. For any edge $e \in T$, let T_e denote the subtree of T rooted at e, that is, T_e contains edge e and all its descendants.

We define a recurrence for $C[e, k]$ that denotes the minimum cost cut that separates the root of T_e from exactly k leaves in T_e. Note that $C[e_r, D]$ gives the optimal value.

For any leaf edge f set:

$$C[f, k] = \begin{cases} 0 & \text{if } k = 0, \\ \ell(f) & \text{if } k = 1, \text{ and} \\ \infty & \text{otherwise.} \end{cases}$$

For any non-leaf edge e with children e_1 and e_2 set:

$$C[e, k] = \begin{cases} 0 & \text{if } k = 0; \\ \min\limits_{k_1+k_2=k} \{C[e_1, k_1] + C[e_2, k_2]\} & \text{if } 1 \le k < |L \cap T_e|; \\ \min \begin{cases} \ell(e), \\ \min\limits_{k_1+k_2=k} \{C[e_1, k_1] + C[e_2, k_2]\} \end{cases} & \text{if } k = |L \cap T_e|; \\ \infty & \text{otherwise.} \end{cases}$$

It can be checked directly that this recurrence computes the desired values in polynomial time. □

4.2. Rounding Algorithm for CST

In this section, we prove the following LP rounding result.

THEOREM 4.2. *Let z be any feasible solution to $\mathsf{LP}_{\mathsf{CST}}$. There is a randomized polynomial time algorithm that returns a random tour P (on tree T) originating from r such that*

$$\mathbb{E}[w(P)] \le 12(3 + \log g_{max}) \sum_e w_e \cdot z_e \qquad and \qquad \Pr[|P \cap g| < k_g] < e^{-3}\ \forall g \in \mathcal{G}.$$

Hence, for any $g \in \mathcal{G}$, $\Pr[w(P) > 96(3 + \log g_{max}) \sum_e w_e \cdot z_e$ or $|P \cap g| < k_g] < \frac{1}{4}$.

Before presenting the algorithm for this, we discuss the basic rounding scheme from Konjevod et al. [2002] (which is an extension of Garg et al. [2000]) and some of its useful properties. We call the rounding scheme ALG-KRS.

PROPOSITION 4.3 (KONJEVOD ET AL. [2002]). *Each edge e is included in the final solution of ALG-KRS with probability z_e.*

PROOF. We prove this by induction on the depth of edge e from r. The base case involves edges incident to the root r, where this property is clearly true. For the inductive step, assume that the parent edge $pe(e)$ of e is included with probability $z_{pe(e)}$; then, by the algorithm description, edge e is included with probability $z_{pe(e)} \cdot \frac{z_e}{z_{pe(e)}} = z_e$. □

ALGORITHM 3: The Rounding Procedure ALG-KRS

INPUT: Undirected tree $T = (V, E)$ rooted at r; $z_e \in [0, 1]$, such that for all $e \in E$, $z_{pe(e)} \geq z_e$.

$S \leftarrow \emptyset$.

For each $e \in E$ incident to the root r, add e to S with probability z_e.

For each $e \in E$ such that $pe(e) \in S$, add e to S with probability $\frac{z_e}{z_{pe(e)}}$.

OUTPUT: The connected component (tree) S.

Definition 4.4 (*KRS properties*). Consider any $z \in [0, 1]^E$, $g \in \mathcal{G}$, $R(g) \subseteq g$, and $0 \leq r_g \leq |R(g)|$. We say that $(z, R(g), r_g)$ satisfies the KRS properties if it satisfies the following:

$$z_{pe(e)} \geq z_e \qquad \forall e \in E, \tag{9}$$

$$\sum_{j \in T(e) \cap R(g)} z_j \leq r_g \cdot z_e \qquad \forall e \in E, \tag{10}$$

where $T(e)$ is the subtree below (and including) edge e.

The first property (Equation (9)) is the same as the constraints of Equation (7). The second property (Equation (10)) is a Lipschitz-type condition, which implies that, conditional on any edge e being chosen, its subtree $T(e)$ can contribute at most r_g to the requirement of $R(g)$.

LEMMA 4.5 (KONJEVOD ET AL. [2002]). *Suppose that $(z, R(g), r_g)$ satisfies the KRS properties for all groups g. Let L_{krs} denote the set of leaves that are covered by ALG-KRS with input $\{z_e : e \in E\}$. Consider any constant $\delta \in [0, 1]$. Then, for any $g \in \mathcal{G}$,*

$$\Pr[|L_{krs} \cap R(g)| \leq (1 - \delta)\mu_g] \leq \exp\Big(-\frac{\delta^2 \cdot \mu_g}{2 + r_g(1 + \ln |R(g)|)}\Big),$$

where $\mu_g := \mathbb{E}[|L_{krs} \cap R(g)|] = \sum_{j \in R(g)} z_j$.

PROOF. We only give a sketch of the proof, since this is implicit in Konjevod et al. [2002]. For any $j, j' \in R(g)$, we say that $j \sim j'$ if and only if (1) $j \neq j'$ and (2) the least common ancestor $\mathtt{lca}(j, j')$ of j and j' is not r. Define

$$\Delta_g := \sum_{j, j' \in R(g): j \sim j', z_{\mathtt{lca}(j,j')} > 0} \frac{z_j \cdot z_{j'}}{z_{\mathtt{lca}(j,j')}}.$$

In Theorem 3.2 in Konjevod et al. [2002], Konjevod et al. showed using the KRS properties that

$$\Delta_g \leq \mu_g(r_g - 1 + r_g \ln |R(g)|),$$

where $\mu_g = \mathbb{E}[|L_{krs} \cap R(g)|] = \sum_{j \in R(g)} \Pr[j \in L_{krs}] = \sum_{j \in R(g)} z_j$ by Proposition 4.3.

We note that the proof of Theorem 3.2 implies this, although it is stated only for $\mu_g = r_g$. Further, they used this bound in Jansen's inequality to obtain, for any $\delta \in [0, 1]$,

$$\Pr[|L_{krs} \cap R(g)| \leq (1 - \delta)\mu_g] \leq \exp\Big(-\frac{\delta^2 \mu_g}{2 + \Delta_g/\mu_g}\Big).$$

Combining the above two inequalities yields the lemma. □

PROOF OF THEOREM 4.2. The rounding algorithm is given as Algorithm 4.

The preprocessing of $\overline{x}$ to obtain $\tilde{x}$ (Line 3) is done as described in the next lemma.

ALGORITHM 4: Rounding Algorithm for Covering Steiner Tree.

INPUT: Tree T with edge lengths, root r, groups $\mathcal{G}$, requirements $\{k_g\}_{g\in\mathcal{G}}$, and solution $\overline{x} \in \mathsf{LP}_{\mathsf{CST}}$.

