MATHEMATICS OF OPERATIONS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0364-765X | EISSN 1526-5471 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000

Approximation Algorithms for Optimal Decision Trees and Adaptive TSP Problems

Anupam Gupta
Computer Science Department, Carnegie Mellon University, Pittsburgh, anupamg@cs.cmu.edu

Viswanath Nagarajan
Industrial and Operations Engineering Department, University of Michigan, viswa@umich.edu

R. Ravi
Tepper School of Business, Carnegie Mellon University, ravi@cmu.edu

We consider the problem of constructing *optimal decision trees*: given a collection of tests which can disambiguate between a set of m possible diseases, each test having a cost, and the a-priori likelihood of any particular disease, what is a good adaptive strategy to perform these tests to minimize the expected cost to identify the disease? This problem has been studied in several works, with $O(\log m)$-approximations known in the special cases when either costs or probabilities are uniform. In this paper, we settle the approximability of the general problem by giving a tight $O(\log m)$-approximation algorithm.

We also consider a substantial generalization, the *adaptive traveling salesman problem*. Given an underlying metric space, a random subset S of vertices is drawn from a known distribution, but S is initially unknown—we get information about whether any vertex is in S only when it is visited. What is a good adaptive strategy to visit all vertices in the random subset S while minimizing the expected distance traveled? This problem has applications in routing message ferries in ad-hoc networks, and also models switching costs between tests in the optimal decision tree problem. We give a poly-logarithmic approximation algorithm for adaptive TSP, which is nearly best possible due to a connection to the well-known group Steiner tree problem. Finally, we consider the related *adaptive traveling repairman problem*, where the goal is to compute an adaptive tour minimizing the expected sum of arrival times of vertices in the random subset S; we obtain a poly-logarithmic approximation algorithm for this problem as well.

Key words: approximation algorithms; stochastic optimization; decision trees; vehicle routing
MSC2000 subject classification: 68W25,90B06,90B15
OR/MS subject classification: Primary: Networks/graphs; secondary: Analysis of algorithms

1. Introduction

Consider the following two adaptive covering optimization problems:

- *Adaptive TSP under stochastic demands* (AdapTSP). A traveling salesperson is given a metric space (V, d) and distinct subsets $S_1, S_2, \ldots, S_m \subseteq V$ such that S_i appears with probability p_i (and $\sum_i p_i = 1$). She needs to serve requests at a random subset S of locations drawn from this distribution. However, she does not know the identity of the random subset: she can only visit locations, at which time she finds out whether or not that location is part of the subset S. What adaptive strategy should she use to minimize the expected time to serve all requests in the random set S?
- *Optimal Decision Trees.* Given a set of m diseases, there are n binary tests that can be used to disambiguate between these diseases. If the cost of performing test $t \in [n]$ is c_t, and we are given the likelihoods $\{p_j\}_{j\in[m]}$ that a typical patient has the disease j, what (adaptive) strategy should the doctor use for the tests to minimize the expected cost to identify the disease?

It can be shown that the optimal decision tree problem is a special case of the adaptive TSP problem: a formal reduction is given in Section 4. In both these problems we want to devise adaptive strategies, which take into account the revealed information in the queries so far (e.g., locations already visited, or tests already done) to determine the future course of action. Such an *adaptive solution* corresponds naturally to a decision tree, where nodes encode the current "state" of the solution and branches represent observed random outcomes: see Definition 2 for a formal definition. A simpler class of solutions, that have been useful in some other adaptive optimization problems, eg. [11, 20, 2], are *non-adaptive solutions*, which are specified by just an ordered list of actions. However there are instances for both the above problems where the optimal adaptive solution costs much less than the optimal non-adaptive solution. Hence it is essential that we find good adaptive solutions.

The *optimal decision tree problem* has long been studied, its NP-hardness was shown by Hyafil and Rivest in 1976 [25] and many references and applications can be found in [33]. There have been a large number of papers providing algorithms for this problem [15, 30, 28, 10, 1, 5, 33, 21]. The best results yield approximation ratios of $O\left(\log \frac{1}{p_{\min}}\right)$ and $O\left(\log(m \frac{c_{max}}{c_{min}})\right)$, where p_{min} is the minimum non-zero probability and c_{max} (resp. c_{min}) is the maximum (resp. minimum) cost. In the special cases when the likelihoods $\{p_j\}$ or the costs $\{c_t\}$ are all polynomially bounded in m, these imply an $O(\log m)$-approximation algorithm. However, there are instances (when probabilities and costs are exponential) demonstrating an $\Omega(m)$ approximation guarantee for all previous algorithms. On the hardness side, an $\Omega(\log m)$ hardness of approximation (assuming $P \neq NP$) is known for the optimal decision tree problem [5]. While the existence of an $O(\log m)$-approximation algorithm for the general optimal decision tree problem has been posed as an open question, it has not been answered prior to this work.

Optimal decision tree is also a basic problem in *average-case active learning* [10, 33, 21]. In this application, there is a set of n data points, each of which is associated with a $+$ or $-$ label. The labels are initially unknown. A classifier is a partition of the data points into $+$ and $-$ labels. The true classifier h^* is the partition corresponding to the actual data labels. The learner knows beforehand, a "hypothesis class" H consisting of m classifiers; it is assumed that the true classifier $h^* \in H$. Furthermore, in the average case model, there is a known distribution π of h^* over H. The learner wants to determine h^* by querying labels at various points. There is a cost c_t associated with querying the label of each data point t. An active learning strategy involves adaptively querying labels of data points until $h^* \in H$ is identified. The goal is to compute a strategy that minimizes the expectation (over π) of the cost of all queried points. This is precisely the optimal decision tree problem, with points being tests and classifiers corresponding to diseases.

Apart from being a natural adaptive routing problem, AdapTSP has many applications in the setting of message ferrying in ad-hoc networks [38, 35, 39, 40, 24]. We cite two examples below:

- *Data collection in sparse sensor networks (see eg. [35]).* A collection of sensors is spread over a large geographic area, and one needs to periodically gather sensor data at a base station. Due to the power and cost overheads of setting up a communication network between the sensors, the data collection is instead performed by a mobile device (the message ferry) that travels in this space from/to the base station. On any given day, there is a known distribution $\mathcal{D}$ of the subset S of sensors that contain new information: this might be derived from historical data or domain experts. The routing problem for the ferry then involves computing a tour (originating from the base station) that visits all sensors in S, at the minimum expected cost.
- *Disaster management (see eg. [39]).* Consider a post-disaster situation, in which usual communication networks have broken down. In this case, vehicles can be used in order to visit locations and assess the damage. Given a distribution of the set of affected locations, the goal here is to route a vehicle that visits all affected locations as quickly as possible in expectation.

In both these applications, due to the absence of a direct communication network, the information at any location is obtained only when it is visited: this is precisely the AdapTSP problem.

1.1. Our Results and Techniques In this paper, we settle the approximability of the optimal decision tree problem:

THEOREM 1. *There is an $O(\log m)$-approximation algorithm for the optimal decision tree problem with arbitrary test costs and arbitrary probabilities, where m is the number of diseases. The problem admits the same approximation ratio even when the tests have non-binary outcomes.*

In fact, this result arises as a special case of the following theorem:

THEOREM 2. *There is an $O(\log^2 n \log m)$-approximation algorithm for the adaptive Traveling Salesman Problem, where n is the number of vertices and m the number of scenarios in the demand distribution.*

To solve the AdapTSP problem, we first solve the "isolation problem", which seeks to identify which of the m scenarios has materialized. Once we know the scenario we can visit its vertices using any constant-factor approximation algorithm for TSP. The high-level idea behind our algorithm for the isolation problem is this—suppose each vertex lies in at most half the scenarios; then if we visit one vertex in each of the m scenarios using a short tour, which is an instance of the *group Steiner tree* problem[1], we'd notice at least one of these vertices to have a demand; this would reduce the number of possible scenarios by at least 50% and we can recursively run the algorithm on the remaining scenarios. This is an over-simplified view, and there are many details to handle: we need not visit all scenarios—visiting all but one allows us to infer the last one by exclusion; the expectation in the objective function means we need to solve a *minimum-sum* version of group Steiner tree; not all vertices need lie in less than half the scenarios. Another major issue is that we do not want our performance to depend on the magnitude of the probabilities, as some of them may be exponentially small. Finally, we need to charge our cost directly against the optimal decision tree. All these issues can indeed be resolved to obtain Theorem 2.

The algorithm for the isolation problem involves an interesting combination of ideas from the group Steiner [16, 6] and minimum latency TSP [3, 7, 12] problems—it uses a greedy approach that is greedy with respect to two different criteria, namely the probability measure and the number of scenarios. This idea is formalized in our algorithm for the *partial latency* group Steiner (LPGS) problem, which is a key subroutine for IsoProb. While this LPGS problem is harder to approximate than the standard group Steiner tree (see Section 2), for which $O(\log^2 n \log m)$ is the best approximation ratio, we show that it admits a better $\big(O(\log^2 n), 4\big)$ *bicriteria* approximation algorithm. Moreover, even this bicriteria approximation guarantee for LPGS suffices to obtain an $O(\log^2 n \cdot \log m)$-approximation algorithm for IsoProb.

We also show that both AdapTSP and the isolation problem are $\Omega(\log^{2-\epsilon} n)$ hard to approximate even on tree metrics; our results are essentially best possible on such metrics, and we lose an extra logarithmic factor to go to general metrics, as in the group Steiner tree problem. Moreover, any improvement to the result in Theorem 2 would lead to a similar improvement for the group Steiner tree problem [16, 23, 8] which is a long-standing open question.

For the optimal decision tree problem, we show that we can use a variant of minimum-sum set cover [14] which is the special case of LPGS on star-metrics. This avoids an $O(\log^2 n)$ loss in the approximation guarantee, and hence gives us an $O(\log m)$-approximation algorithm which is best possible [5]. Although this variant of min-sum set cover is $\Omega(\log m)$-hard to approximate (it generalizes set cover as shown in Section 2), we again give a constant factor bicriteria approximation

[1] In the group Steiner tree problem [16] the input is a metric (V, d) with root $r \in V$ and groups $\{X_i \subseteq V\}$ of vertices; the goal is to compute a minimum length tour originating from r that visits at least one vertex of each group.

algorithm, which leads to the $O(\log m)$-approximation for optimal decision tree. Our result further reinforces the close connection between the min-sum set cover problem and the optimal decision tree problem that was first noticed by [5].

Finally, we consider the related adaptive traveling repairman problem (AdapTRP), which has the same input as AdapTSP, but the objective is to minimize the expected sum of arrival times at vertices in the materialized demand set. In this setting, we cannot first isolate the scenario and then visit all its nodes, since a long isolation tour may negatively impact the arrival times. So AdapTRP (unlike AdapTSP) cannot be reduced to the isolation problem. However, we show that our techniques for AdapTSP are robust, and can be used to obtain:

THEOREM 3. *There is an $O(\log^2 n \log m)$-approximation algorithm for the adaptive Traveling Repairman Problem, where n is the number of vertices and m the number of scenarios in the demand distribution.*

Paper Outline: The results on the isolation problem appear in Section 3. We obtain the improved approximation algorithm for optimal decision tree in Section 4. The algorithm for the adaptive traveling salesman problem is in Section 5; Appendix A contains a nearly matching hardness of approximation result. Finally, Section 6 is on the adaptive traveling repairman problem.

1.2. Other Related Work The optimal decision tree problem has been studied earlier by many authors, with algorithms and hardness results being shown by [15, 25, 30, 28, 1, 10, 5, 4, 21]. As mentioned above, the algorithms in these papers gave $O(\log m)$-approximation ratios only when the probabilities or costs (or both) are polynomially-bounded. The early papers on optimal decision tree considered tests with only binary outcomes. More recently, [5] studied the generalization with $K \geq 2$ outcomes per test, and gave an $O(\log K \cdot \log m)$-approximation under uniform costs. Subsequently, [4] improved this bound to $O(\log m)$, again under uniform costs. Later, [21] gave an algorithm under arbitrary costs and probabilities, achieving an approximation ratio of $O\left(\log \frac{1}{p_{\min}}\right)$ or $O\left(\log(m \frac{c_{max}}{c_{min}})\right)$. This is the previous best approximation guarantee; see also Table 1 in [21] for a summary of these results. We note that in terms of the number m of diseases, the previous best approximation guarantee is only $\Omega(m)$. On the other hand, there is an $\Omega(\log m)$ hardness of approximation for the optimal decision tree problem [5]. Our $O(\log m)$-approximation algorithm for arbitrary costs and probabilities solves an open problem from these papers. A crucial aspect of this algorithm is that it is non greedy. All previous results were based on variants of a greedy algorithm.

There are many results on adaptive optimization dealing with covering problems. E.g., [18] considered the adaptive set-cover problem; they gave an $O(\log n)$-approximation when sets may be chosen multiple times, and an $O(n)$-approximation when each set may be chosen at most once. The latter approximation ratio was improved in [31] to $O(\log^2 n \log m)$, and subsequently to the best-possible $O(\log n)$-approximation ratio by [29], also using a greedy algorithm. In recent work [19] generalized adaptive set-cover to a setting termed 'adaptive submodularity', and gave many applications. In all these problems, the adaptivity-gap (ratio between optimal adaptive and non-adaptive solutions) is large, as is the case for the problems considered in this paper, and so the solutions need to be inherently adaptive.

The AdapTSP problem is related to universal TSP [27, 22] and *a priori* TSP [26, 34, 36] only in spirit—in both the universal and *a priori* TSP problems, we seek a master tour which is shortcut once the demand set is known, and the goal is to minimize the worst-case or expected length of the shortcut tour. The crucial difference is that the demand subset is revealed *in toto* in these two problems, leaving no possibility of adaptivity—this is in contrast to the slow revelation of the demand subset that occurs in AdapTSP.

2. Preliminaries We work with a finite metric (V, d) that is given by a set V of n vertices and distance function $d : V \times V \to \mathbb{R}_+$. As usual, we assume that the distance function is symmetric and satisfies the triangle inequality. For any integer $t \geq 1$, we let $[t] := \{1, 2, \ldots, t\}$.