1: $\overline{E} \leftarrow \{e \in E \mid \overline{x}_e \geq 1/2\}$, $R(g) \leftarrow g\backslash\overline{E}$ and $r_g \leftarrow k_g - |g \cap \overline{E}|$.
2: Shrink all edges in $\overline{E}$ in T and let $\tilde{T}$ be the resulting tree with the edge set $\tilde{E} := E\backslash\overline{E}$.
3: Obtain solution $\tilde{x}$ from $\overline{x}$ using Lemma 4.6.
4: For each $e \in \tilde{E}$, $z_e \leftarrow 2\tilde{x}_e$; note that $z_e \in [0, 1]$.
5: $\mathcal{S} \leftarrow \emptyset$.
6: **repeat** the following $6(3 + \log g_{max})$ times:
7: $\quad\tau \leftarrow$ the tree produced by ALG-KRS with fractional solution z on tree $\tilde{T}$
8: $\quad$Add τ to $\mathcal{S}$
9: Combine all trees in $\mathcal{S}$ with $\overline{E}$ and take an Euler tour P of the resulting tree.

OUTPUT: Path P originating from r.

LEMMA 4.6. *We can find in polynomial time $\tilde{x}_e \in [0, \overline{x}_e]$, $\forall e \in E\backslash\overline{E}$ such that $\forall g \in \mathcal{G}$:*

(1) $(\tilde{x}, R(g), r_g)$ satisfies the KRS properties in tree $\tilde{T}$.
(2) $\sum_{j\in R(g)} \tilde{x}_j \;\geq\; r_g$ (coverage property).

PROOF. Consider constraints (8) of LPCST. Fix a group $g \in \mathcal{G}$ and let $A := g \cap \overline{E}$. Consider tree $\tilde{T}$ as a flow network with each leaf edge f having capacity $\overline{x}_f$ and each non-leaf edge e having capacity $r_g \cdot \overline{x}_e$. The root r is the source and leaves $R(g) = g\backslash A$ are the sinks. Then constraints (8) imply that the min cut separating r from $R(g)$ has value at least r_g: Note that although these constraints are for the original tree T, they imply similar constraints for $\tilde{T}$ since $\tilde{T}$ is obtained from T by edge contraction.[4] Hence there must exist a max-flow of volume at least r_g from r to $R(g)$ in the above network. Let $\tilde{x}_f$ denote the volume of this flow into each leaf edge $f \in R(g)$; clearly, we have that $\tilde{x}_f \leq \overline{x}_f$ (due to capacity on leaves) and

$$\sum_{j\in R(g)} \tilde{x}_j \geq r_g. \tag{11}$$

Moreover, by the capacities on non-leaves,

$$\sum_{j\in T(e)\cap R(g)} \tilde{x}_j \leq r_g \cdot \overline{x}_e, \qquad \forall e \in E\backslash\overline{E}. \tag{12}$$

We can use the above procedure on each group $g \in \mathcal{G}$ separately to compute $\tilde{x}_f$ for all leaf edges $f \in E\backslash\overline{E}$; this is well defined since groups are disjoint. For each non-leaf edge $e \in E\backslash\overline{E}$ set $\tilde{x}_e := \overline{x}_e$. Thus we have $0 \leq \tilde{x}_e \leq \overline{x}_e$ for all $e \in E\backslash\overline{E}$. Observe that this computation can easily be done in polynomial time.

Now, Equation (12) implies the second KRS property (10). Property (9) follows, since for each $e \in E\backslash\overline{E}$, we have $\tilde{x}_{pe(e)} = \overline{x}_{pe(e)} \geq \overline{x}_e \geq \tilde{x}_e$; the first inequality is due to constraint (7) of LPCST. Finally, Equation (11) implies the coverage property claimed in the lemma. □

Consider any group $g \in \mathcal{G}$. Since all edges in $\overline{E}$ are included in P with probability 1, group g is covered by P if and only if at least r_g vertices in its residual set $R(g)$ are covered by the union of trees τ in Line (7) of Algorithm 4. This motivates us to derive the following.

[4]In particular, every cut B' separating r from $g\backslash A$ in $\tilde{T}$ is also a cut separating r from $g\backslash A$ in T.

LEMMA 4.7. *For any $g \in \mathcal{G}$,*

$$\Pr[|\tau \cap R(g)| < r_g] \leq \exp\left(-\frac{1}{2(3+\ln g_{max})}\right).$$

PROOF. From Lemma 4.6 it follows that $(\tilde{x}, R(g), r_g)$ satisfies the KRS properties on tree $\tilde{T}$. Since $z = 2 \cdot \tilde{x}$, $(z, R(g), r_g)$ also satisfies the KRS properties. Furthermore, using the coverage property in Lemma 4.6,

$$\mu_g \;:=\; \mathbb{E}[|\tau \cap R(g)|] \;=\; \sum_{j \in R(g)} z_j = \; 2 \cdot \sum_{j \in R(g)} \tilde{x}_j \;\geq\; 2r_g.$$

Here we also used Proposition 4.3 that $\Pr[j \in \tau] = z_j$. By applying Lemma 4.5 with $\delta = 1/2$, we have

$$\Pr[|\tau \cap R(g)| < r_g] \;\leq\; \exp\left(-\frac{r_g}{2(2 + r_g(1 + \ln|R(g)|))}\right) \;\leq\; \exp\left(-\frac{1}{2(3+\ln g_{max})}\right).$$

This proves Lemma 4.7. □

CLAIM 4.8. *The expected length $\mathbb{E}[w(P)] \leq 12(3 + \log g_{max}) \sum_e w_e \cdot \overline{x}_e$.*

PROOF. Consider the tree τ in any iteration of Line (7) of Algorithm 4. By Proposition 4.3, we know that each edge $e \in \tilde{E}$ is included in τ with probability $z_e = 2\tilde{x}_e \leq 2\overline{x}_e$. Since for all $e \in \overline{E}$, $\overline{x}_e \geq 1/2$, the expected total weight of the edges in $\overline{E}$ and τ is upper bounded by

$$\sum_{e \in \overline{E}} w_e + \sum_{e \in \tilde{E}} w_e \cdot 2\tilde{x} \quad\leq\quad 2\sum_{e \in E} w_e \cdot \overline{x}_e.$$

Since P contains $6(3 + \log g_{max})$ independent "copies" of tree τ, the claim follows. □

CLAIM 4.9. *Consider any group $g \in \mathcal{G}$. The probability that P does not cover g is $\Pr[|P \cap g| < k_g] < e^{-3}$.*

PROOF. Since P contains $6(3 + \log g_{max})$ independent samples of trees τ from Line (7) of Algorithm 4, by Lemma 4.7 it follows that group g is not covered by P with probability at most $1/e^3$. □

The first part of Theorem 4.2 now follows from Claims 4.8 and 4.9. The second part then follows using Markov's inequality and a union bound.

4.3. LP Relaxation for LCST

We formulate the following linear relaxation for tree instances of the Latency Covering Steiner Tree,

$$\min \quad \frac{1}{2} \cdot \sum_{\ell \geq 0} 2^\ell \sum_{g \in \mathcal{G}} (1 - y_g^\ell) \tag{$\mathsf{LP_{LCST}}$}$$

$$\text{s.t.} \quad x^\ell_{pe(e)} \geq x^\ell_e \qquad \forall \ell \geq 0, e \in E, \tag{13}$$

$$\sum_{j \in E} w_e x^\ell_e \leq 2^\ell \qquad \forall \ell \geq 0, \tag{14}$$

$$(k_g - |A|) \sum_{j \in B \setminus L} x^\ell_j + \sum_{j \in B \cap L \setminus A} x^\ell_j \geq (k_g - |A|) \cdot y^\ell_g \quad \forall \ell \geq 0, g \in \mathcal{G}, A \subseteq g, B \in \mathtt{cut}(r, g \setminus A), \tag{15}$$

$$
\begin{aligned}
&y_g^{\ell+1} \geq y_g^{\ell} && \forall \ell \geq 0, g \in \mathcal{G} && (16)\\
&x_e^{\ell} \in [0,1] && \forall \ell \geq 0, e \in E &&\\
&y_g^{\ell} \in [0,1] && \forall \ell \geq 0, g \in \mathcal{G}, &&
\end{aligned}
$$

To see that this is a valid relaxation, let OPT denote the optimal path. For any $\ell \geq 0$, let $\mathsf{OPT}(2^\ell)$ denote the prefix of length 2^ℓ in OPT. We construct a feasible integral solution to $\mathsf{LP}_{\mathsf{LCST}}$ as follows. The variable x_e^ℓ indicates if edge e lies in $\mathsf{OPT}(2^\ell)$. The indicator variable y_g^ℓ has value 1 if and only if group g is covered by $\mathsf{OPT}(2^\ell)$, that is, at least k_g vertices of g are contained in $\mathsf{OPT}(2^\ell)$. Constraints (13) follow from the fact that $\mathsf{OPT}(2^\ell)$ is a path starting at r. Constraints (14) say that the edges in $\mathsf{OPT}(2^\ell)$ have a total weight of at most 2^ℓ, which is clearly true. Note that for each $\ell \geq 0$, there is a set of constraints (15) that is similar to the constraints (8) in $\mathsf{LP}_{\mathsf{CST}}$; the validity of these constraints (15) can be shown exactly as for (8). Constraints (16) enforce the fact that if group g is covered by $\mathsf{OPT}(2^\ell)$, then it must be covered by $\mathsf{OPT}(2^{\ell+1})$ as well, which is trivially true. Now consider the objective value: The total contribution of a group g that is covered by OPT at some time $t \in (2^k,\ 2^{k+1}]$ is $\frac{1}{2} \cdot \sum_{\ell=0}^{k} 2^\ell \leq 2^k$. Thus the objective value of this integral solution is at most OPT.

We can ensure by standard scaling arguments, at the loss of a $1 + o(1)$ factor in the objective, that all distances are polynomially bounded. This implies that the length of any optimal path is also polynomial, and so it suffices to consider $O(\log n)$ many values of ℓ. Thus, the number of variables in $\mathsf{LP}_{\mathsf{LCST}}$ is polynomial. Note that constraints (15) are exponentially many. However, for each fixed ℓ and g, we can use the same separation oracle that we used for the constraints (8) of $\mathsf{LP}_{\mathsf{CST}}$.

4.4. Rounding Algorithm for LCST

We are now ready to present our algorithm to round $\mathsf{LP}_{\mathsf{LCST}}$, described formally as ALG-LCST below. Let $(\overline{x}, \overline{y})$ denote a fixed optimal solution to $\mathsf{LP}_{\mathsf{LCST}}$. The algorithm proceeds in *phases* $\ell = 0, 1, 2, \cdots$ where the ℓth phase rounding uses variables with superscript ℓ in $\mathsf{LP}_{\mathsf{LCST}}$.

ALGORITHM 5: Rounding Algorithm for Latency Covering Steiner Tree (ALG-LCST).

INPUT: Tree T with edge lengths, root r, groups $\mathcal{G}$, and requirements $\{k_g\}_{g\in\mathcal{G}}$.

1: $\pi \leftarrow \emptyset$.
2: Let $(\overline{x}, \overline{y})$ be an optimal solution to $\mathsf{LP}_{\mathsf{LCST}}$.
3: **for** $\ell = 0, 1, 2, \ldots$ **do**
4:     Run Algorithm 4 on solution $x^\ell := \min\{2\overline{x}^\ell, \mathbf{1}\}$ to obtain tour P^ℓ originating from r.
5:     **if** P^ℓ has weight at most $192(3 + \log g_{max}) \cdot 2^\ell$ **then**
6:         $\pi \leftarrow \pi \cdot P^\ell$.
7: **end for**

OUTPUT: Path π originating from r.

We now prove that Algorithm 5 achieves an $O(\log g_{max})$ approximation for LCST on tree instances. Using probabilistic tree embedding [Fakcharoenphol et al. 2004], it would follow that it yields an $O(\log g_{max} \cdot \log |V|)$ approximation for general metrics, thereby proving Theorem 1.2.

For any group $g \in \mathcal{G}$, define $\ell(g)$ to be the smallest $\ell \geq 0$ such that $\overline{y}_g^\ell \geq \frac{1}{2}$. By constraints (16), for any $\ell \geq \ell(g)$, we have $\overline{y}_g^\ell \geq \frac{1}{2}$. Hence, solution x^ℓ in line 4 is feasible to LP_{LCST} restricted to the groups $\{g \in \mathcal{G} : \ell(g) \le \ell\}$. Applying Theorem 4.2 to solution $x^\ell \le 2\overline{x}^\ell$ we obtain (using $\sum_e w_e \cdot \overline{x}_e^\ell \le 2^\ell$)

PROPOSITION 4.10. *For any $g \in \mathcal{G}$ and $\ell \ge \ell(g)$,*

$$\Pr\left[w(P^\ell) > 192(3 + \log g_{max}) \cdot 2^\ell \;\; or \;\; |P^\ell \cap g| < k_g\right] \quad < \quad \frac{1}{4}.$$

Fix any group $g \in \mathcal{G}$, and $\ell \ge \ell(g)$. Among $P^{\ell(g)}, P^{\ell(g)+1}, \ldots, P^\ell$, consider the paths that are added to π. Clearly, the total weight of such paths is at most $O(\log g_{max} \cdot 2^\ell)$. By Proposition 4.10, the probability that none of these paths covers g is at most $\frac{1}{4^{\ell-\ell(g)+1}}$. Hence the expected cover time of g is at most

$$\sum_{\ell \ge \ell(g)} O(\log g_{max}) \cdot 2^\ell \cdot \frac{1}{4^{\ell-\ell(g)+1}} = O(\log g_{max}) \cdot 2^{\ell(g)}.$$

So the expected total cover time is at most $O(\log g_{max}) \cdot \sum_{g \in \mathcal{G}} 2^{\ell(g)}$.