DEFINITION 1 (r-TOUR). Given a metric (V, d) and vertex $r \in V$, an *r-tour* is any sequence $(r = u_0, u_1, \cdots, u_k = r)$ of vertices that begins and ends at r. The length of such an r-tour is $\sum_{i=1}^{k} d(u_i, u_{i-1})$, the total length of all edges in the tour.

Throughout this paper, we deal with demand distributions over vertex-subsets that are specified explicitly. A demand distribution $\mathcal{D}$ is specified by m distinct subsets $\{S_i \subseteq V\}_{i=1}^{m}$ having associated probabilities $\{p_i\}_{i=1}^{m}$ such that $\sum_{i=1}^{m} p_i = 1$. This means that the realized subset $D \subseteq V$ of demand-vertices will always be one of $\{S_i\}_{i=1}^{m}$, where $D = S_i$ with probability p_i (for all $i \in [m]$). We also refer to the subsets $\{S_i\}_{i=1}^{m}$ as *scenarios*. The following definition captures adaptive strategies.

DEFINITION 2 (DECISION TREE). A decision tree T in metric (V, d) is a rooted binary tree where each non-leaf node of T is labeled with a vertex $u \in V$, and its two children u_{yes} and u_{no} correspond to the subtrees taken if there is demand at u or if there is no demand at u. Thus given any realized demand $D \subseteq V$, a unique path T_D is followed in T from the root down to a leaf.
Depending on the problem under consideration, there are additional constraints on decision tree T and the *expected cost* of T is also suitably defined. There is a (problem-specific) cost C_i associated with each scenario $i \in [m]$ that depends on path T_{S_i}, and the expected cost of T (under distribution $\mathcal{D}$) is then $\sum_{i=1}^{m} p_i \cdot C_i$. For example in AdapTSP, cost C_i corresponds to the length of path T_{S_i}.

Since we deal with explicitly specified demand distributions $\mathcal{D}$, all decision trees we consider will have size polynomial in m (support size of $\mathcal{D}$) and n (number of vertices).

Adaptive Traveling Salesman This problem consists of a metric (V, d) with root $r \in V$ and a demand distribution $\mathcal{D}$ over subsets of vertices. The information on whether or not there is demand at a vertex v is obtained only when that vertex v is visited. The objective is to find an adaptive strategy that minimizes the expected time to visit all vertices of the realized scenario drawn from $\mathcal{D}$.

We assume that the distribution $\mathcal{D}$ is specified *explicitly* with a support-size of m. This allows us to model demand distributions that are arbitrarily correlated across vertices. We note however that the running time and performance of our algorithm will depend on the support size. The most general setting would be to consider black-box access to the distribution $\mathcal{D}$: however, as shown in [32], in this setting there is no $o(n)$-approximation algorithm for AdapTSP that uses a polynomial number of samples from the distribution. One could also consider AdapTSP under independent demand distributions. In this case there is a trivial constant-factor approximation algorithm, that visits all vertices having non-zero probability along an approximately minimum TSP tour; note that any feasible solution must visit all vertices with non-zero probability as otherwise (due to the independence assumption) there would be a positive probability of not satisfying a demand.

DEFINITION 3 (ADAPTIVE TSP). The input is a metric (V, d), root $r \in V$ and demand distribution $\mathcal{D}$ given by m distinct subsets $\{S_i \subseteq V\}_{i=1}^{m}$ with probabilities $\{p_i\}_{i=1}^{m}$ (where $\sum_{i=1}^{m} p_i = 1$). The goal in AdapTSP is to compute a decision tree T in metric (V, d) such that:

- the root of T is labeled with the root vertex r, and
- for each scenario $i \in [m]$, the path T_{S_i} followed on input S_i contains all vertices in S_i.

The objective function is to minimize the expected tour length $\sum_{i=1}^{m} p_i \cdot d(T_{S_i})$, where $d(T_{S_i})$ is the length of the tour that starts at r, visits the vertices on path T_{S_i} in that order, and returns to r.

Isolation Problem This is closely related to AdapTSP. The input is the same as AdapTSP, but the goal is just to identify the unique scenario that has materialized, and not to visit all the vertices in the realized scenario.

DEFINITION 4 (ISOLATION PROBLEM). Given metric (V, d), root r and demand distribution $\mathcal{D}$, the goal in IsoProb is to compute a decision tree T in metric (V, d) such that:

- the root of T is labeled with the root vertex r, and
- for each scenario $i \in [m]$, the path T_{S_i} followed on input S_i ends at a distinct leaf-node of T.

The objective is to minimize the expected tour length $\mathsf{IsoTime}(T) := \sum_{i=1}^{m} p_i \cdot d(T_{S_i})$, where $d(T_{S_i})$ is the length of the r-tour that visits the vertices on path T_{S_i} in that order, and returns to r.

The only difference between IsoProb and AdapTSP is that the tree path T_{S_i} in IsoProb need not contain all vertices of S_i, and the paths for different scenarios must end at distinct leaf-nodes. In Section 5 we show that any approximation algorithm for IsoProb leads to an approximation algorithm for AdapTSP. So we focus on designing algorithms for IsoProb.

Optimal Decision Tree This problem involves identifying a random disease from a set of possible diseases using binary tests.

Definition 5 (Optimal Decision Tree). The input is a set of m diseases with probabilities $\{p_i\}_{i=1}^{m}$ that sum to one, and a collection $\{T_j \subseteq [m]\}_{j=1}^{n}$ of n binary tests with costs $\{c_j\}_{j=1}^{n}$. There is exactly one realized disease: each disease $i \in [m]$ occurs with probability p_i. Each test $j \in [n]$ returns a positive outcome for subset T_j of diseases and returns a negative outcome for the rest $[m] \setminus T_j$. The goal in ODT is to compute a decision tree Q where each internal node is labeled by a test and has two children corresponding to positive/negative test outcomes, such that for each $i \in [m]$ the path Q_i followed under disease i ends at a distinct leaf node of Q. The objective is to minimize the expected cost $\sum_{i=1}^{m} p_i \cdot c(Q_i)$ where $c(Q_i)$ is the sum of test-costs along path Q_i.

Notice that the optimal decision tree problem is exactly IsoProb on a weighted star metric. Indeed, given an instance of ODT, consider a metric (V, d) induced by a weighted star with center r and n leaves corresponding to the tests. For each $j \in [n]$, we set $d(r, j) = \frac{c_j}{2}$. The demand scenarios are as follows: for each $i \in [m]$ scenario i has demands $S_i = \{j \in [n] \mid i \in T_j\}$. It is easy to see that this IsoProb instance corresponds exactly to the optimal decision tree instance. See Section 4 for an example. So any algorithm for IsoProb on star-metrics can be used to solve ODT as well.

Useful Deterministic Problems Recall that the group Steiner tree problem [16, 23] consists of a metric (V, d), root $r \in V$ and g groups of vertices $\{X_i \subseteq V\}_{i=1}^{g}$, and the goal is to find an r-tour of minimum length that contains at least one vertex from each group $\{X_i\}_{i=1}^{g}$. Our algorithms for the above stochastic problems rely on solving some variants of group Steiner tree.

Definition 6 (Group Steiner Orienteering). The input is a metric (V, d), root $r \in V$, g groups of vertices $\{X_i \subseteq V\}_{i=1}^{g}$ with associated profits $\{\phi_i\}_{i=1}^{g}$ and a length bound B. The goal in GSO is to compute an r-tour of length at most B that maximizes the total profit of covered groups. A group $i \in [g]$ is covered if any vertex from X_i is visited by the tour.

An algorithm for GSO is said to be a (β, γ)-bicriteria approximation algorithm if on any instance of the problem, it finds an r-tour of length at most $\gamma \cdot B$ that has profit at least $\frac{1}{\beta}$ times the optimal (which has length at most B).

Definition 7 (Partial Latency Group Steiner). The input is a metric (V, d), g groups of vertices $\{X_i \subseteq V\}_{i=1}^{g}$ with associated weights $\{w_i\}_{i=1}^{g}$, root $r \in V$ and a target $h \leq g$. The goal in LPGS is to compute an r-tour τ that covers at least h groups and minimizes the weighted sum of arrival times over all groups. The *arrival time* of group $i \in [g]$ is the length of the shortest prefix of tour τ that contains an X_i-vertex; if the group is not covered, its arrival time is set to be the entire tour-length. The LPGS objective is termed *latency*, i.e.

$$\text{latency}(\tau) \quad = \quad \sum_{i \text{ covered}} w_i \cdot \text{arrival time}_\tau(X_i) \; + \; \sum_{i \text{ uncovered}} w_i \cdot \text{length}(\tau). \tag{1}$$

An algorithm for LPGS is said to be a (ρ, σ)-bicriteria approximation algorithm if on any instance of the problem, it finds an r-tour that covers at least h/σ groups and has latency at most ρ times the optimal (which covers at least h groups). The reason we focus on a bicriteria approximation

for LPGS is that it is harder to approximate than the group Steiner tree problem (see below) and we can obtain a better bicriteria guarantee for LPGS.

To see that LPGS is at least as hard to approximate as the group Steiner tree problem, consider an arbitrary instance of group Steiner tree with metric (V,d), root $r \in V$ and g groups $\{X_i \subseteq V\}_{i=1}^{g}$. Construct an instance of LPGS as follows. The vertices are $V' = V \cup \{u\}$ where u is a new vertex. Let $L := n^2 \cdot \max_{a,b} d(a,b)$. The distances in metric (V',d') are: $d'(a,b) = d(a,b)$ if $a,b \in V$ and $d'(a,u) = L + d(a,r)$ if $a \in V$. There are $g' = g+1$ groups with $X'_i = X_i$ for $i \in [g]$ and $X'_{g+1} = \{u\}$. The target $h = g$. The weights are $w_i = 0$ for $i \in [g]$ and $w_{g+1} = 1$. Since the distance from r to u is very large, no approximately optimal LPGS solution will visit u. So any such LPGS solution covers *all* the groups $\{X'_i\}_{i=1}^{g}$ and has latency equal to the length of the solution (as group X'_{g+1} has weight one and all others have weight zero). This reduction also shows that LPGS on weighted star-metrics (which is used in the ODT algorithm) is at least as hard to approximate as set cover: this is because when metric (V,d) is a star-metric with center r, so is the new metric (V',d').[2]

3. Approximation Algorithm for the Isolation Problem Recall that an instance of IsoProb is specified by a metric (V,d), a root vertex $r \in V$, and m scenarios $\{S_i\}_{i=1}^{m}$ with associated probability values $\{p_i\}_{i=1}^{m}$. The main result of this section is:

THEOREM 4. *If there is a $(4,\gamma)$-bicriteria approximation algorithm for group Steiner orienteering then there is an $O(\gamma \cdot \log m)$-approximation algorithm for the isolation problem.*

We prove this in two steps. First, in Subsection 3.1 we show that a $(\rho,4)$-bicriteria approximation algorithm for LPGS can be used to obtain an $O(\rho \cdot \log m)$-approximation algorithm for IsoProb. Then, in Subsection 3.2 we show that any $(4,\gamma)$-bicriteria approximation algorithm for GSO leads to an $(O(\gamma),4)$-bicriteria approximation algorithm for LPGS.

Note on reading this section: While the results of this section apply to the isolation problem on general metrics, readers interested in just the optimal decision tree problem need to only consider weighted star metrics (as discussed after Definition 5). In the ODT case, we have the following simplifications (1) a tour is simply a sequence of tests, (2) the tour length is the sum of test costs in the sequence, and (3) concatenating tours corresponds to concatenating test sequences.

3.1. Algorithm for IsoProb using LPGS Recall the definition of IsoProb and LPGS from Section 2. Here we will prove:

THEOREM 5. *If there is a $(\rho,4)$-bicriteria approximation algorithm for* LPGS *then there is an $O(\rho \cdot \log m)$-approximation algorithm for* IsoProb.

We first give a high-level description of our algorithm. The algorithm uses an iterative approach and maintains a candidate set of scenarios that contains the realized scenario. In each iteration, the algorithm eliminates a constant fraction of scenarios from the candidate set. So the number of iterations will be bounded by $O(\log m)$. In each iteration we solve a suitable instance of LPGS in order to refine the candidate set of scenarios.

Single iteration of* IsoProb *algorithm As mentioned above, we use LPGS in each iteration of the IsoProb algorithm- we now describe how this is done. At the start of each iteration, our algorithm maintains a candidate set $M \subseteq [m]$ of scenarios that contains the realized scenario. The probabilities associated with the scenarios $i \in M$ are not the original p_is but their conditional probabilities $q_i := \frac{p_i}{\sum_{j \in M} p_j}$. The algorithm Partition (given as Algorithm 1) uses LPGS to compute an r-tour τ such that after observing the demands on τ, the number of scenarios consistent with these observations is guaranteed to be a constant factor smaller than $|M|$.

[2] Recall that group Steiner tree on star-metrics is equivalent to the set cover problem.

To get some intuition for this algorithm, consider the simplistic case when there is a vertex $u \in V$ located near the root r such that $\approx 50\%$ of the scenarios in M contain it. Then just visiting vertex u would reduce the number candidate scenarios by $\approx 50\%$, irrespective of the observation at u, giving us the desired notion of progress. However, each vertex may give a very unbalanced partition of M: so we may have to visit multiple vertices before ensuring that the number of candidate scenarios reduces by a constant factor. Moreover, some vertices may be too expensive to visit from r: so we need to carefully take the metric into account in choosing the set of vertices to visit. Addressing these issues is precisely where the LPGS problem comes in.

Algorithm 1 Algorithm Partition($\langle M, \{q_i\}_{i \in M} \rangle$)

1: **let** $g = |M|$. For each $v \in V$, define $F_v := \{i \in M \mid v \in S_i\}$, and $D_v := \begin{cases} F_v & \text{if } |F_v| \le g/2 \\ M \setminus F_v & \text{if } |F_v| > g/2 \end{cases}$
2: **for each** $i \in M$, set $X_i \leftarrow \{v \in V \mid i \in D_v\}$.
3: **run** the $(\rho, 4)$-bicriteria approximation algorithm for LPGS on the instance with metric (V, d), root r, groups $\{X_i\}_{i \in M}$ with weights $\{q_i\}_{i \in M}$, and target $h := g - 1$.
let $\tau := r, v_1, v_2, \cdots, v_{t-1}, r$ be the r-tour returned.
4: **let** $\{P_k\}_{k=1}^t$ be the partition of M where $P_k := \begin{cases} D_{v_k} \setminus \left(\cup_{j<k} D_{v_j}\right) & \text{if } 1 \le k \le t-1 \\ M \setminus \left(\cup_{j<t} D_{v_j}\right) & \text{if } k = t \end{cases}$
5: **return** tour τ and the partition $\{P_k\}_{k=1}^t$.