By definition of $\ell(g)$ we know

$$\text{OPT} \ge \frac{1}{2} \cdot \sum_{\ell \ge 0} 2^\ell \sum_{g \in \mathcal{G}} (1 - \overline{y}_g^\ell) \ge \frac{1}{2} \sum_{g \in \mathcal{G}} 2^{\ell(g)-1}\left(1 - \overline{y}_g^{\ell(g)-1}\right) \ge \frac{1}{8} \cdot \sum_{g \in \mathcal{G}} 2^{\ell(g)}.$$

It follows that Algorithm 5 achieves an $O(\log g_{max})$ approximation for LCST on tree instances, as desired.

5. WEIGHTED STOCHASTIC SUBMODULAR RANKING

In this section, we study the WSSR. The input consists of a set $\mathcal{A} = \{X_1, \ldots, X_n\}$ of n independent random variables (stochastic elements), each over domain Δ, with integer lengths $\{\ell_j\}_{j=1}^n$ (deterministic) and m monotone submodular functions $f_1, \ldots, f_m : 2^\Delta \to [0, 1]$ on ground set Δ. We are also given the distribution (over Δ) of each stochastic element $\{X_j\}_{j=1}^n$. (We assume explicit probability distributions, that is, for each X_j and $b \in \Delta$ we are given $\Pr[X_j = b]$.) The realization $x_j \in \Delta$ of the random variable X_j is known immediately after scheduling it. Here, X_j requires ℓ_j units of time to be scheduled; if X_j is started at time t, then it completes at time $t + \ell_j$ at which point its realization $x_j \in \Delta$ is also known. A feasible solution/policy is an adaptive ordering of $\mathcal{A}$, represented naturally by a decision tree with branches corresponding to the realization of the stochastic elements. We use $\langle \pi(1), \ldots, \pi(n) \rangle$ to denote this ordering, where each $\pi(k)$ is a random variable denoting the index of the kth scheduled element.

The cover time $\texttt{cov}(f_i)$ of any function f_i is defined as the earliest time t such that f_i has value one for the realization of the elements that are completely scheduled within time t. More formally, $\texttt{cov}(f_i)$ is the earliest time t such that $f_i(\{x_{\pi(1)}, \ldots, x_{\pi(k_t)}\})$ is equal to 1 where k_t is the maximum index such that $\ell_{\pi(1)} + \ell_{\pi(2)} + \ldots + \ell_{\pi(k_t)} \le t$. If the function value never reaches 1 (due to the stochastic nature of elements), then $\texttt{cov}(f_i) = \ell_1 + \ell_2 + \ldots + \ell_n$, which is the maximum time of any schedule. Note that the cover time is a random variable since the order π is random. The goal is to find a policy that (approximately) minimizes the expected total cover time $\mathbb{E}[\sum_{i \in [m]} \texttt{cov}(f_i)]$.

5.1. Applications

Our stochastic extension of submodular ranking captures many interesting applications.

Stochastic Set Cover. We are given as input a ground set Δ and a collection $\mathcal{S} \subseteq 2^\Delta$ of deterministic subsets. There are stochastic elements $\{X_j : j \in [n]\}$, each associated with a probability distribution over Δ and having respective costs $\{\ell_j : j \in [n]\}$. The goal is to find an adaptive policy that hits all sets in $\mathcal{S}$ at the minimum expected cost. This problem was studied in Goemans and Vondrák [2006], Munagala et al. [2007], and Liu et al. [2008]. The problem can be shown to be an instance of WSSR with a single monotone submodular function $f_1(A) := \frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}} \min\{1, |A \cap S|\}$ and parameter $\epsilon = 1/|\mathcal{S}|$.

Shared Filter Evaluation. This problem was introduced by Munagala et al. [2007], and the result was improved to an essentially optimal solution in Liu et al. [2008]. In this problem, there is a collection of independent "filters" $X_1, X_2, \ldots, X_n$, each of which gets evaluated either to `True` or `False`. For each filter $j \in [n]$, we are given the "selectivity" $p_j = \Pr[X_i \text{ is true}]$ and the cost ℓ_j of running the filter. We are also given a collection $\mathcal{Q}$ of queries, where each query Q_i is a conjunction of a subset of filters. We would like to determine each query in $\mathcal{Q}$ to be `True` or `False` by (adaptively) testing filters of the minimum expected cost. In order to cast this problem as WSSR, we use $\Delta = \bigcup_{j=1}^{n}\{\mathtt{True}_j, \mathtt{False}_j\}$; for each $j \in [n]$, $X_j = \mathtt{True}_j$ with probability p_j, and $X_j = \mathtt{False}_j$ with the remaining probability $1-p_j$. We create one monotone submodular function:

$$f_1(A) := \frac{\sum_{Q_i\in\mathcal{Q}} \min\left\{1,\ |A\cap\{\mathtt{False}_j : j\in Q_i\}| + \frac{1}{|Q_i|}\cdot|A\cap\{\mathtt{True}_j : j\in Q_i\}|\right\}}{|\mathcal{Q}|}.$$

(Note that a query Q_i gets evaluated to `False` if any one of its filters is `False` and `True` if all its filters are `True`.) Here the parameter $\epsilon = 1/(|\mathcal{Q}|\max_i |Q_i|)$.

We note that the Shared Filter Evaluation problem also can be studied for a latency type of objective. In this case, for each query $Q_i \in \mathcal{Q}$, we create a separate submodular function:

$$f_i(A) := \min\left\{1,\ |A\cap\{\mathtt{False}_j : j\in Q_i\}| + \frac{1}{|Q_i|}\cdot|A\cap\{\mathtt{True}_j : j\in Q_i\}|\right\}.$$

In this case, the WSSR problem corresponds precisely to filter evaluation that minimizes the *average time* to answer queries in $\mathcal{Q}$. The parameter $\epsilon = 1/(\max_i |Q_i|)$.

Stochastic Generalized Min-Sum Set Cover. We are given as input a ground set Δ and a collection $\mathcal{S} \subseteq 2^{\Delta}$ of deterministic subsets with requirement $k(S)$ for each $S \in \mathcal{S}$. There are stochastic elements $\{X_j : j \in [n]\}$, each defined over Δ. Set $S \in \mathcal{S}$ is said to be completed when at least $k(S)$ elements from S have been scheduled. The goal is to find an adaptive ordering of $[n]$ to minimize the expected total completion time. This can be reduced to WSSR by defining function $f^S(A) := \min\{1, |A \cap S|/k(S)\}$ for each $S \in \mathcal{S}$; here $\epsilon = 1/k_{max}$, where k_{max} denotes the maximum requirement.