Note that the information at any vertex v corresponds to a bi-partition $(F_v, M \setminus F_v)$ of the scenario set M, with scenarios F_v having demand at v and scenarios $M \setminus F_v$ having no demand at v. So either the presence of demand or the absence of demand reduces the number of candidate scenarios by half (and represents progress). To better handle this asymmetry, Step 1 associates vertex v with subset D_v which is the smaller of $\{F_v, M \setminus F_v\}$; this corresponds to the set of scenarios under which just the observation at v suffices to reduce the number of candidate scenarios below $|M|/2$ (and represents progress). In Steps 2 and 3, we view vertex v as covering the scenarios D_v.

The overall algorithm for IsoProb Here we describe how the different iterations are combined to solve IsoProb. The final algorithm IsoAlg (given as Algorithm 2) is described in a recursive manner where each "iteration" is a new call to IsoAlg. As mentioned earlier, at the start of each iteration, the algorithm maintains a candidate set $M \subseteq [m]$ of scenarios such that the realized scenario lies in M. Upon observing demands along the tour produced by algorithm Partition, a new set $M' \subseteq M$ containing the realized scenario is identified such that the number of candidate scenarios reduces by a constant factor (specifically $|M'| \le \frac{7}{8} \cdot |M|$). Then IsoAlg recurses on scenarios M', which corresponds to the next iteration. After $O(\log m)$ such iterations the realized scenario would be correctly identified.

Algorithm 2 Algorithm IsoAlg$\langle M, \{q_i\}_{i \in M} \rangle$

1: If $|M| = 1$, return this unique scenario as realized.
2: **run** Partition$\langle M, \{q_i\}_{i \in M} \rangle$
let $\tau = (r, v_1, v_2, \cdots, v_{t-1}, r)$ be the r-tour and $\{P_k\}_{k=1}^t$ be the partition of M returned.
3: **let** $q'_k := \sum_{i \in P_k} q_i$ **for all** $k = 1 \ldots t$.
4: **traverse** tour τ and return directly to r after visiting the first (if any) vertex v_{k^*} (for $k^* \in [t-1]$) that determines that the realized scenario is in $P_{k^*} \subseteq M$. If there is no such vertex until the end of the tour τ, then set $k^* \leftarrow t$.
5: **run** IsoAlg$\langle P_{k^*}, \{\frac{q_i}{q'_{k^*}}\}_{i \in P_{k^*}} \rangle$ to isolate the realized scenario within the subset P_{k^*}.

Note that the adaptive Algorithm IsoAlg implicitly defines a decision tree too: indeed, we create a path $(r, v_1, v_2, \cdots, v_{t-1}, v_t = r)$, and hang the subtrees created in the recursive call on each instance $\langle P_k, \{\frac{q_i}{q'_k}\}\rangle$ from the respective node v_k. See also Figure 1.

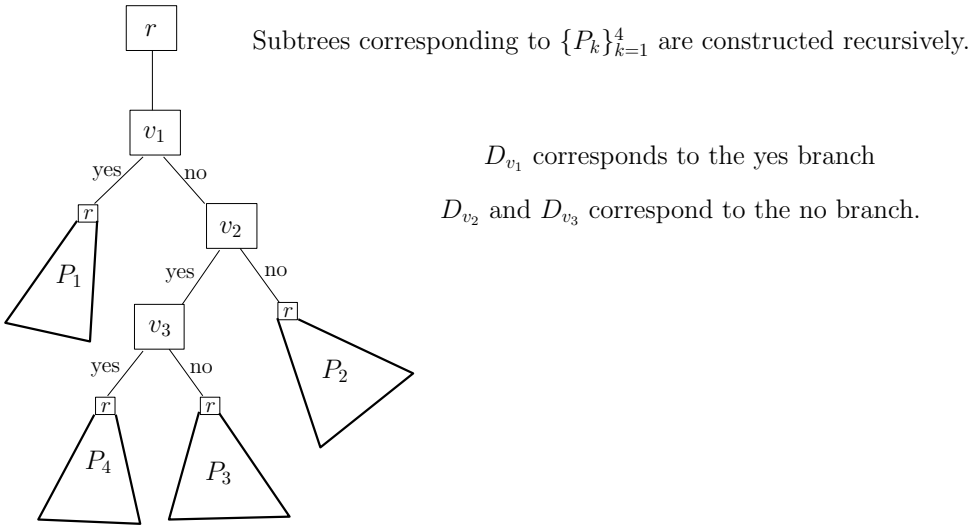


FIGURE 1. Example of decision tree in single iteration using tour $\tau = (r, v_1, v_2, v_3, r)$.

Analysis The rest of this subsection analyzes IsoAlg and proves Theorem 5. We first provide an outline of the proof. It is easy to show that IsoAlg correctly identifies the realized scenario after $O(\log m)$ iterations: this is shown formally in Claim 5. We relate the objective values of the LPGS and IsoProb instances in two steps: Claim 1 shows that LPGS has a smaller optimal value than IsoProb, and Claim 3 shows that any approximate LPGS solution can be used to construct a *partial* IsoProb solution incurring the same cost (in expectation). Since different iterations of IsoAlg deal with different sub-instances of IsoProb, we need to relate the optimal cost of these sub-instances to that of the original instance: this is done in Claim 4.

Recall that the original instance of IsoProb is defined on metric (V, d), root r and set $\{S_i\}_{i=1}^m$ of scenarios with probabilities $\{p_i\}_{i=1}^m$. IsoAlg works with many sub-instances of the isolation problem. Such an instance $\mathcal{J}$ is specified by a subset $M \subseteq [m]$ which implicitly defines (conditional) probabilities $q_i = \frac{p_i}{\sum_{j \in M} p_j}$ for all $i \in M$. In other words, $\mathcal{J}$ involves identifying the realized scenario *conditioned on* it being in set M (the metric and root remain the same as the original instance). Let $\mathsf{IsoTime}^*(\mathcal{J})$ denote the optimal value of any instance $\mathcal{J}$.

CLAIM 1. *For any instance $\mathcal{J} = \langle M, \{q_i\}_{i \in M}\rangle$, the optimal value of the* LPGS *instance considered in Step 3 of algorithm* $\mathsf{Partition}(\mathcal{J})$ *is at most* $\mathsf{IsoTime}^*(\mathcal{J})$.

Proof: Let T be an optimal decision tree corresponding to IsoProb instance $\mathcal{J}$, and hence $\mathsf{IsoTime}^*(\mathcal{J}) = \mathsf{IsoTime}(T)$. Note that by definition of the sets $\{F_v\}_{v \in V}$, any internal node in T labeled vertex v has its two children v_{yes} and v_{no} corresponding to the realized scenario being in F_v and $M \setminus F_v$ (respectively); and by definition of $\{D_v\}_{v \in V}$, nodes v_{yes} and v_{no} correspond to the realized scenario being in D_v and $M \setminus D_v$ (now not necessarily in that order).

We now define an r-tour σ based on a specific root-leaf path in T. Consider the root-leaf path that at any node labeled v, moves to the child v_{yes} or v_{no} that corresponds to $M \setminus D_v$, until it reaches a leaf-node ℓ. Let $r, u_1, u_2, \cdots, u_j$ denote the sequence of vertices in this root-leaf path, and define r-tour $\sigma = \langle r, u_1, u_2, \cdots, u_j, r\rangle$. Since T is a feasible decision tree for the isolation instance, there is at most one scenario $a \in M$ such that the path T_{S_a} traced in T under demands S_a ends

at leaf-node ℓ. In other words, every scenario $b \in M \setminus \{a\}$ gives rise to a root-leaf path T_{S_b} that diverges from the root-ℓ path. By our definition of the root-ℓ path, the scenarios that diverge from it are precisely $\cup_{k=1}^{j} D_{u_k}$, and so $\cup_{k=1}^{j} D_{u_k} = M \setminus \{a\}$.

Next, we show that σ is a feasible solution to the LPGS instance in Step 3. By definition of the groups $\{X_i\}_{i \in M}$ (Step 2 of Algorithm 1), it follows that tour σ covers groups $\cup_{k=1}^{j} D_{u_k}$. So the number of groups covered is at least $|M| - 1 = h$, and σ is a feasible LPGS solution.

Finally, we bound the LPGS objective value of σ in terms of the isolation cost $\mathsf{IsoTime}(T)$. To reduce notation let $u_0 = r$ below. The arrival times in tour σ are:

$$\text{arrival time}_\sigma(X_i) = \begin{cases} \sum_{s=1}^{k} d(u_{s-1}, u_s) & \text{if } i \in D_{u_k} \setminus \cup_{s=1}^{k-1} D_{u_s}, \text{ for } k = 1, \cdots, j \\ \text{length}(\sigma) & \text{if } i = a \end{cases}$$

Fix any $k = 1, \cdots, j$. For any scenario $i \in D_{u_k} \setminus \cup_{s=1}^{k-1} D_{u_s}$, the path T_{S_i} traced in T contains the prefix labeled $r, u_1, \cdots, u_k$ of the root-ℓ path; so $d(T_{S_i}) \geq \sum_{s=1}^{k} d(u_{s-1}, u_s) = \text{arrival time}_\sigma(X_i)$. Moreover, for scenario a which is the only scenario not in $\cup_{k=1}^{j} D_{u_k}$, we have $d(T_{S_a}) = \text{length}(\sigma) = \text{arrival time}_\sigma(X_i)$. Now by (1), $\text{latency}(\sigma) \leq \sum_{i \in M} q_i \cdot d(T_{S_i}) = \mathsf{IsoTime}(T) = \mathsf{IsoTime}^*(\mathcal{J})$. □

If we use a $(\rho, 4)$-bicriteria approximation algorithm for LPGS, we get the following claim:

Claim 2. *For any instance $\mathcal{J} = \langle M, \{q_i\}_{i \in M} \rangle$, the latency of tour τ returned by Algorithm* Partition *is at most $\rho \cdot \mathsf{IsoTime}^*(\mathcal{J})$. Furthermore, the resulting partition $\{P_k\}_{k=1}^{t}$ has each $|P_k| \leq \frac{7}{8}|M|$ for each $k \in [t]$, when $|M| \geq 2$.*

Proof: By Claim 1, the optimal value of the LPGS instance in Step 3 of algorithm Partition is at most $\mathsf{IsoTime}^*(\mathcal{J})$; now the $(\rho, 4)$-bicriteria approximation guarantee implies that the latency of the solution tour τ is at most ρ times that. This proves the first part of the claim.

Consider $\tau := \langle r = v_0, v_1, \cdots, v_{t-1}, v_t = r \rangle$ the tour returned by the LPGS algorithm in Step 3 of algorithm Partition; and $\{P_k\}_{k=1}^{t}$ the resulting partition. The $(\rho, 4)$-bicriteria approximation guarantee implies that the number of groups covered by τ is $|\cup_{k=1}^{t-1} D_{v_k}| \geq \frac{h}{4} = \frac{|M|-1}{4} \geq \frac{|M|}{8}$ (when $|M| \geq 2$). By definition of the sets D_v, it holds that $|D_v| \leq |M|/2$ for all $v \in V$. Since all but the last part P_t is a subset of some D_v, it holds that $|P_k| \leq \frac{|M|}{2}$ for $1 \leq k \leq t-1$. Moreover, the set P_t has size $|P_t| = |M \setminus (\cup_{j<t} D_{v_j})| \leq \frac{7}{8}|M|$. This proves the second part of the claim. □

Of course, we don't really care about the latency of the tour *per se*, we care about the expected cost incurred in isolating the realized scenario. But the two are related (by their very construction), as the following claim formalizes:

Claim 3. *At the end of Step 4 of* $\mathsf{IsoAlg}\langle M, \{q_i\}_{i \in M} \rangle$*, the realized scenario lies in P_{k^*}. The expected distance traversed in this step is at most $2\rho \cdot \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i \in M} \rangle)$.*

Proof: Consider the tour $\tau := \langle r = v_0, v_1, \cdots, v_{t-1}, v_t = r \rangle$ returned by the Partition algorithm. Recall that visiting any vertex v reveals whether the scenario lies in D_v, or in $M \setminus D_v$. In step 4 of algorithm IsoAlg, we traverse τ and one of the following happens:

- $1 \leq k^* \leq t-1$. Tour returns directly to r from the first vertex v_k (for $1 \leq k \leq t-1$) such that the realized scenario lies in D_{v_k}; here $k = k^*$. Since the scenario did not lie in any earlier D_{v_j} for $j < k$, the definition of $P_k = D_{v_k} \setminus (\cup_{j<k} D_{v_j})$ gives us that the realized scenario is indeed in P_k.
- $k^* = t$. Tour τ is completely traversed and we return to r. In this case, the realized scenario does not lie in any of $\{D_{v_k} \mid 1 \leq k \leq t-1\}$, and it is inferred to be in the complement set $M \setminus (\cup_{j<t} D_{v_j})$, which is P_t by definition.

Hence for k^* as defined in Step 4 of $\mathsf{IsoAlg}\langle M, \{q_i\}_{i \in M} \rangle$, it follows that P_{k^*} contains the realized scenario; this proves the first part of the claim (and correctness of the algorithm).

For each $i \in M$, let α_i denote the arrival time of group X_i in tour τ; recall that this is the length of the shortest prefix of τ until it visits an X_i-vertex, and is set to the entire tour length if τ does not cover X_i. The construction of partition $\{P_k\}_{k=1}^t$ from τ implies that

$$\alpha_i \;=\; \textstyle\sum_{j=1}^{k} d(v_{j-1}, v_j); \qquad \forall i \in P_k,\ \forall 1 \le k \le t,$$

and hence $\mathsf{latency}(\tau) = \sum_{i \in M} q_i \cdot \alpha_i$.

To bound the expected distance traversed, note the probability that the traversal returns to r from vertex v_k (for $1 \le k \le t-1$) is exactly $\sum_{i \in P_k} q_i$; with the remaining $\sum_{i \in P_t} q_i$ probability the entire tour τ is traversed. Now, using symmetry and triangle-inequality of the distance function d, we have $d(v_k, r) \le \sum_{j=1}^{k} d(v_{j-1}, v_j)$ for all $k \in [t]$. Hence the expected length traversed is at most:

$$\sum_{k=1}^{t} \left(\sum_{i \in P_k} q_i \right) \cdot \left(d(v_k, r) + \sum_{j=1}^{k} d(v_{j-1}, v_j) \right) \;\le\; 2 \cdot \sum_{k=1}^{t} \left(\sum_{i \in P_k} q_i \right) \cdot \left(\sum_{j=1}^{k} d(v_{j-1}, v_j) \right) \;=\; 2 \cdot \sum_{i \in M} q_i \cdot \alpha_i,$$

which is exactly $2 \cdot \mathsf{latency}(\tau)$. Finally, by Claim 5, this is at most $2 \cdot \rho \cdot \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i \in M} \rangle)$. □

Now, the following simple claim captures the "sub-additivity" of $\mathsf{IsoTime}^*$.

Claim 4. *For any instance $\langle M, \{q_i\}_{i \in M} \rangle$ and any partition $\{P_k\}_{k=1}^t$ of M,*

$$\textstyle\sum_{k=1}^{t} q'_k \cdot \mathsf{IsoTime}^*(\langle P_k, \{\frac{q_i}{q'_k}\}_{i \in P_k} \rangle) \quad \le \quad \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i \in M} \rangle), \tag{2}$$

where $q'_k = \sum_{i \in P_k} q_i$ for all $1 \le k \le t$.

Proof: Let T denote the optimal decision tree for the instance $\mathcal{J}_0 := \langle M, \{q_i\}_{i \in M} \rangle$. For each $k \in [t]$, consider instance $\mathcal{J}_k := \langle P_k, \{\frac{q_i}{q'_k}\}_{i \in P_k} \rangle$; a feasible decision tree for instance $\mathcal{J}_k$ is obtained by taking the decision tree T and considering only paths to the leaf-nodes labeled by $\{i \in P_k\}$. Note that this is a feasible solution since T isolates all scenarios $\cup_{k=1}^t P_k$. Moreover, the expected cost of such a decision tree for $\mathcal{J}_k$ is $\sum_{i \in P_k} \frac{q_i}{q'_k} \cdot d(T_{S_i})$; recall that T_{S_i} denotes the tour traced by T under scenario $i \in P_k$. Hence $\mathsf{Opt}(\mathcal{J}_k) \le \sum_{i \in P_k} \frac{q_i}{q'_k} \cdot d(T_{S_i})$. Summing over all parts $k \in [t]$, we get

$$\sum_{k=1}^{t} q'_k \cdot \mathsf{Opt}(\mathcal{J}_k) \quad \le \quad \sum_{k=1}^{t} q'_k \cdot \sum_{i \in P_k} \frac{q_i}{q'_k} \cdot d(T_{S_i}) \quad = \quad \sum_{i \in M} q_i \cdot d(T_{S_i}) \quad = \quad \mathsf{Opt}(\mathcal{J}_0), \tag{3}$$

where the penultimate equality uses the fact that $\{P_k\}_{k=1}^t$ is a partition of M. □

Given the above claims, we can bound the overall expected cost of the algorithm.

Claim 5. *The expected length of the decision tree given by $\mathsf{IsoAlg}\langle M, \{q_i\}_{i \in M} \rangle$ is at most:*

$$2\rho \cdot \log_{8/7} |M| \cdot \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i \in M} \rangle).$$

Proof: We prove this by induction on $|M|$. The base case of $|M| = 1$ is trivial, since zero length is traversed. Now consider $|M| \ge 2$. Let instance $\mathcal{I}_0 := \langle M, \{q_i\}_{i \in M} \rangle$. For each $k \in [t]$, consider the instance $\mathcal{I}_k := \langle P_k, \{\frac{q_i}{q'_k}\}_{i \in P_k} \rangle$, where $q'_k = \sum_{i \in P_k} q_i$. Note that $|P_k| \le \frac{7}{8}|M| < |M|$ for all $k \in [t]$ by Claim 5 (as $|M| \ge 2$). By the inductive hypothesis, for any $k \in [t]$, the expected length of $\mathsf{IsoAlg}(\mathcal{I}_k)$ is at most $2\rho \cdot \log_{8/7} |P_k| \cdot \mathsf{IsoTime}^*(\mathcal{I}_k) \le 2\rho \cdot (\log_{8/7} |M| - 1) \cdot \mathsf{IsoTime}^*(\mathcal{I}_k)$, since $|P_k| \le \frac{7}{8}|M|$.

By Claim 3, the expected length traversed in Step 4 of $\mathsf{IsoAlg}(\mathcal{I}_0)$ is at most $2\rho \cdot \mathsf{IsoTime}^*(\mathcal{I}_0)$. The probability of recursing on $\mathcal{I}_k$ is exactly q'_k for each $k \in [t]$. So,

$$\begin{aligned}
\text{expected length of } \mathsf{IsoAlg}(\mathcal{I}_0) &\le 2\rho \cdot \mathsf{IsoTime}^*(\mathcal{I}_0) + \textstyle\sum_{k=1}^{t} q'_k \cdot (\text{expected length of } \mathsf{IsoAlg}(\mathcal{I}_k)) \\
&\le 2\rho \cdot \mathsf{IsoTime}^*(\mathcal{I}_0) + \textstyle\sum_{k=1}^{t} q'_k \cdot 2\rho \cdot (\log_{8/7} |M| - 1) \cdot \mathsf{IsoTime}^*(\mathcal{I}_k) \\
&\le 2\rho \cdot \mathsf{IsoTime}^*(\mathcal{I}_0) + 2\rho \cdot (\log_{8/7} |M| - 1) \cdot \mathsf{IsoTime}^*(\mathcal{I}_0) \\
&= 2\rho \cdot \log_{8/7} |M| \cdot \mathsf{IsoTime}^*(\mathcal{I}_0)
\end{aligned}$$

where the third inequality uses Claim 4. □

Claim 5 implies that our algorithm achieves an $O(\rho \log m)$-approximation for $\mathsf{IsoProb}$. This completes the proof of Theorem 5.

3.2. Algorithm for LPGS using GSO Recall the definitions of LPGS and GSO from Section 2. Here we will prove:

THEOREM 6. *If there is a $(4,\gamma)$-bicriteria approximation algorithm for* GSO *then there is an $(O(\gamma),4)$-bicriteria approximation algorithm for* LPGS.

We now describe the algorithm for LPGS in Theorem 6. Consider any instance of LPGS with metric (V,d), root $r \in V$, g groups of vertices $\{X_i \subseteq V\}_{i=1}^{g}$ having weights $\{w_i\}_{i=1}^{g}$, and target $h \leq g$. Let ζ^* be an optimal tour for the given instance of LPGS: let Lat^* denote the latency and D^* the length of ζ^*. We assume (without loss of generality) that the minimum non-zero distance in the metric is one. Let parameter $a := \frac{5}{4}$. Algorithm 3 is the approximation algorithm for LPGS. The "guess" in the first step means the following. We run the algorithm for all choices of l and return the solution having minimum latency *amongst* those that cover at least $h/4$ groups. Since $1 < D^* \leq n \cdot \max_e d_e$, the number of choices for l is at most $\log(n \cdot \max_e d_e)$, and so the algorithm runs in polynomial time.

Algorithm 3 Algorithm for LPGS

1: **guess** an integer l such that $a^{l-1} < D^* \leq a^l$.
2: **mark** all groups as *uncovered*.
3: **for** $i = 1 \ldots l$ **do**
4: **run** the (β,γ)-bicriteria approximation algorithm for GSO on the instance with groups $\{X_i\}_{i=1}^{g}$, root r, length bound a^{i+1}, and profits:

$$\phi_i := \begin{cases} 0 & \text{for each covered group } i \in [g] \\ w_i & \text{for each uncovered group } i \in [g] \end{cases}$$

5: **let** $\tau^{(i)}$ denote the r-tour obtained above.
6: **mark** all groups visited by $\tau^{(i)}$ as *covered*.
7: **end for**
8: **construct** tour $\tau \leftarrow \tau^{(1)} \circ \tau^{(2)} \circ \cdots \circ \tau^{(l)}$, the concatenation of all the above r-tours.
9: *Extend τ if necessary to ensure that $d(\tau) \geq \gamma \cdot a^l$ (this is only needed for the analysis).*
10: **run** the (β,γ)-bicriteria approximation algorithm for GSO on the instance with groups $\{X_i\}_{i=1}^{g}$, root r, length bound a^l, and *unit profit* for each group, i.e. $\phi_i = 1$ for all $i \in [g]$.
11: **let** σ denote the r-tour obtained above.
12: **output** tour $\pi := \tau \circ \sigma$ as solution to the LPGS instance.

Analysis In order to prove Theorem 6, we will show that the algorithm's tour covers at least $\frac{h}{4}$ groups and has latency $O(\gamma) \cdot \mathsf{Lat}^*$.

CLAIM 6. *The tour τ in Step 9 has length $\Theta(\gamma) \cdot D^*$ and latency $O(\gamma) \cdot \mathsf{Lat}^*$.*

Proof: Due to the (β,γ)-bicriteria approximation guarantee of the GSO algorithm used in Step 4, the length of each r-tour $\tau^{(i)}$ is at most $\gamma \cdot a^{i+1}$. So the length of τ in Step 8 is at most $\gamma \sum_{i=1}^{l} a^{i+1} \leq \frac{\gamma}{a-1} a^{l+2} \leq \frac{\gamma a^3}{a-1} D^*$. Moreover, the increase in Step 9 ensures that $d(\tau) \geq \gamma \cdot D^*$. Thus the length of τ in Step 8 is $\Theta(\gamma) \cdot D^*$, which proves the first part of the claim.

The following proof for bounding the latency is based on techniques from the *minimum latency TSP* [7, 12]. Recall the optimal solution ζ^* to the LPGS instance, where $d(\zeta^*) = D^* \in (a^{l-1}, a^l]$. For each $i \in [l]$, let N_i^* denote the total weight of groups visited in ζ^* by time a^i; note that N_l^* equals the total weight of the groups covered by ζ^*. Similarly, for each $i \in [l]$, let N_i denote the

total weight of groups visited in $\tau^{(1)} \cdots \tau^{(i)}$, i.e. by iteration i of the algorithm. Set $N_0 = N_0^* := 0$, and $W := \sum_{i=1}^{g} w_i$ the total weight of all groups. We have:

$$\begin{aligned}\text{latency}(\tau) &\le \sum_{i=1}^{l}(N_i - N_{i-1}) \cdot \sum_{j=1}^{i} \gamma a^{j+1} \;+\; (W - N_l)\cdot d(\tau) \le \sum_{i=1}^{l}(N_i - N_{i-1}) \cdot \frac{\gamma a^{i+2}}{a-1} \;+\; (W - N_l)\cdot d(\tau) \\ &= \sum_{i=1}^{l}((W - N_{i-1}) - (W - N_i)) \cdot \frac{\gamma a^{i+2}}{a-1} \;+\; (W - N_l)\cdot d(\tau) \le \sum_{i=0}^{l}(W - N_i) \cdot \frac{\gamma a^{i+3}}{a-1} \quad =: \quad T.\end{aligned}$$

The last inequality uses the bound $d(\tau) \le \frac{\gamma}{a-1} a^{l+2}$ from above.

The latency of the optimal tour ζ^* is

$$\begin{aligned}\mathsf{Lat}^* &\ge \sum_{i=1}^{l-1} a^{i-1}(N_i^* - N_{i-1}^*) \;+\; (W - N_l^*)\cdot D^* \\ &\ge \sum_{i=1}^{l-1} a^{i-1}\left((W - N_{i-1}^*) - (W - N_i^*)\right) \;+\; (W - N_l^*)\cdot a^{l-1} \quad \ge \quad (1 - \frac{1}{a})\sum_{i=0}^{l} a^i (W - N_i^*).\end{aligned}$$

Consider any iteration $i \in [l]$ of the algorithm in Step 4. Note that the optimal value of the GSO instance solved in this iteration is at least $N_i^* - N_{i-1}$: the a^i length prefix of tour ζ^* corresponds to a feasible solution to this GSO instance with profit at least $N_i^* - N_{i-1}$. The GSO algorithm implies that the profit obtained in $\tau^{(i)}$, i.e. $N_i - N_{i-1} \ge \frac{1}{4}\cdot(N_i^* - N_{i-1})$, i.e. $W - N_i \le \frac{3}{4}\cdot(W - N_{i-1}) + \frac{1}{4}\cdot(W - N_i^*)$. Using this,

$$\begin{aligned}(a-1)\frac{T}{\gamma} = \sum_{i=0}^{l} a^{i+3}\cdot(W - N_i) \quad &\le \quad a^3 \cdot W + \frac{1}{4}\sum_{i=1}^{l} a^{i+3}(W - N_i^*) + \frac{3}{4}\sum_{i=1}^{l} a^{i+3}(W - N_{i-1}) \\ &\le \frac{a^4}{a-1}\cdot\mathsf{Lat}^* + \frac{3}{4}\sum_{i=1}^{l} a^{i+3}(W - N_{i-1}) \quad = \quad \frac{a^4}{a-1}\cdot\mathsf{Lat}^* + \frac{3a}{4}\sum_{i=0}^{l-1} a^{i+3}(W - N_i) \\ &\le \frac{a^4}{a-1}\cdot\mathsf{Lat}^* + \frac{3a}{4}\cdot(a-1)\frac{T}{\gamma}\end{aligned}$$

This implies $T \le \gamma \cdot \frac{a^4}{(a-1)^2(1-3a/4)} \cdot \mathsf{Lat}^* = O(\gamma)\cdot\mathsf{Lat}^*$ since $a = \frac{5}{4}$. This completes the proof. □

Claim 7. *The tour σ in Step 10 covers at least $\frac{h}{4}$ groups and has length $O(\gamma)\cdot D^*$.*

Proof: Since we know that the optimal tour ζ^* has length at most a^l and covers at least h groups, it is a feasible solution to the GSO instance defined in Step 10. So the GSO algorithm ensures that the tour σ has length at most $\gamma a^l = O(\gamma)D^*$ and profit (i.e. number of groups) at least $h/4$. □

Lemma 1. *Tour $\pi = \tau\cdot\sigma$ covers at least $\frac{h}{4}$ groups and has latency $O(\gamma)\cdot\mathsf{Lat}^*$.*

Proof: Since π visits all the vertices in σ, Claim 7 implies that π covers at least $\frac{h}{4}$ groups. For each group $i \in [g]$, let α_i denote its *arrival time* under the tour τ after Step 9—recall that the arrival time α_i for any group i that is not covered by τ is set to the length of the tour $d(\tau)$. Claim 6 implies that the latency of tour τ, $\sum_{i=1}^{g} w_i\cdot\alpha_i = O(\gamma)\cdot\mathsf{Lat}^*$. Observe that for each group i that is *covered* in τ, its arrival time under tour $\pi = \tau\cdot\sigma$ remains α_i. For any group j *not covered* in τ, its arrival time under τ is $d(\tau) \ge \gamma\cdot a^l$ (due to Step 9), and its arrival time under π is $d(\pi) \le O(\gamma)\cdot D^* = O(1)\cdot d(\tau)$. Hence, the arrival time under π of each group $i \in [g]$ is $O(1)\cdot\alpha_i$, i.e., at most a constant factor more than its arrival time in τ. Now using Claim 6 completes the proof. □

Finally, Lemma 1 directly implies Theorem 6.