For this problem, our result implies an $O(\log k_{max})$ approximation to adaptive policies. However, for non-adaptive policies (where the ordering of elements is fixed *a priori*), one can obtain a better $O(1)$-approximation algorithm by combining the Sample Average Approximation method [Kleywegt et al. 2002; Charikar et al. 2005] with $O(1)$ approximations known for the non-stochastic version [Bansal et al. 2010; Skutella and Williamson 2011].

We also note that the analysis in Azar and Gamzu [2011] for deterministic submodular ranking was only for elements having unit sizes. Our analysis also holds under non-uniform sizes.

5.2. Algorithm and Analysis

We consider adaptive policies: This chooses at each time $\ell_{\pi(1)} + \ell_{\pi(2)} + \ldots + \ell_{\pi(k-1)}$ the element

$$X_{\pi(k)} \in \mathcal{A} \backslash \{X_{\pi(1)}, X_{\pi(2)}, X_{\pi(3)}, \ldots, X_{\pi(k-1)}\},$$

after observing the realizations $x_{\pi(1)}, \ldots, x_{\pi(k-1)}$. So it can be described as a decision tree. Our main result is an $O(\log \frac{1}{\epsilon})$-approximate adaptive policy, which proves Theorem 1.3. This result is again inspired by our simpler analysis of the algorithm from Azar and Gamzu [2011].

To formally describe our algorithm, we quickly define the probability spaces we are concerned with. We use $\Omega = \Delta^n$ to denote the outcome space of $\mathcal{A}$. We use the same notation Ω to denote the probability space induced by this outcome space. For any $S \subseteq \mathcal{A}$ and its realization s, let $\Omega(s)$ denote the outcome subspace that conforms to s. We can naturally define the probability space defined by $\Omega(s)$ as follows: The probability that $w \in \Omega(s)$ occurs is $\Pr_\Omega[w]/\Pr_\Omega[\Omega(s)]$; we also use $\Omega(s)$ to denote this probability space.

The main algorithm is given below and is a natural extension of the deterministic algorithm [Azar and Gamzu 2011]. Let $\alpha := 1 + \ln(\frac{1}{\epsilon})$. In the output, $\pi(k)$ denotes the kth element in $\mathcal{A}$ that is scheduled.

ALGORITHM 6: Algorithm for Stochastic Submodular Cover (ALG-AG-STO).

1: **INPUT**: $\mathcal{A} = \{X_1, \ldots, X_n\}$ with $\{\ell_1, \ldots, \ell_n\}$; $f_i : 2^\Delta \to [0, 1]$, $i \in [m]$.
2: $S \leftarrow \emptyset$. (S are the elements completely scheduled so far, and s their instantiation.)
3: **while** there exists function f_i with $f_i(s) < 1$ **do**
4: Choose element X_e as follows,

$$X_e = \arg\max_{X_e \in \mathcal{A} \backslash S} \frac{\mathbb{E}_{\Omega(s)}\left[\sum_{i \in [m],\, f_i(s) < 1} \frac{f_i(s \cup \{X_e\}) - f_i(s)}{1 - f_i(s)}\right]}{\ell_e}$$

5: $S \leftarrow S \bigcup \{X_e\}$.
6: $\pi(|S|) \leftarrow X_e$. Schedule X_e and observe its realization.
7: **end while**
8: **OUTPUT**: An adaptive ordering π of $\mathcal{A}$.

Observe that taking expectation over $\Omega(s)$ in Step 4 is the same as expectation over the distribution of X_e since $X_e \notin S$ and the elements are independent. This value can be computed exactly since we have an explicit probability distribution of X_e. Also note that this algorithm implicitly defines a decision tree. We will show that ALG-AG-STO is an $O(\ln(\frac{1}{\epsilon}))$-approximation algorithm for WSSR.

To simplify notation, without loss of generality, we assume that α is an integer. Let $R(t)$ denote the (random) set of functions that are not satisfied by ALG-AG-STO before time t. Note that the set $R(t)$ includes the functions that are satisfied exactly at time t. Analogously, the set $R^*(t)$ is defined for the optimal policy. For notational convenience, we use $i \in R(t)$ interchangeably with $f_i \in R(t)$. Let $C(t) := \{f_1, \ldots, f_m\} \backslash R(t)$ and $C^*(t) := \{f_1, \ldots, f_m\} \backslash R^*(t)$. Note that all the sets $C(\cdot)$, $C^*(\cdot)$, $R(\cdot)$, and $R^*(\cdot)$ are stochastic. We have that $\mathsf{ALG} = \sum_{t \in [n]} |R(t)|$ and $\mathsf{OPT} = \sum_{t \in [n]} |R^*(t)|$ and hence ALG and OPT are stochastic quantities. We show that $\mathbb{E}[\mathsf{ALG}] = O(\alpha) \cdot \mathbb{E}[\mathsf{OPT}]$, which suffices to prove the desired approximation ratio.

We are interested in the number of unsatisfied functions at times $\{8\alpha 2^j : j \in \mathbb{Z}_+\}$ by ALG-AG-STO and the number of unsatisfied functions at times $\{2^j : j \in \mathbb{Z}_+\}$ by the optimal policy. Let $R_j := R(8\alpha 2^j)$ and $R_j^* = R^*(2^j)$. It is important to note that R_j and R_j^* are concerned with different times, and they are stochastic. For notational simplicity, we let $R_{-1} := \emptyset$.

We show the following key lemma. Once we prove this lemma, we can complete the proof similar to the proof of Theorem 2.1 via Lemma 2.2.

LEMMA 5.1. *For any $j \geq 0$, we have $\mathbb{E}[|R_j|] \leq \frac{1}{4}\mathbb{E}[|R_{j-1}|] + \mathbb{E}[|R_j^*|]$.*

PROOF. The lemma trivially holds for $j = 0$, so we consider any $j \geq 1$. For any $t \geq 1$, we use s_{t-1} to denote the set of elements *completely* scheduled by ALG-AG-STO by time $t-1$ along with their instantiations; clearly this is a random variable. Also, for $t \geq 1$ let $\sigma(t) \in [n]$ denote the (random) index of the element being scheduled by ALG-AG-STO during time slot $(t-1, t]$. Since elements have different sizes, note that $\sigma(t)$ differs from $\pi(t)$, which is the tth element scheduled by ALG-AG-STO. Observe that s_{t-1} determines $\sigma(t)$ precisely but not the instantiation of $X_{\sigma(t)}$.

Let $E_j^* \subseteq \mathcal{A}$ be the (stochastic) set of elements that are completely scheduled by the optimal policy within time 2^j. For any stochastic set (or element) S, we denote its realization under an outcome w as $S(w)$. For example, $X_i(w) \in \Delta$ is the realization of element X_i for outcome w; and $E_j^*(w)$ is the set of elements completely scheduled by time 2^j in OPT (under w) along with their realizations.