Remark: The above approach also leads to an approximation algorithm for the *minimum latency group Steiner* problem, which is the special case of LPGS when the target $h = g$.

Definition 8 (Minimum Latency Group Steiner). The input is a metric (V,d), g groups of vertices $\{X_i \subseteq V\}_{i=1}^{g}$ with associated non-negative weights $\{w_i\}_{i=1}^{g}$ and root $r \in V$. The goal in LGS is to compute an r-tour that covers all groups with positive weight and minimizes the weighted sum of arrival times of the groups. The *arrival time* of group $i \in [g]$ is the length of the shortest prefix of the tour that contains a vertex from X_i.

Note that the objective here is to minimize the sum of weighted arrival times where every group has to be visited. The algorithm for latency group Steiner is in fact simpler than Algorithm 3: we do not need the "guess" l (Step 1) and we just repeat Step 4 until *all* groups are covered (instead of stopping after l iterations). A proof identical to that in Claim 6 gives:

Corollary 1. *If there is a $(4,\gamma)$-bicriteria approximation algorithm for* GSO *then there is an $O(\gamma)$-approximation algorithm for the latency group Steiner problem.*

Combined with the $(4, O(\log^2 n)$-bicriteria approximation algorithm for GSO (see Section 5.1) we obtain an $O(\log^2 n)$-approximation algorithm for LGS. It is shown in [32] that any α-approximation algorithm for LGS can be used to obtain an $O(\alpha \cdot \log g)$-approximation algorithm for group Steiner tree. Thus improving this $O(\log^2 n)$-approximation algorithm for latency group Steiner would also improve the best known bound for the standard group Steiner tree problem.

4. Optimal Decision Tree Problem

Recall that the *optimal decision tree problem* consists of a set of diseases with their probabilities (where exactly one disease occurs) and a set of binary tests with costs, and the goal is to identify the realized disease at minimum expected cost. In this section we prove Theorem 1.

As noted in Section 2 the optimal decision tree problem (Definition 5) is a special case of IsoProb (Definition 4). We recall the reduction for convenience. Given an instance of ODT, consider a metric (V,d) induced by a weighted star with center r and n leaves corresponding to the tests. For each $j \in [n]$, we set $d(r,j) = \frac{c_j}{2}$. The demand scenarios are as follows: for each $i \in [m]$ scenario i has demands $S_i = \{j \in [n] \mid i \in T_j\}$. It is easy to see that this IsoProb instance corresponds exactly to the optimal decision tree instance. Figure 2 gives an example.

The main observation here is the following:

Theorem 7. *There is a $(1-\frac{1}{e})$-approximation algorithm for the group Steiner orienteering problem on weighted star metrics.*

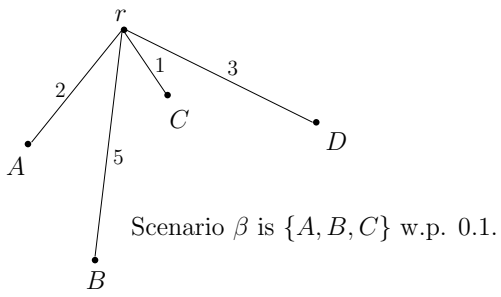
Proof: Consider an instance of GSO (Definition 6) on weighted star-metric (V,d) with center r (which is also the root in GSO) and leaves $[n]$, g groups $\{X_i \subseteq [n]\}_{i=1}^{g}$ with profits $\{\phi_i\}_{i=1}^{g}$, and length bound B. If for each $j \in [n]$, we define set $Y_j := \{i \in [g] \mid j \in X_i\}$ of cost $c_j := \frac{d(r,j)}{2}$, then solving the GSO instance is the same as computing a collection $K \subseteq [n]$ of the sets with $\sum_{j\in K} c_j \leq B/2$ that maximizes $f(K) := \sum\{\phi_i \mid i \in \cup_{j\in K} Y_j\}$. But the latter problem is precisely an instance of maximizing a monotone submodular function over a knapsack constraint ($\sum_{j\in K} c_j \leq B/2$), for which a $(1-\frac{1}{e})$-approximation algorithm is known [37]. □

Combining this result with Theorem 4, we obtain an $O(\log m)$ approximation algorithm for IsoProb on weighted star-metrics and hence ODT. This proves the first part of Theorem 1.

Multiway tests. Our algorithm can be easily extended to the generalization of ODT where tests have multiway (instead of binary) outcomes. In this setting (when each test has at most l outcomes), any test $j \in [n]$ induces a partition $\{T_j^k\}_{k=1}^{l}$ of $[m]$ into l parts (some of them may be empty), and performing test j determines which part the realized disease lies in. Note that this problem is also a special case of IsoProb. As before, consider a metric (V,d) induced by a weighted star with center r and n leaves corresponding to the tests. For each $j \in [n]$, we set $d(r,j) = \frac{c_j}{2}$. Additionally, for each $j \in [n]$, introduce l copies of test-vertex j, labeled $(j,1), \cdots, (j,l)$, at zero

Binary tests $\{A, B, C, D\}$ and diseases $\{\alpha, \beta, \gamma, \delta\}$.

		cost : 4	10	2	6
prob.		A	B	C	D
0.1	α	+	+	−	−
0.1	β	+	+	+	−
0.2	γ	+	−	+	+
0.6	δ	−	−	−	+



Multiway tests $\{A, B, C\}$ with $l = 3$ outcomes. Diseases $\{\alpha, \beta, \gamma, \delta\}$.

		cost : 4	10	2
prob.		A	B	C
0.1	α	0	1	0
0.1	β	1	2	0
0.2	γ	1	2	2
0.6	δ	2	2	1

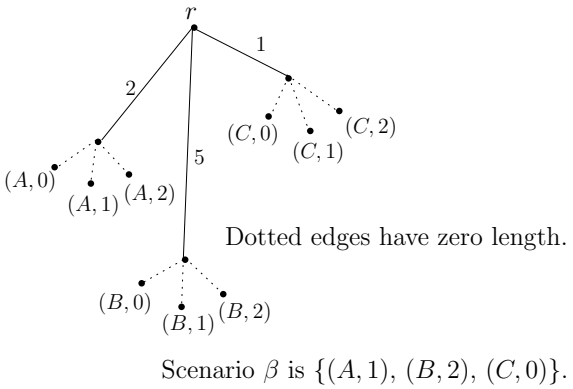


FIGURE 2. Reducing optimal decision tree to Isolation: binary tests (top), multiway tests (bottom).

distance from each other. The demand scenarios are defined naturally: for each $i \in [m]$, scenario i has demands $S_i = \{(j,k) \mid i \in T_j^k\}$. See also an example in Figure 2. Clearly this IsoProb instance is equivalent to the (multiway) decision tree instance. Since the resulting metric is still a weighted star (we only made vertex copies), Theorem 7 along with Theorem 4 implies an $O(\log m)$-approximation for the multiway decision tree problem. This proves the second part of Theorem 1.

5. Adaptive Traveling Salesman Problem Recall that the adaptive TSP (Definition 3) consists of a metric (V, d) with root $r \in V$ and demand distribution $\mathcal{D}$, and the goal is to visit all demand vertices (drawn from $\mathcal{D}$) using an r-tour of minimum expected cost. We first show the following simple fact relating this problem to the isolation problem.

LEMMA 2. *If there is an α-approximation algorithm for* IsoProb *then there is an $\left(\alpha + \frac{3}{2}\right)$-approximation algorithm for* AdapTSP.

Proof: We first claim that any feasible solution T to AdapTSP is also feasible for IsoProb. For this it suffices to show that the paths $T_{S_i} \neq T_{S_j}$ for any two scenarios $i, j \in [m]$ with $i \neq j$. Suppose (for a contradiction) that paths $T_{S_i} = T_{S_j} = \pi$ for some $i \neq j$. By feasibility of T for AdapTSP, path π contains all vertices in $S_i \bigcup S_j$. Since $S_i \neq S_j$, there is some vertex in $(S_i \setminus S_j) \bigcup (S_j \setminus S_i)$; let $u \in S_i \setminus S_j$ (the other case is identical). Consider the point where π is at a node labeled u: then path T_{S_i} must take the *yes* child, whereas path T_{S_j} must take the *no* child. This contradicts the assumption $T_{S_i} = T_{S_j} = \pi$. Thus any solution to AdapTSP is also feasible for IsoProb; moreover the expected cost remains the same. Hence the optimal value of IsoProb is at most that of AdapTSP.

Now, using any α-approximation algorithm for IsoProb, we obtain a decision tree T' that isolates the realized scenario and has expected cost $\alpha \cdot \mathsf{Opt}$, where Opt denotes the optimal value of the AdapTSP instance. This suggests the following feasible solution for AdapTSP:

1. Implement T' to determine the realized scenario $k \in [m]$, and return to r.
2. Traverse a $\frac{3}{2}$-approximate TSP tour [9] on vertices $\{r\} \bigcup S_k$.

From the preceding argument, the expected length in the first phase is at most $\alpha \cdot \mathsf{Opt}$. The expected length in the second phase is at most $\frac{3}{2} \sum_{i=1}^{m} p_i \cdot \mathsf{Tsp}(S_i)$, where $\mathsf{Tsp}(S_i)$ denotes the minimum length of a TSP tour on $\{r\} \bigcup S_i$. Note that $\sum_{i=1}^{m} p_i \cdot \mathsf{Tsp}(S_i)$ is a lower bound on the optimal AdapTSP value. So we obtain a solution that has expected cost at most $(\alpha + \frac{3}{2})\mathsf{Opt}$, as claimed. □

Therefore, it suffices to obtain an approximation algorithm for IsoProb. In the next subsection we obtain a $(4, O(\log^2 n))$-bicriteria approximation algorithm for GSO, which combined with Theorem 4 and Lemma 2 yields an $O(\log^2 n \cdot \log m)$-approximation algorithm for both IsoProb and AdapTSP. This would prove Theorem 2.

5.1. Algorithm for Group Steiner Orienteering Recall the GSO problem (Definition 6). Here we obtain a bicriteria approximation algorithm for GSO.

THEOREM 8. *There is a $(4, O(\log^2 n))$-bicriteria approximation algorithm for* GSO*, where n is the number of vertices in the metric. That is, the algorithm's tour has length $O(\log^2 n) \cdot B$ and has profit at least $\frac{1}{4}$ times the optimal profit of a length B tour.*

This algorithm is based on a greedy framework that is used in many maximum-coverage problems: the solution is constructed iteratively where each iteration adds an r-tour that maximizes the ratio of profit to length. In order to find an r-tour (approximately) maximizing the profit to length ratio, we use a slight modification of an existing algorithm [6]; see Theorem 9 below. The final GSO algorithm is then given as Algorithm 5.

THEOREM 9. *There is a polynomial time algorithm that given any instance of* GSO*, outputs an r-tour σ having profit-to-length ratio $\frac{\phi(\sigma)}{d(\sigma)} \geq \frac{1}{\alpha} \cdot \frac{\mathsf{Opt}}{B}$. Here $\phi(\sigma)$ and $d(\sigma)$ denote the profit and length (respectively) of tour σ,* Opt *is the optimal value of the* GSO *instance, B is the length bound in* GSO *and $\alpha = O(\log^2 n)$ where n is the number of vertices in the metric.*

Proof: This result essentially follows from [6], but requires some modifications which we present here for completeness. We first preprocess the metric to only include vertices within distance $B/2$ from the root r: note that since the optimal GSO tour cannot visit any excluded vertex, the optimal profit remains unchanged by this. To reduce notation, we refer to this restricted vertex-set also as V and let $|V| = n$. We denote the set of all edges in the metric by $E = \binom{V}{2}$. We assume (without loss of generality) that every group is covered by some vertex in V; otherwise the group can be dropped from the GSO instance. By averaging, there is some vertex $u \in V$ covering groups of total profit at least $\frac{1}{n} \sum_{i=1}^{g} \phi_i$. If $\mathsf{Opt} \leq \frac{4}{n} \sum_{i=1}^{g} \phi_i$ then the r-tour that just visits vertex u has profit-to-length ratio at least $\frac{\mathsf{Opt}}{4B}$ and is output as the desired tour σ. Below we assume that $\frac{1}{n} \sum_{i=1}^{g} \phi_i < \frac{\mathsf{Opt}}{4}$.

We use the following linear programming relaxation $\mathsf{LP}_{\mathsf{GSO}}$ for GSO:

$$\begin{array}{lll} \max & \sum_{i=1}^{g} \phi_i \cdot y_i & \\ \text{s.t.} & x(\delta(S)) \geq y_i & \forall S \subseteq V : r \notin S,\ X_i \subseteq S; \quad \forall i \in [g] \\ & \sum_{e \in E} d_e \cdot x_e \leq B & \\ & 0 \leq y_i \leq 1 & \forall i \in [g] \\ & x_e \geq 0 & \forall e \in E \end{array} \tag{4}$$

It is easy to see that this a valid relaxation of GSO: any feasible GSO solution corresponds to a feasible solution above where the x, y variables are $\{0, 1\}$ valued. So the optimal value $\sum_{i=1}^{g} \phi_i \cdot y_i \geq$

Opt. The algorithm is given as Algorithm 4 and uses the following known results: Theorem 10 shows how to round fractional solutions to $\mathsf{LP}_{\mathsf{GSO}}$ on tree metrics and Theorem 11 shows how to transform an $\mathsf{LP}_{\mathsf{GSO}}$ solution on general metrics to one on a tree.

THEOREM 10 ([6]). *There is a polynomial time algorithm that given any fractional solution (x,y) to $\mathsf{LP}_{\mathsf{GSO}}$ on a tree metric where all variables are integral multiples of $\frac{1}{N}$, finds a subtree A containing r such that $\frac{d(A)}{\phi(A)} \le O(\log N) \cdot \frac{\sum_{e\in E} d_e \cdot x_e}{\sum_{i=1}^{g} \phi_i \cdot y_i}$. Here $\phi(A)$ and $d(A)$ denote the profit and length (respectively) of subtree A.*

THEOREM 11 ([13]). *There is a polynomial time algorithm that given any metric (V,d) with edges $E = \binom{V}{2}$ and capacity function $x : E \to \mathbb{R}_+$, computes a spanning tree T in this metric such that $\sum_{f\in T} d_f \cdot x_T(f) \le O(\log n) \cdot \sum_{e\in E} d_e \cdot x(e)$, where*

$$x_T(f) \quad := \quad \sum_{u,v\,:\, f\in uv\,path\,in\,T} x(u,v), \qquad \forall f \in T.$$

Algorithm 4 Algorithm for GSO maximizing profit-to-length ratio.

1: **solve** the linear program $\mathsf{LP}_{\mathsf{GSO}}$ to obtain solution (x,y).
2: **run** the algorithm from Theorem 11 on metric (V,d) with edge-capacities x to obtain a spanning tree T with "new capacities" x_T on edges of T.
3: **round** down each $x_T(e)$ to an integral multiple of $\frac{1}{n^3}$.
4: **for** each group $i \in [g]$, let y'_i be the maximum flow from r to group X_i under capacities x_T.
5: **run** the algorithm from Theorem 10 using variables x_T and y' to obtain subtree A.
6: **output** an Euler tour σ of the subtree A.

By definition of the new edge-capacities x_T on edges of T (see Theorem 11) it is clear that the capacity of each cut under x_T is at least as much as under x; i.e. $\sum_{e\in\delta(S)} x_T(e) \ge \sum_{e\in\delta(S)} x(e)$ for all $S \subseteq V$. For each group $i \in [g]$, since capacities x support y_i units of flow from r to X_i, it follows that the new capacities x_T on tree T also support such a flow. So (x_T, y) is a feasible solution to $\mathsf{LP}_{\mathsf{GSO}}$ on tree T with budget $O(\log n) \cdot B$. In order to apply the rounding algorithm from [6] for GSO on trees, we need to ensure the technical condition (see Theorem 10) that every variable is an integral multiple of $\frac{1}{N}$ for some $N = poly(n)$. This is the reason behind modifying capacities x_T in Step 3. Note that this step reduces the capacity $x_T(e)$ of each edge $e \in T$ by at most $\frac{1}{n^3}$. Since any cut in tree T has at most n edges, the capacity of any cut decreases by at most $\frac{1}{n^2}$ after Step 3; and by the max-flow min-cut theorem, the maximum flow value for group X_i is $y'_i \ge y_i - \frac{1}{n^2}$ for each $i \in [g]$ (in Step 4). Furthermore, since all edge capacities are integer multiples of $\frac{1}{n^3}$, so are all the flow values y'_is. So (x_T, y') is a feasible solution to $\mathsf{LP}_{\mathsf{GSO}}$ on tree T (with budget $O(\log n) \cdot B$) that satisfies the condition required in Theorem 10, with $N = n^3$. Also note that this rounding down does not change the fractional profits much, since

$$\sum_{i=1}^{g} \phi_i \cdot y'_i \;\ge\; \sum_{i=1}^{g} \phi_i \cdot y_i - \frac{1}{n^2}\sum_{i=1}^{g} \phi_i \;\ge\; \frac{3}{4}\cdot \mathsf{Opt} - \frac{1}{n^2}\sum_{i=1}^{g} \phi_i \;\ge\; \frac{3}{4}\cdot \mathsf{Opt} - \frac{\mathsf{Opt}}{4n} \;\ge\; \frac{\mathsf{Opt}}{2} \tag{5}$$

where the second last inequality follows from $\frac{1}{n}\sum_{i=1}^{g} \phi_i \le \frac{\mathsf{Opt}}{4}$ (by the preprocessing). Now applying Theorem 10 implies that subtree A satisfies:

$$\begin{aligned}\frac{d(A)}{\phi(A)} &\le_{(Theorem\ 10)} O(\log N) \cdot \frac{\sum_{e\in T} d_e \cdot x_T(e)}{\sum_{i=1}^{g} \phi_i \cdot y'_i} \le_{(5)} O(\log N) \cdot \frac{\sum_{e\in T} d_e \cdot x_T(e)}{\mathsf{Opt}} \\ &\le_{(Theorem\ 11)} O(\log N \log n) \cdot \frac{\sum_{e\in E} d_e \cdot x(e)}{\mathsf{Opt}} \le O(\log^2 n) \cdot \frac{B}{\mathsf{Opt}}.\end{aligned}$$

Finally, since we output an Euler tour of A, the theorem follows. □

Remark: A simpler approach in Theorem 9 might have been to use the randomized algorithm from [16] rather than the deterministic algorithm (Theorem 10) from [6]. This however does not work directly since [16] only yields a random solution A' with expected length $\mathbf{E}[d(A')] \leq O(\log n) \cdot \sum_{e \in E} d_e \cdot x_e$ and expected profit $\mathbf{E}[\phi(A')] \geq \sum_{i=1}^{g} \phi_i \cdot y_i$. While this does guarantee the existence of a solution with length-to-profit ratio at most $O(\log n) \cdot \frac{\sum_{e \in E} d_e \cdot x_e}{\sum_{i=1}^{g} \phi_i \cdot y_i}$, it may not find such a solution with reasonable (inverse polynomial) probability.

Algorithm The GSO algorithm first preprocesses the metric to only include vertices within distance $B/2$ from the root r: note that the optimal profit remains unchanged by this. The algorithm then follows a standard greedy approach (see eg. Garg [17]), and is given as Algorithm 5.

Algorithm 5 Algorithm for GSO.

1: **initialize** r-tour $\tau \leftarrow \emptyset$ and mark all groups as uncovered.
2: **while** length of τ does not exceed $\alpha \cdot B$ **do**
3: **set** residual profits:

$$\widetilde{\phi}_i := \begin{cases} 0 & \text{for each covered group } i \in [g] \\ \phi_i & \text{for each uncovered group } i \in [g] \end{cases}$$

4: **run** the algorithm from Theorem 9 on the GSO instance with profits $\widetilde{\phi}$ to obtain r-tour σ.
5: **if** $d(\sigma) \leq \alpha B$ then $\tau' \leftarrow \tau \circ \sigma$.
6: **if** $d(\sigma) > \alpha B$ then:
(i) partition tour σ into at most $2 \cdot \frac{d(\sigma)}{\alpha B}$ paths, each of length at most αB;
(ii) let σ' denote the path containing maximum profit;
(iii) let $\langle r, \sigma', r \rangle$ be the r-tour obtained by connecting both end-vertices of path σ' to r.
(iv) set $\tau' \leftarrow \tau \cup \langle r, \sigma', r \rangle$.
7: **set** $\tau \leftarrow \tau'$. Mark all groups visited in τ as covered.
8: **end while**
9: **output** the r-tour τ.

Analysis Let Opt denote the optimal profit of the given GSO instance. In the following, let $\alpha := O(\log^2 n)$ which comes from Theorem 9. We prove that Algorithm 5 achieves a $(4, 2\alpha + 1)$ bicriteria approximation guarantee, i.e. solution τ has profit at least $\mathsf{Opt}/4$ and length $(2\alpha+1) \cdot B$.

By the description of the algorithm, we iterate as long as the total length of edges in τ is at most αB. Note that the increase in length of τ in any iteration is at most $(\alpha + 1) \cdot B$ since every vertex is at distance at most $B/2$ from r. So the final length $d(\tau) \leq (2\alpha + 1) \cdot B$. This proves the bound on the length.

It now suffices to show that the final subgraph τ gets profit at least $\frac{\mathsf{Opt}}{4}$. At any iteration, let $\phi(\tau)$ denote the profit of the current solution τ, and $d(\tau)$ its length. Since $d(\tau) > \alpha B$ upon termination, it suffices to show the following invariant over the iterations of the algorithm:

$$\phi(\tau) \quad \geq \quad \min\left\{ \frac{\mathsf{Opt}}{4}, \frac{\mathsf{Opt}}{2\alpha B} \cdot d(\tau) \right\} \tag{6}$$

At the start of the algorithm, inequality (6) holds trivially since $d(\tau) = 0$ for $\tau = \emptyset$. Consider any iteration where $\phi(\tau) < \mathsf{Opt}/4$ at the beginning: otherwise (6) trivially holds for the next iteration. The invariant now ensures that $d(\tau) < \alpha B/2$ and hence we proceed further with the iteration. Moreover, in Step 4 the optimal value of the "residual" GSO instance with profits $\widetilde{\phi}$ is

$\widetilde{\mathsf{Opt}} \geq \mathsf{Opt} - \phi(\tau) \geq \frac{3}{4} \cdot \mathsf{Opt}$ (by considering the optimal tour for the GSO instance with profits ϕ). By Theorem 9, the r-tour σ satisfies $d(\sigma)/\widetilde{\phi}(\sigma) \leq \alpha \cdot B/\widetilde{\mathsf{Opt}} \leq 2\alpha \cdot B/\mathsf{Opt}$.

We finish by handling the two possible cases (Steps 5 and 6).

- If $d(\sigma) \leq \alpha B$, then $\phi(\tau') = \phi(\tau) + \widetilde{\phi}(\sigma) \geq \frac{\mathsf{Opt}}{2\alpha B} \cdot d(\tau) + \frac{\mathsf{Opt}}{2\alpha B} \cdot d(\sigma) = \frac{\mathsf{Opt}}{2\alpha B} \cdot d(\tau')$.
- If $d(\sigma) > \alpha B$, then σ is partitioned into at most $\frac{2d(\sigma)}{\alpha B}$ paths of length αB each. The path σ' of best profit has $\widetilde{\phi}(\sigma') \geq \frac{\alpha B}{2 \cdot d(\sigma)} \widetilde{\phi}(\sigma) \geq \frac{\mathsf{Opt}}{4}$; so $\phi(\tau') \geq \widetilde{\phi}(\sigma') \geq \frac{\mathsf{Opt}}{4}$.

In either case r-tour τ' satisfies inequality (6), and since $\tau \leftarrow \tau'$ at the end of the iteration, the invariant holds for next iteration as well. This completes the proof of Theorem 8.

6. Adaptive Traveling Repairman In this section we consider the adaptive traveling repairman problem (AdapTRP), where given a demand distribution, the goal is to find an adaptive strategy that minimizes the expected *sum of arrival times* at demand vertices. As in adaptive TSP, we assume that the demand distribution $\mathcal{D}$ is specified explicitly in terms of its support.

Definition 9 (Adaptive Traveling Repairman). The input is a metric (V, d), root r and demand distribution $\mathcal{D}$ given by m distinct subsets $\{S_i\}_{i=1}^m$ with probabilities $\{p_i\}_{i=1}^m$ (which sum to one). The goal in AdapTRP is to compute a decision tree T in metric (V, d) such that:

- the root of T is labeled with the root vertex r, and
- for each scenario $i \in [m]$, the path T_{S_i} followed on input S_i contains all vertices in S_i.

The objective function is to minimize the expected latency $\sum_{i=1}^m p_i \cdot \mathsf{Lat}(T_{S_i})$, where $\mathsf{Lat}(T_{S_i})$ is the sum of arrival times at vertices S_i along path T_{S_i}.

We obtain an $O(\log^2 n \log m)$-approximation algorithm for AdapTRP (Theorem 3). The high-level approach here is similar to that for AdapTSP, but there are some important differences. Unlike AdapTSP, we can not directly reduce AdapTRP to the isolation problem: so there is no analogue of Lemma 2 here. The following example illustrates this.

Example 1. Consider an instance of AdapTRP on a star-metric with center r and leaves $\{v, u_1, \cdots, u_n\}$. Edges (r, u_i) have unit length for each $i \in [n]$, and edge (r, v) has length $\sqrt{n}$. There are $m = n + 1$ scenarios: scenario $S_0 = \{v\}$ occurs with $1 - \frac{1}{n}$ probability; and for each $i \in [n]$, scenario $S_i = \{v, u_i\}$ occurs with $\frac{1}{n^2}$ probability. The optimal IsoProb value for this instance is $\Omega(n)$ and any reasonable solution clearly will not visit vertex v: it appears in all scenarios and hence provides no information. So if we first follow such an IsoProb solution, the arrival time for v is $\Omega(n)$; since $S_0 = \{v\}$ occurs with $1 - o(1)$ probability, the resulting expected latency is $\Omega(n)$. However, the AdapTRP solution that first visits v, and then vertices $\{u_1, \cdots, u_n\}$ has expected latency $O(\sqrt{n})$.

On the other hand, one can not ignore the "isolation aspect" in AdapTRP either.

Example 2. Consider another instance of AdapTRP on a star-metric with center r and leaves $\{v_i\}_{i=1}^n \cup \{u_i\}_{i=1}^n$. For each $i \in [n]$, edge (r, v_i) has unit length and edge (r, u_i) has length n. There are n scenarios: for each $i \in [n]$, scenario $S_i = \{v_i, u_i\}$ occurs with $\frac{1}{n}$ probability. The optimal values for both AdapTRP and IsoProb are $\Theta(n)$. Moreover, any reasonable AdapTRP solution will involve first isolating the realized scenario (by visiting vertices v_is).