For any time t and corresponding outcome s_{t-1}, define a set function:

$$f^{s_{t-1}}(D) := \sum_{i\in[m],\, f_i(s_{t-1})<1} \frac{f_i(s_{t-1}\cup D) - f_i(s_{t-1})}{1 - f_i(s_{t-1})}, \qquad \forall D \subseteq \Delta.$$

We also use $f_i^{s_{t-1}}(D)$ to denote the term inside the above summation.

The function $f^{s_{t-1}} : 2^\Delta \to \mathbb{R}_+$ is monotone and submodular since it is a summation of monotone and submodular functions. We also define

$$F^{s_{t-1}}(X_e) := \mathbb{E}_{\,w\leftarrow\Omega(s_{t-1})}\left[f^{s_{t-1}}(X_e(w))\right], \qquad \forall X_e \in \mathcal{A}. \tag{17}$$

Observe that this is zero for elements $X_e \in s_{t-1}$.

PROPOSITION 5.2. *Consider any time t and outcome s_{t-1}. Note that s_{t-1} determines $\sigma(t)$. Then:*

$$\frac{1}{\ell_{\sigma(t)}} \cdot F^{s_{t-1}}(X_{\sigma(t)}) \geq \frac{1}{\ell_i} \cdot F^{s_{t-1}}(X_i), \qquad \forall X_i \in \mathcal{A}. \quad \square$$

PROOF. At some time $t' \leq t-1$ (right after s_{t-1} is observed) ALG-AG-STO chose to schedule element $X_{\sigma(t)}$ over all elements $X_i \in \mathcal{A}\backslash s_{t-1}$. By the greedy rule, we know that the claimed inequality holds for any $X_i \in \mathcal{A}\backslash s_{t-1}$. Furthermore, the inequality holds for any element $X_i \in s_{t-1}$, since here $F^{s_{t-1}}(X_i) = 0$. $\square$

We now define the *expected gain* by ALG-AG-STO in step t as

$$G_t := \mathbb{E}_{s_{t-1}}\left[\frac{1}{\ell_{\sigma(t)}} F^{s_{t-1}}(X_{\sigma(t)})\right] \tag{18}$$

and the expected total gain as

$$\Delta_j := \sum_{t=8\alpha 2^{j-1}}^{8\alpha 2^j} G_t. \tag{19}$$

We complete the proof of Lemma 5.1 by upper and lower bounding Δ_j.

Upper Bound for Δ_j. Fix any outcome $w \in \Omega$. Below, all variables are *conditioned on* w, and hence they are all deterministic. (For ease of notation, we do not write w in front of the variables.)

$$\begin{aligned}
\Delta_j &:= \sum_{t=8\alpha 2^{j-1}}^{8\alpha 2^j} \frac{1}{\ell_{\sigma(t)}} f^{s_{t-1}}(x_{\sigma(t)}) = \sum_{t=8\alpha 2^{j-1}}^{8\alpha 2^j} \frac{1}{\ell_{\sigma(t)}} \sum_{i \in [m]: f_i(s_{t-1})<1} f_i^{s_{t-1}}(x_{\sigma(t)}) \\
&\le \sum_{t=8\alpha 2^{j-1}}^{8\alpha 2^j} \frac{1}{\ell_{\sigma(t)}} \sum_{i \in R_{j-1}} f_i^{s_{t-1}}(x_{\sigma(t)}) \le \sum_{t \ge 1} \frac{1}{\ell_{\sigma(t)}} \sum_{i \in R_{j-1}} f_i^{s_{t-1}}(x_{\sigma(t)}) \\
&= \sum_{i \in R_{j-1}} \sum_{k=1}^{n} \frac{f_i(T_k) - f_i(T_{k-1})}{1 - f_i(T_{k-1})}.
\end{aligned}$$

The first inequality uses the fact that any $i \notin R_{j-1}$ has f_i already covered before time $8\alpha\, 2^{j-1}$, and so it never contributes to Δ_j. In the last expression, $T_k := \{x_{\pi(1)}, \ldots, x_{\pi(k)}\} \subseteq \Delta$, the first k instantiations seen under w. The equality uses the fact that for each $\sum_{j=1}^{k-1} \ell_{\pi(j)} < t \le \sum_{j=1}^{k} \ell_{\pi(j)}$ we have $s_{t-1} = T_{k-1}$ and $\sigma(t) = \pi(k)$. Finally, by Claim 2.4, the contribution of each function $f_i \in R_{j-1}$ is at most $\alpha := 1 + \ln \frac{1}{\epsilon}$. Thus, we obtain $\Delta_j(w) \le \alpha |R_{j-1}(w)|$, and taking expectations,

$$\Delta_j \le \alpha \mathbb{E}[|R_{j-1}|]. \tag{20}$$

Lower Bound for Δ_j. Consider any $8\alpha 2^{j-1} \le t \le 8\alpha 2^j$. We lower bound G_t. Condition on s_{t-1}; this determines $\sigma(t)$ (but not $x_{\sigma(t)}$). Note that $\sum_{i=1}^{n} \ell_i \cdot \Pr[X_i \in E_j^* | s_{t-1}] \le 2^j$ by definition of E_j^* being the elements that are completely scheduled by time 2^j in OPT. Hence, we have

$$\sum_{X_i \in \mathcal{A}} \frac{\ell_i}{2^j} \cdot \Pr[X_i \in E_j^* | s_{t-1}] \le 1.$$

By applying Proposition 5.2 with the convex multipliers (over i) given above,

$$\begin{aligned}
\frac{1}{\ell_{\sigma(t)}} F^{s_{t-1}}(X_{\sigma(t)}) &\ge \sum_{X_i \in \mathcal{A}} \frac{\ell_i}{2^j} \Pr[X_i \in E_j^* | s_{t-1}] \cdot \frac{1}{\ell_i} F^{s_{t-1}}(X_i) \\
&= \frac{1}{2^j} \sum_{X_i \in \mathcal{A}} \Pr[X_i \in E_j^* | s_{t-1}] \sum_{x_i \in \Delta} \Pr[X_i = x_i | s_{t-1}] \cdot f^{s_{t-1}}(x_i) \\
&= \frac{1}{2^j} \sum_{X_i \in \mathcal{A} \setminus s_{t-1}} \Pr[X_i \in E_j^* | s_{t-1}] \sum_{x_i \in \Delta} \Pr[X_i = x_i | s_{t-1}] \cdot f^{s_{t-1}}(x_i) \\
&= \frac{1}{2^j} \sum_{X_i \in \mathcal{A} \setminus s_{t-1}} \sum_{x_i \in \Delta} \Pr[X_i \in E_j^* \wedge X_i = x_i | s_{t-1}] \cdot f^{s_{t-1}}(x_i) \\
&= \frac{1}{2^j} \sum_{w \in \Omega(s_{t-1})} \Pr[w | s_{t-1}] \sum_{X_i \in E_j^*(w) \setminus s_{t-1}} f^{s_{t-1}}(X_i(w)) \\
&= \frac{1}{2^j} \sum_{w \in \Omega(s_{t-1})} \Pr[w | s_{t-1}] \sum_{X_i \in E_j^*(w)} f^{s_{t-1}}(X_i(w)).
\end{aligned} \tag{21}$$

The first equality is by definition of $F^{s_{t-1}}(\cdot)$ from Equation (17). The second equality uses the fact that for any $X_i \in s_{t-1}$ and $x_i \in \Delta$, either $\Pr[X_i = x_i | s_{t-1}] = 0$ (if

$X_i|s_{t-1} \neq x_i$) or $f^{s_{t-1}}(x_i) = 0$ (if $X_i|s_{t-1} = x_i$). The third equality holds since the optimal policy must decide whether to schedule X_i (by time 2^j) without knowing the realization of X_i. The last equality uses $f^{s_{t-1}}(X_i) = 0$ for all $X_i \in s_{t-1}$. Now for each $w \in \Omega(s_{t-1})$, due to submodularity of the function $f^{s_{t-1}}(\cdot)$, we get

$$\sum_{X_i \in E_j^*(w)} f^{s_{t-1}}(X_i(w)) \;\geq\; f^{s_{t-1}}(E_j^*(w)) \;=\; \sum_{i \in [m],\, f_i(s_{t-1})<1} \frac{f_i(E_j^*(w) \cup s_{t-1}) - f_i(s_{t-1})}{1 - f_i(s_{t-1})}$$
$$\geq\; |C_j^*(w)| - |C(t, w)|. \tag{22}$$

Recall that $E_j^*(w)$ denotes the set of elements scheduled by time 2^j in OPT (conditional on w), as well as the realizations of these elements. The equality comes from the definition of $f^{s_{t-1}}$. The last inequality holds because $C(t, w) = \{i \in [m] : f_i(s_{t-1}) = 1\}$ and set $E_j^*(w)$ covers functions $C_j^*(w)$. Combining (21) and (22) gives

$$\frac{1}{\ell_{\sigma(t)}} F^{s_{t-1}}(X_{\sigma(t)}) \quad \geq \quad \frac{\left(\mathbb{E}\left[|C_j^*| \mid s_{t-1}\right] - \mathbb{E}\left[|C(t)| \mid s_{t-1}\right]\right)}{2^j}.$$

By deconditioning the above inequality (taking expectation over s_{t-1}) and using Equation (18), we derive:

$$G_t \;\geq\; \frac{1}{2^j} \cdot \left(\mathbb{E}[|C_j^*|] - \mathbb{E}[|C(t)|]\right) \;\geq\; \frac{1}{2^j} \cdot \left(\mathbb{E}[|C_j^*|] - \mathbb{E}[|C_j|]\right),$$

where the last inequality uses $\mathbb{E}[C(t)]$ is non-decreasing and $t \leq 8\alpha 2^j$.

Now summing over all $t \in [8\alpha 2^{j-1}, 8\alpha 2^j)$ yields:

$$\Delta_j \;=\; \sum_{t=8\alpha 2^{j-1}}^{8\alpha 2^j} G_t \;\geq\; 4\alpha\left(\mathbb{E}[|C_j^*|] - \mathbb{E}[|C_j|]\right) \;=\; 4\alpha\left(\mathbb{E}[|R_j|] - \mathbb{E}[|R_j^*|]\right). \tag{23}$$

Combining Equations (23) and (20), we obtain:

$$4\alpha(\mathbb{E}[|R_j|] - \mathbb{E}[|R_j^*|]) \;\leq\; \alpha\mathbb{E}[|R_{j-1}|],$$

which simplifies to the desired inequality in Lemma 5.1.

Using exactly the same calculations as in the proof of Theorem 2.1 from Lemma 2.2, Lemma 5.1 implies an $O(\alpha)$-approximation ratio for ALG-AG-STO. This completes the proof of Theorem 1.3.

6. CONCLUSION

In this article we considered the minimum latency submodular cover problem in general metrics, which is a common generalization of many well-studied problems. We also studied the stochastic Submodular Ranking problem, which generalizes a number of stochastic optimization problems. Both results were based on a new analysis of the algorithm for Submodular Ranking [Azar and Gamzu 2011]. Our result for stochastic Submodular Ranking is tight, and any significant improvement (more than a $\log^{\delta}|V|$ factor) of the result for minimum latency submodular cover would also improve the approximation ratio for the Group Steiner Tree, which is a long-standing open problem. An interesting open question is to obtain a poly-logarithmic approximation for stochastic minimum latency submodular cover (on general metrics), for which the main difficulty lies in designing an algorithm for a stochastic version of submodular orienteering.

APPENDIX

A. PROOF OF THEOREM 3.1

In this section, we discuss how Theorem 3.1 follows from Calinescu and Zelikovsky [2005]. The Polymatroid Steiner Tree problem (PST) considered in Calinescu and Zelikovsky [2005] is a variant of SOP where, given metric (V, d) and monotone integer-valued submodular function $f : 2^V \rightarrow \mathbb{R}_+$, the goal is to find a minimum length tree that spans a "base" of the polymatroid associated with f. Recall that a subset $S \subseteq V$ is said to be a base of the polymatroid of f if $f(S) = f(V)$. Theorem 3 in Calinescu and Zelikovsky [2005] provides an $O((\log|V|)^{2+\delta} \cdot \log f(V))$-approximation algorithm for PST, where $\delta > 0$ is any constant. The main difference from SOP is that one wants to cover the submodular function rather than maximizing the function value given a length bound. The other differences are very minor: finding a tree rather than a path, restricting to integer-valued f, and having no specified root vertex. The relation between PST and SOP is similar to that between the set-cover and maximum-coverage problems: This is why the approximation ratio in Theorem 3.1 is better by a log-factor compared to Theorem 3 in Calinescu and Zelikovsky [2005].

The algorithm in Calinescu and Zelikovsky [2005] initially transforms the metric into a rooted tree with some additional properties at the loss of an $O(\log^{1+\delta}|V|)$ approximation factor. The transformation allows the root to be specified arbitrarily. Then, "cost-efficient" trees are recursively found and concatenated until a certain condition is satisfied—the only change we need to make is when to stop. Let $T_1, T_2, \ldots$ be the trees in the order they are found. Let S_i be the set of vertices that are covered by $T_1, \ldots, T_i$; for simplicity, let $S_0 = \emptyset$. Lemma 4 in Section 3 of Calinescu and Zelikovsky [2005] states that the discovered trees have the following property:

$$\frac{c(T_i)}{f^{S_{i-1}}(S_i)} \;\leq\; O(\log|V|) \cdot \frac{c(T^*)}{f^{S_{i-1}}(S^*)}$$

for an arbitrary fixed tree T^* with S^* being the vertices that T^* spans and any monotone submodular function f. We remind the reader that $f^S(X) = f(X \cup S) - f(S)$, and $c(T)$ denotes the metric length of tree T. Set T^* to a fixed optimal solution of the SOP instance: It has length $c(T^*) \leq B$ and $g(S^*) = \mathsf{OPT}$, where g is the input function to SOP. We assume, without loss of generality, that we know the value of OPT within an arbitrary small constant factor (via a simple binary search). We set f (the input function to PST) to be $f(S) := \min\{g(S), \mathsf{OPT}\}$ for all $S \subseteq V$; clearly, f is also monotone submodular.