Hence, the algorithm needs to interleave the two goals of isolating scenarios and visiting high-probability vertices. This will become clear in the construction of the latency group Steiner instances used by our algorithm (Step 3 in Algorithm 6).

Algorithm Outline Although we can not reduce AdapTRP to IsoProb, we are still able to use ideas from the IsoProb algorithm. The AdapTRP algorithm also follows an iterative approach and maintains a candidate set $M \subseteq [m]$ containing the realized scenario. We also associate conditional probabilities $q_i := \frac{p_i}{\sum_{j \in M} p_j}$ for each scenario $i \in M$. In each iteration, the algorithm eliminates a constant fraction of scenarios from M: so the number of iterations will be $O(\log m)$. Each iteration involves solving an instance of the *latency group Steiner* (LGS) problem: recall Definition 8 and the

$O(\log^2 n)$-approximation algorithm for LGS (Corollary 1). The construction of this LGS instance is the main point of difference from the IsoProb algorithm. Moreover, we will show that the *expected latency* incurred in each iteration is $O(\log^2 n)\cdot \mathsf{Opt}$. Adding up the latency over all iterations, would yield an $O(\log^2 n \log m)$-approximation algorithm for AdapTRP.

Using* LGS *to partition scenarios M In each iteration, the algorithm formulates an LGS instance and computes an r-tour τ using Corollary 1. The details are in Algorithm 6 below. An important property of this tour τ is that the number of candidate scenarios after observing demands on τ will be at most $|M|/2$ (see Claim 8).

Given a candidate set M of scenarios, it will be convenient to partition the vertices into two parts: H consists of vertices which occur in more than half the scenarios, and $L := V \setminus H$ consists of vertices occurring in at most half the scenarios. In the LGS instance (Step 3 below), we introduce $|S_i \cap H| + 1$ groups (with suitable weights) corresponding to each scenario $i \in M$.

Algorithm 6 PartnLat($\langle M, \{q_i\}_{i\in M}, \{S_i\}_{i\in M}\rangle$)

1: **define** $F_v := \{i \in M \mid v \in S_i\}$ for each $v \in V$.
2: **let** $L := \left\{u \in V : |F_u| \le \frac{|M|}{2}\right\}$, $H := V \setminus L$, and $D_v := \begin{cases} F_v & \text{if } v \in L \\ M \setminus F_v & \text{if } v \in H \end{cases}$
3: **define** instance $\mathcal{G}$ of LGS (Definition 8) on metric (V, d), root r and the following groups: for each scenario $i \in M$,
- the *main* group X_i of scenario i has weight $|S_i \cap L| p_i$ and vertices $(L \cap S_i) \cup (H \setminus S_i)$.
- for each $v \in S_i \cap H$, group Y_i^v has weight p_i and vertices $\{v\} \cup (L \cap S_i) \cup (H \setminus S_i)$.

4: **run** the LGS algorithm (from Corollary 1) on instance $\mathcal{G}$.
let $\tau := \langle r, v_1, v_2, \cdots, v_{t-1}, r\rangle$ be the r-tour returned.
5: **let** $\{P_k\}_{k=1}^t$ be the partition of M where $P_k := \begin{cases} D_{v_k} \setminus \left(\cup_{j<k} D_{v_j}\right) & \text{if } 1 \le k \le t-1 \\ M \setminus \left(\cup_{j<t} D_{v_j}\right) & \text{if } k = t \end{cases}$
6: **return** tour $\tau = \langle r, v_1, v_2, \cdots, v_{t-1}, r\rangle$ and partition $\{P_k\}_{k=1}^t$.

Claim 8. *When $|M| \ge 2$, partition $\{P_k\}_{k=1}^t$ returned by* PartnLat *satisfies $|P_k| \le |M|/2$, $\forall k \in [t]$.*

Proof: For each $k \in [t-1]$, we have $P_k \subseteq D_{v_k}$ and so $|P_k| \le |M|/2$. We now show that $|P_t| \le 1$ which would prove the claim. Let $V(\tau) = \{v_1, \ldots, v_{t-1}\}$ denote the vertices visited in the tour τ output by PartnLat. Consider any $i \in P_t$: we will show that it is unique. By definition of P_t, we have $i \notin \bigcup_{k=1}^{t-1} D_{v_k}$. By the definition of group X_i and sets D_vs, this means that X_i is *not covered* by $V(\tau)$. Since τ is a feasible solution to $\mathcal{G}$, X_i's weight must be zero, i.e. $|S_i \cap L| = 0$. Thus we have $S_i \subseteq H$. Furthermore, if $v_k \in H \setminus S_i$ for any $k \in [t-1]$ then $i \in D_{v_k}$, which implies $i \notin P_t$; so $H \cap V(\tau) \subseteq S_i$. Note that each $Y_i^v = \{v\} \cup X_i$ (for $v \in H \cap S_i = S_i$) must be covered by τ, since Y_i^vs have weight $p_i > 0$. Also since X_i is not covered by $V(\tau)$, we must have $v \in V(\tau)$ for all $v \in S_i$. Thus we have $S_i \subseteq H \cap V(\tau)$, and combined with the earlier observation, $H \cap V(\tau) = S_i$. This determines $i \in M$ uniquely, and so $|P_t| = 1 \le |M|/2$. □

Final* AdapTRP *algorithm and analysis Given the above partitioning scheme, Algorithm 7 describes the overall AdapTRP algorithm in a recursive manner.

The analysis for this algorithm is similar to that for the isolation problem (Section 3.1) and we follow the same outline. For any sub-instance $\mathcal{J}$ of AdapTRP, let $\mathsf{Opt}(\mathcal{J})$ denote its optimal value. Just as in the isolation case (Claim 4), it can be easily seen that the latency objective function is also sub-additive.

Claim 9. *For any sub-instance $\langle M, \{q_i\}_{i\in M}, \{S_i\}_{i\in M}\rangle$ and any partition $\{P_k\}_{k=1}^t$ of M,*

$$\sum_{k=1}^t q'_k \cdot \mathsf{Opt}(\langle P_k, \{\tfrac{q_i}{q'_k}\}_{i\in P_k}, \{S_i\}_{i\in P_k}\rangle) \quad \le \quad \mathsf{Opt}(\langle M, \{q_i\}_{i\in M}, \{S_i\}_{i\in M}\rangle), \tag{7}$$

Algorithm 7 $\mathsf{AdapTRP}\langle M, \{q_i\}_{i\in M}, \{S_i\}_{i\in M}\rangle$

1: If $|M| = 1$, visit the vertices in this scenario using the $O(1)$-approximation algorithm [12] for deterministic traveling repairman, and quit.
2: **run** $\mathsf{PartnLat}\langle M, \{q_i\}_{i\in M}\rangle$
let $\tau = (r, v_1, v_2, \cdots, v_{t-1}, r)$ be the r-tour and $\{P_k\}_{k=1}^{t}$ be the partition of M returned.
3: **let** $q'_j := \sum_{i\in P_k} q_i$ **for all** $j = 1 \dots t$.
4: **traverse** tour τ and return directly to r after visiting the first vertex v_{k^*} (for $k^* \in [t]$) that determines that the realized scenario is in $P_{k^*} \subseteq M$.
5: **update** the scenarios in P_{k^*} by removing vertices visited in τ until v_{k^*}, i.e.

$$S'_i \leftarrow S_i \setminus \{v_1, \dots, v_{k^*}\}, \qquad \text{for all } i \in P_{k^*}.$$

6: **run** $\mathsf{AdapTRP}\langle P_{k^*}, \{\frac{q_i}{q'_{k^*}}\}_{i\in P_{k^*}}, \{S'_i\}_{i\in P_{k^*}}\rangle$ to recursively cover the realized scenario within P_{k^*}.

where $q'_k = \sum_{i\in P_k} q_i$ *for all* $1 \le k \le t$.

The next property we show is that the optimal cost of the LGS instance $\mathcal{G}$ considered in Steps (3)-(4) of Algorithm 6 is not too high.

LEMMA 3. *For any instance* $\mathcal{J} = \langle M, \{q_i\}_{i\in M}, \{S_i\}_{i\in M}\rangle$ *of* AdapTRP*, the optimal value of the latency group Steiner instance* $\mathcal{G}$ *in Step 4 of Algorithm* $\mathsf{PartnLat}(\mathcal{J})$ *is at most* $\mathsf{Opt}(\mathcal{J})$.

Proof: Let T be an optimal decision tree for the given AdapTRP instance $\mathcal{J}$. Note that any internal node of T, labeled v, has two children corresponding to the realized scenario being in F_v (*yes* child) or $M \setminus F_v$ (*no* child). Now consider the root-leaf path in T (and corresponding tour σ in the metric) which starts at r, and at any internal node v, moves on to the *no* child if $v \in L$, and moves to the *yes* child if $v \in H$. We claim that this tour is a feasible solution to $\mathcal{G}$, the latency group Steiner instance $\mathcal{G}$.

To see why, first consider any scenario $i \in M$ that branched off from path σ in decision-tree T; let v be the vertex where the tree path of scenario i branched off from σ. If $v \in L$ then by the way we defined σ, it follows the "no" child of v, and so $v \in S_i \cap L$. On the other hand, if $v \in H$, then it must be that $v \in H \setminus S_i$ (again from the way σ was defined). In either case, $v \in (S_i \cap L) \cup (H \setminus S_i)$, and hence visiting v covers *all* groups, associated with scenario i, i.e. X_i and $\{Y_i^v \mid v \in S_i \cap H\}$. Thus σ covers all groups of all the scenarios that branched off it in T.

Note that there is exactly one scenario (say $a \in M$) that does not branch off σ; scenario a traverses σ in T. Since T is a feasible solution for AdapTRP, σ must visit every vertex in S_a. Therefore σ covers all the groups associated with scenario a: clearly $\{Y_a^v \mid v \in S_a \cap H\}$ are covered; X_a is also covered unless $S_a \cap L = \emptyset$ (however in that case group X_a has zero weight and does not need to be covered- see Definition 8). Thus σ is a feasible solution to $\mathcal{G}$.

We now bound the latency cost of tour σ for instance $\mathcal{G}$. In path σ, let α_i (for each $i \in M$) denote the coverage time for group X_i, and β_i^v (for $i \in M$ and $v \in S_i \cap H$) the coverage time for group Y_i^v. The next claim shows that the latency of σ for instance $\mathcal{G}$ is at most $\mathsf{Opt}(\mathcal{J})$.

CLAIM 10. *The expected cost of* T, $\mathsf{Opt}(\mathcal{J}) \ge \sum_{i\in M} p_i \cdot |L \cap S_i| \cdot \alpha_i + \sum_{i\in M} \sum_{v\in S_i\cap H} p_i \cdot \beta_i^v$, *which is exactly the latency of tour* σ *for the latency group Steiner instance* $\mathcal{G}$.

Proof: Fix any $i \in M$; let σ_i denote the shortest prefix of σ containing a vertex from X_i. Note that by definition, σ_i has length α_i. We will lower bound separately the contributions of $S_i \cap L$ and $S_i \cap H$ to the cost of T.

As all but the last vertex in σ_i are from $(L \setminus S_i) \cup (H \cap S_i)$, by definition of σ, the path T_{S_i} traced in the decision-tree T when scenario i is realized, agrees with this prefix σ_i. Moreover, no vertex of

$S_i \cap L$ is visited before the end of σ_i. So under scenario S_i, the total arrival time for vertices $L \cap S_i$ is at least $|L \cap S_i| \cdot \alpha_i$. Hence $S_i \cap L$ contributes at least $p_i \cdot |L \cap S_i| \cdot \alpha_i$ towards $\mathsf{Opt}(\mathcal{J})$.

Now consider some vertex $v \in S_i \cap H$; let σ_i^v denote the shortest prefix of σ containing a Y_i^v-vertex. Note that σ_i^v has length β_i^v, and it is a prefix of σ_i since $Y_i^v \supseteq X_i$. As observed earlier, the path traced in decision tree T under scenario i contains σ_i: so vertex v is visited (under scenario i) only after tracing path σ_i^v. So the contribution of v (under scenario i) to $\mathsf{Opt}(\mathcal{J})$ is at least $p_i \cdot \beta_i^v$, i.e. the contribution of $S_i \cap H$ is at least $\sum_{v \in S_i \cap H} p_i \cdot \beta_i^v$ □

Thus we have demonstrated a feasible solution to $\mathcal{G}$ of latency at most $\mathsf{Opt}(\mathcal{J})$. □

It remains to bound the expected additional latency incurred in Step 4 of Algorithm 7 when a random scenario is realized. Below we assume a $\rho = O(\log^2 n)$ approximation algorithm for latency group Steiner tree (from Corollary 1).

LEMMA 4. *At the end of Step 4 of* $\mathsf{AdapTRP}\langle M, \{q_i\}_{i \in M}, \{S_i\}_{i \in M}\rangle$, *the realized scenario lies in* P_{k^*}. *The expected increase in latency due to this step is at most* $2\rho \cdot \mathsf{Opt}(\langle M, \{q_i\}_{i \in M}, \{S_i\}_{i \in M}\rangle)$.

Proof: The proof that the realized scenario always lies in the P_{k^*} determined in Step 4 is identical to that in Claim 3 of the IsoProb algorithm, and is omitted. We now bound the expected latency incurred. In the solution τ to the latency group Steiner instance $\mathcal{G}$, define α_i as the coverage time for group X_i, $\forall i \in M$; and β_i^v as the coverage time for group Y_i^v, $\forall i \in M$ and $v \in S_i \cap H$.