Let i^* be the first i such that $f(S_i) \geq f(S^*)/2$. Note that $f(S_{i^*-1}) < f(S^*)/2$; so, for any $i \leq i^*$, we have $f^{S_{i-1}}(S^*) \geq f(S^*) - f(S_{i-1}) \geq f(S^*) - f(S_{i^*-1}) > f(S^*)/2$. Then, it follows that

$$\begin{aligned}\sum_{i=1}^{i^*} c(T_i) \;&\leq\; O(\log|V|) \sum_{i=1}^{i^*} f^{S_{i-1}}(S_i) \cdot \frac{c(T^*)}{f^{S_{i-1}}(S^*)} \;\leq\; O(\log|V|) \sum_{i=1}^{i^*} f^{S_{i-1}}(S_i) \cdot 2\frac{c(T^*)}{f(S^*)}\\ &=\; O(\log|V|) \cdot f(S_{i^*}) \cdot 2\frac{c(T^*)}{f(S^*)} \;\leq\; O(\log|V|) \cdot c(T^*).\end{aligned}$$

The last inequality uses the fact that $f(S_{i^*}) \leq \mathsf{OPT}$ by definition of f.

Further, it is easy to see that one can obtain a tour with length at most $2\sum_{i=1}^{i^*} c(T_i)$ by concatenating trees $T_1, \ldots, T_{i^*}$ and doubling edges. Finally, recall that we capped the function f at OPT. The final solution quality can only be improved when the capping is removed. Hence we obtained a tour of length $O(\log^{2+\epsilon}|V|) \cdot B$ that achieves a g-function value of at least OPT/2, proving Theorem 3.1.

ACKNOWLEDGMENTS

We thank an anonymous *Transactions on Algorithms* reviewer for helpful comments on improving the presentation of Section 4.

REFERENCES

Y. Azar and I. Gamzu. 2011. Ranking with submodular valuations. In *22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1070–1079.

Y. Azar, I. Gamzu, and X. Yin. 2009. Multiple intents re-ranking. In *41st Annual ACM Symposium on Theory of Computing (STOC)*. 669–678.

N. Bansal, A. Gupta, and R. Krishnaswamy. 2010. A constant factor approximation algorithm for generalized min-sum set cover. In *21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1539–1545.

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

A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. 2010. Detecting high log-densities: An $n^{1/4}$ approximation for densest k-subgraph. In *42nd ACM Symposium on Theory of Computing (STOC)*. 201–210.

G. Calinescu and A. Zelikovsky. 2005. The polymatroid Steiner problems. *J. Combin. Optimiz.* 9, 3 (2005), 281–294.

R. D. Carr, L. Fleischer, V. J. Leung, and C. A. Phillips. 2000. Strengthening integrality gaps for capacitated network design and covering problems. In *11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 106–115.

D. Chakrabarty and C. Swamy. 2011. Facility location with client latencies: Linear programming based techniques for minimum latency problems. In *15th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 92–103.

M. Charikar, C. Chekuri, and M. Pál. 2005. Sampling bounds for stochastic optimization. In *9th International Workshop on Randomization and Computation (RANDOM)*. 257–269.

K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. 2003. Paths, trees, and minimum latency tours. In *44th Symposium on Foundations of Computer Science (FOCS)*. 36–45.

C. Chekuri, G. Even, and G. Kortsarz. 2006. A greedy approximation algorithm for the group Steiner problem. *Discr. Appl. Math.* 154, 1 (2006), 15–34.

C. Chekuri and M. Pál. 2005. A recursive greedy algorithm for walks in directed graphs. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 245–253.

J. Fakcharoenphol, C. Harrelson, and S. Rao. 2007. The k-traveling repairmen problem. *ACM Trans. Algor.* 3, 4 (2007).

J. Fakcharoenphol, S. Rao, and K. Talwar. 2004. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.* 69, 3 (2004), 485–497.

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

N. Garg, G. Konjevod, and R. Ravi. 2000. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algor.* 37, 1 (2000), 66–84.

M. X. Goemans and J. Vondrák. 2006. Stochastic covering and adaptivity. In *7th Latin American Symposium on Theoretical Informatics (LATIN)*. 532–543.

D. Golovin and A. Krause. 2010. Adaptive submodularity: A new approach to active learning and stochastic optimization. In *23rd Conference on Learning Theory (COLT)*. 333–345.

A. Guillory and J. A. Bilmes. 2011. Online submodular set cover, ranking, and repeated active learning. In *25th Annual Conference on Neural Information Processing Systems (NIPS)*. 333–345.

A. Gupta, V. Nagarajan, and R. Ravi. 2010. Approximation algorithms for optimal decision trees and adaptive TSP problems. In *37th International Colloquium on Automata, Languages and Programming (ICALP)*. 690–701.

A. Gupta and A. Srinivasan. 2006. An improved approximation ratio for the covering Steiner problem. *Theor. Comput.* 2, 1 (2006), 53–64.

E. Halperin and R. Krauthgamer. 2003. Polylogarithmic inapproximability. In *35th Annual ACM Symposium on Theory of Computing (STOC)*. 585–594.

A. J. Kleywegt, A. Shapiro, and T. Homem de Mello. 2002. The sample average approximation method for stochastic discrete optimization. *SIAM J. Optimiz.* 12, 2 (2002), 479–502.

G. Konjevod, R. Ravi, and A. Srinivasan. 2002. Approximation algorithms for the covering Steiner problem. *Rand. Struct. Algor.* 20, 3 (2002), 465–482.

Z. Liu, S. Parthasarathy, A. Ranganathan, and H. Yang. 2008. Near-optimal algorithms for shared filter evaluation in data stream systems. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 133–146.

K. Munagala, U. Srivastava, and J. Widom. 2007. Optimization of continuous queries with shared expensive filters. In *27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. 215–224.

V. Nagarajan. 2009. *Approximation Algorithms for Sequencing Problems*. Ph.D. Dissertation. Tepper School of Business, Carnegie Mellon University.

A. Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, Berlin.

M. Skutella and D. P. Williamson. 2011. A note on the generalized min-sum set cover problem. *Operat. Res. Lett.* 39, 6 (2011), 433–436.

L. A. Wolsey. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica* 2, 4 (1982), 385–393.

Received February 2013; revised June 2015; accepted August 2016

QUERIES

Q1: AU: Please spell out TSP; please spell out KRS at first occurrence in text.
Q2: AU: Please provide full mailing and email addresses for all authors.