Let i denote the realized scenario. Suppose that $k^* = \ell \le t-1$ in Step 4. Then by definition of the parts P_ks, we have $v_\ell \in X_i = (S_i \cap L) \cup (H \setminus S_i)$ and $X_i \bigcap \{v_1, \ldots, v_{\ell-1}\} = \emptyset$. So the length along τ until v_ℓ equals α_i. Moreover the total length spent in this step is at most $2 \cdot \alpha_i$, to travel till v_ℓ and then return to r (this uses the symmetry and triangle-inequality properties of the metric). So the latency of any S_i-vertex increases by at most this amount. Furthermore we claim that the latency of any $v \in S_i \cap H$ increases by at most $2 \cdot \beta_i^v$: this is clearly true if $\beta_i^v = \alpha_i$; on the other hand if $\beta_i^v < \alpha_i$ then v is visited before v_ℓ and so it only incurs latency β_i^v. So the increase in latency of S_i is at most $2 \sum_{v \in S_i \cap H} \beta_i^v + 2 \cdot |S_i \cap L| \, \alpha_i$.

If $k^* = t$ then by the proof of Claim 8 the realized scenario i satisfies: $S_i \subseteq H$, group X_i is not visited by τ (so α_i is undefined), and all of S_i is visited by τ. In this case the total latency of S_i is $\sum_{v \in S_i \cap H} \beta_i^v$ which is clearly at most $2 \sum_{v \in S_i \cap H} \beta_i^v + 2 \cdot |S_i \cap L| \, \alpha_i$; note that $|S_i \cap L| = 0$ here.

Thus the expected latency incurred in Step 4 is at most $2 \sum_{i \in M} p_i \cdot \left[|S_i \cap L| \, \alpha_i + \sum_{v \in S_i \cap H} \beta_i^v \right]$ which is twice the latency of τ for the latency group Steiner instance $\mathcal{G}$. Finally, since τ is a ρ-approximate solution to $\mathcal{G}$ and using Lemma 3, we obtain the claim. □

Finally, combining Claim 8, Lemma 4 and Claim 9, by a proof identical to that of Theorem 4, it follows that the final AdapTRP solution has cost $O(\log^2 n \log m) \cdot \mathsf{Opt}$. This completes the proof of Theorem 3.

We note that for the AdapTRP problem on metrics induced by a tree, our algorithm achieves an $O(\log n \log m)$ approximation ratio (the guarantees in Theorem 8 and Corollary 1 improve by a logarithmic factor on tree metrics). There is also an $\Omega(\log^{1-\epsilon} n)$-hardness of approximation the AdapTRP problem on tree metrics [32]. So there is still a logarithmic gap between the best upper and lower bounds for the AdapTRP problem on tree metrics. In going from tree metrics to general, we lose another logarithmic factor in the approximation ratio.

7. Concluding Remarks In this paper, we studied the problem of constructing optimal decision trees; this widely studied problem was previously known to admit logarithmic approximation algorithms for the case of uniform costs or uniform probabilities. The greedy algorithms used in these cases do not extend to the case of non-uniform costs and probabilities, and we gave a new algorithm that seeks to be greedy with respect to two different criteria; our $O(\log m)$-approximation is asymptotically optimal. We then considered a generalization to the adaptive traveling salesman problem, and obtained an $O(\log^2 n \log m)$-approximation algorithm for this adaptive TSP problem.

We also showed that any asymptotic improvement on this result would imply an improved approximation algorithm for the group Steiner tree problem, which is a long-standing open problem. Finally, we gave an $O(\log^2 n \log m)$-approximation algorithm for the adaptive traveling repairman problem— closing the gap between the known upper and lower bounds in this case remains an interesting open problem.

Appendix A: Hardness of Approximation for AdapTSP We show that AdapTSP is at least as hard to approximate as group Steiner tree.

THEOREM 12. *If there is an α-approximation algorithm for* AdapTSP *then there is an $(\alpha + o(1))$-approximation algorithm for group Steiner tree. Hence* AdapTSP *is $\Omega(\log^{2-\epsilon} n)$ hard to approximate even on tree metrics.*

Proof: This reduction is similar to the reduction [5] from set-cover to the optimal decision tree problem; we give a proof in context of AdapTSP for completeness.

Consider an arbitrary instance of group Steiner tree on metric (V, d) with root r and groups $X_1, \cdots, X_g \subseteq V$; let Opt denote its optimal value. Assume without loss of generality that $X_i \neq X_j$ for all $i \neq j$, and the minimum non-zero distance in d is one. We construct an instance of AdapTSP as follows. Let $V' = V \cup \{s\}$ where s is a new vertex (representing a copy of r), and define metric d' on V' as:

$$d'(u,v) := \begin{cases} d(u,v) & \text{for } u,v \in V \\ d(u,r) & \text{for } u \in V,\ v = s \end{cases}, \qquad \forall (u,v) \in \binom{V'}{2}$$

There are $g+1$ scenarios in the AdapTSP instance: $S_i := X_i \cup \{s\}$ for $i \in [g]$, and $S_{g+1} := \{s\}$, with probabilities

$$p_i := \begin{cases} \frac{1}{gL} & \text{if } 1 \leq i \leq g \\ 1 - \frac{1}{L} & \text{if } i = g+1 \end{cases},$$

Above $L \gg 2n \cdot \max_{u,v} d(u,v)$ is some large value. The root in the AdapTSP instance remains r. Let Opt′ denote the optimal value time of this instance. We will show that $(1 - o(1)) \cdot \mathsf{Opt} \leq \mathsf{Opt}' \leq \mathsf{Opt} + 1$ which would prove the theorem.

(A) $\left(1 - \frac{1}{L}\right) \mathsf{Opt} \leq \mathsf{Opt}'$**.** Consider the optimal solution to the AdapTSP instance; let σ denote the r-tour traversed by this decision tree under scenario S_{g+1}. We now argue that σ is a feasible solution to the group Steiner tree instance, i.e., $\mathsf{Opt} \leq d(\sigma)$. Suppose for a contradiction that σ does not visit any X_i-vertex for some $i \in [g]$. Then observe that the r-tour traversed by this decision tree under scenario S_i is also σ, since the decision tree can not distinguish scenarios S_i and S_{g+1} (the only way to do this is by visiting some X_i-vertex). However this violates the requirement that the tour (namely σ) under scenario S_i must visit all vertices $S_i \supseteq X_i$. Finally, we have $\mathsf{Opt}' \geq (1 - \frac{1}{L}) \cdot d(\sigma) \geq \left(1 - \frac{1}{L}\right) \mathsf{Opt}$ as required.

(B) $\mathsf{Opt}' \leq \mathsf{Opt} + 1$**.** Let τ denote an optimal r-tour for the given GST instance, so $d(\tau) = \mathsf{Opt}$. Consider the following solution for AdapTSP:

1. Traverse r-tour τ to determine whether or not X_{g+1} is the realized scenario.
2. If no demands observed on τ (i.e. scenario S_{g+1} is realized), visit vertex s and stop.
3. If some demand observed on τ (i.e. one of scenarios $\{S_i\}_{i=1}^g$ is realized), then visit *all* vertices in V along an arbitrary r-tour and stop.

It is clear that this decision tree is feasible for the AdapTSP instance. For any $i \in [g+1]$, let π_i denote the r-tour traversed under scenario S_i in the above AdapTSP decision tree. We have $d(\pi_{g+1}) = d(\tau) \leq \mathsf{Opt}$, and $d(\pi_i) \leq 2n \cdot \max_{u,v} d(u,v) \leq L$ for all $i \in [g]$. Thus the resulting AdapTSP objective is at most:

$$\left(1 - \frac{1}{L}\right) \cdot \mathsf{Opt} + g \cdot \frac{1}{gL} \cdot L \leq \mathsf{Opt} + 1$$

Thus we have the desired reduction. □

Acknowledgments. A preliminary version appeared in the proceedings of the International Colloquium on Automata, Languages and Programming (ICALP), 2010. We thank Ravishankar Krishnaswamy for many useful conversations; the results on the adaptive traveling repairman problem were obtained in joint discussions, and we thank him for permission to include the results here. We also thank the MOR referees for helpful suggestions that improved the presentation of the paper. A. Gupta's research was supported in part by NSF awards CCF-0448095 and CCF-0729022, and an Alfred P. Sloan Fellowship. R. Ravi's research was supported in part by NSF grant CCF-0728841.

References

[1] Adler, Micah, Brent Heeringa. 2012. Approximating optimal binary decision trees. *Algorithmica* **62**(3-4) 1112–1121.

[2] Bansal, Nikhil, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, Atri Rudra. 2012. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica* **63**(4) 733–762.

[3] Blum, A., P. Chalasani, D. Coppersmith, W. R. Pulleyblank, P. Raghavan, M. Sudan. 1994. The minimum latency problem. *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*. 163–171.

[4] Chakaravarthy, Venkatesan, Vinayaka Pandit, Sambuddha Roy, Yogish Sabharwal. 2009. Approximating Decision Trees with Multiway Branches. *ICALP*. 210–221.

[5] Chakaravarthy, Venkatesan T., Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, Mukesh K. Mohania. 2011. Decision trees for entity identification: Approximation algorithms and hardness results. *ACM Transactions on Algorithms* **7**(2) 15.

[6] Charikar, Moses, Chandra Chekuri, Ashish Goel, Sudipto Guha. 1998. Rounding via trees: deterministic approximation algorithms for group Steiner trees and k median. *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*. 114–123.

[7] Chaudhuri, K., B. Godfrey, S. Rao, K. Talwar. 2003. Paths, trees, and minimum latency tours. *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. 36–45.

[8] Chekuri, Chandra, Martin Pál. 2005. A recursive greedy algorithm for walks in directed graphs. *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. 245–253.

[9] Christofides, N. 1977. Worst-case analysis of a new heuristic for the travelling salesman problem. *GSIA, CMU-Report 388* .

[10] Dasgupta, Sanjoy. 2004. Analysis of a greedy active learning strategy. *Advances in Neural Information Processing Systems (NIPS)*.

[11] Dean, Brian C., Michel X. Goemans, Jan Vondrák. 2008. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.* **33**(4) 945–964.

[12] Fakcharoenphol, Jittat, Chris Harrelson, Satish Rao. 2007. The k-traveling repairmen problem. *ACM Transactions on Algorithms* **3**(4).

[13] Fakcharoenphol, Jittat, Satish Rao, Kunal Talwar. 2004. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.* **69**(3) 485–497.

[14] Feige, Uriel, László Lovász, Prasad Tetali. 2004. Approximating min sum set cover. *Algorithmica* **40**(4) 219–234.

[15] Garey, M.R., R.L. Graham. 1974. Performance bounds on the splitting algorithm for binary testing. *Acta Informatica* **3** 347–355.

[16] Garg, N., G. Konjevod, R. Ravi. 2000. A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem. *Journal of Algorithms* **37(1)** 66–84.

[17] Garg, Naveen. 1996. A 3-Approximation for the Minimum Tree Spanning k Vertices. *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*. 302–309.

[18] Goemans, Michel, Jan Vondrák. 2006. Stochastic covering and adaptivity. *LATIN 2006: Theoretical informatics*, *Lecture Notes in Comput. Sci.*, vol. 3887. Springer, Berlin, 532–543.

[19] Golovin, Daniel, Andreas Krause. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res. (JAIR)* **42** 427–486.

[20] Guha, Sudipto, Kamesh Munagala. 2009. Multi-armed bandits with metric switching costs. *ICALP*. 496–507.

[21] Guillory, Andrew, Jeff Bilmes. 2009. Average-Case Active Learning with Costs. *Algorithmic Learning Theory*. Springer Berlin / Heidelberg, 141–155.

[22] Gupta, Anupam, Mohammad T. Hajiaghayi, Harald Räcke. 2006. Oblivious network design. *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. 970–979.

[23] Halperin, Eran, Robert Krauthgamer. 2003. Polylogarithmic inapproximability. *Proceedings of the 35th Annual Symposium on Theory of Computing*. 585–594.

[24] He, Ting, Kang-Won Lee, Ananthram Swami. 2010. Flying in the dark: controlling autonomous data ferries with partial observations. *MobiHoc*. 141–150.

[25] Hyafil, Laurent, Ronald L. Rivest. 1976/77. Constructing optimal binary decision trees is NP-complete. *Information Processing Lett.* **5**(1) 15–17.

[26] Jaillet, Patrick. 1988. A priori solution of a travelling salesman problem in which a random subset of the customers are visited. *Operations Research* **36 (6)**.

[27] Jia, Lujun, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, Ravi Sundaram. 2005. Universal approximations for TSP, Steiner tree, and set cover. *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. 386–395.

[28] Kosaraju, S. Rao, Teresa M. Przytycka, Ryan S. Borgstrom. 1999. On an Optimal Split Tree Problem. *Proceedings of the 6th International Workshop on Algorithms and Data Structures*. 157–168.

[29] Liu, Zhen, Srinivasan Parthasarathy, Anand Ranganathan, Hao Yang. 2008. Near-optimal algorithms for shared filter evaluation in data stream systems. *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 133–146.

[30] Loveland, Donald W. 1985. Performance bounds for binary testing with arbitrary weights. *Acta Inform.* **22**(1) 101–114.

[31] Munagala, Kamesh, Utkarsh Srivastava, Jennifer Widom. 2007. Optimization of continuous queries with shared expensive filters. *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 215–224.

[32] Nagarajan, Viswanath. 2009. Approximation algorithms for sequencing problems. Ph.D. thesis, Tepper School of Business, Carnegie Mellon University.

[33] Nowak, Robert D. 2011. The geometry of generalized binary search. *IEEE Transactions on Information Theory* **57**(12) 7893–7906.

[34] Schalekamp, F., D. Shmoys. 2008. Algorithms for the universal and a priori TSP. *Operations Research Letters* **36(1)** 1–3.

[35] Shah, Rahul C., Sumit Roy, Sushant Jain, Waylon Brunette. 2003. Data mules: modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks* **1**(2-3) 215–233.

[36] Shmoys, David, Kunal Talwar. 2008. A Constant Approximation Algorithm for the a priori Traveling Salesman Problem. *Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization*. 331–343.

[37] Sviridenko, M. 2004. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters* **32** 41–33.

[38] Zhao, Wenrui, Mostafa H. Ammar. 2003. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. *FTDCS*. 308–314.

[39] Zhao, Wenrui, Mostafa H. Ammar, Ellen W. Zegura. 2004. A message ferrying approach for data delivery in sparse mobile ad hoc networks. *MobiHoc*. 187–198.

[40] Zhao, Wenrui, Mostafa H. Ammar, Ellen W. Zegura. 2005. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. *INFOCOM*. 1407–1418.