Approximation Algorithms for Optimal Decision Trees and Adaptive TSP Problems

Anupam Gupta* Ravishankar Krishnaswamy* Viswanath Nagarajan† R. Ravi‡

Abstract

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 the patient having any particular disease, what is a good adaptive strategy to perform these tests to minimize the expected cost to identify the disease?* We settle the approximability of this problem by giving a tight $O(\log m)$-approximation algorithm. The optimal decision tree problem was known to be $\Omega(\log m)$-hard to approximate, and previously $O(\log m)$-approximations were known only under either uniform costs or uniform probabilities.

We also consider a more substantial generalization, the *Adaptive TSP* problem. Given an underlying metric space, a random subset S of cities is drawn from a known distribution, but S is initially unknown to us—we get information about whether any city is in S only when we visit the city in question. What is a good adaptive way of visiting all the cities in the random subset S while minimizing the expected distance traveled? For this adaptive TSP problem, we give the first poly-logarithmic approximation, and show that this algorithm is best possible unless we can improve the approximation guarantees for the well-known group Steiner tree problem. We also give an approximation algorithm with the same guarantee for the *adaptive traveling repairman* problem.

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), distinct subsets $S_1, S_2, \ldots, S_m \subseteq V$ such that S_i appears with probability p_i (and $\sum_i p_i = 1$), and she needs to serve requests at this subset of locations. 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. What adaptive strategy should she use to minimize the expected time to serve all requests?

- *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?

*Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Supported in part by NSF awards CCF-0448095 and CCF-0729022, and an Alfred P. Sloan Fellowship.

†IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA.

‡Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Supported in part by NSF grant CCF-0728841.

It can be shown that the optimal decision tree problem is a special case of the former problem (a formal reduction is given in Section 5.) In both these problems we want to devise adaptive strategies, which take into account the revealed information in the queries so far (e.g., cities already visited, or tests already done) to determine the future course of action—i.e., our output must be a decision tree. In contrast, a non-adaptive strategy corresponds to a decision list. While some problems have the property that the best adaptive and non-adaptive strategies have similar performances, both the above problems have a large *adaptivity gap* [10] (the worst ratio between the performance of the optimal non-adaptive and optimal adaptive strategies). Hence it is essential that we seek good *adaptive* strategies.

The *optimal decision tree problem* has long been studied (its NP-hardness was shown by Hyafil and Rivest in 1976 [21], and many references and applications can be found in [29]). On the algorithms side, $O(\log m)$-approximation algorithms have been given for the special cases where the likelihoods $\{p_j\}$ are all equal, or where the test costs $\{c_t\}$ are all equal [13, 26, 24, 9, 1, 4, 29, 18]. But the problem has remained open for the case when both the probabilities and the costs are arbitrary—the previous results only imply $O\left(\log \frac{1}{p_{\min}}\right)$ and $O\left(\log(m \frac{c_{max}}{c_{min}})\right)$ approximation algorithms for the general case. An $O(\log m)$-approximation for general costs and probabilities would be the best possible unless $NP \subseteq DTIME[n^{O(\log\log n)}]$ [4]; and while the existence of such an algorithm has been posed as an open question, it has not been answered prior to this work.

In this paper, we settle the approximability of the optimal decision tree problem:

Theorem 1 *There is an $O(\log m)$-approximation for the optimal decision tree problem with arbitrary test costs and arbitrary probabilities; 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 $O(\log^2 n \log m)$-factor approximation algorithm for the adaptive TSP problem.*

In this paper, we first prove the latter result for adaptive TSP, and then show how that algorithm specialized to the optimal decision tree problem actually gives us the claimed $O(\log m)$-factor approximation.

A Word about our Techniques. 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, say, Christofides' heuristic. The idea behind our algorithm for Isolation is this—suppose each vertex lies in at most half the scenarios; if we visit one vertex in each of the *m* scenarios using a short tour (which is just group Steiner tree [14]), we'd notice at least one of these vertices to have a demand—this would reduce the number of possible scenarios by 50%. This is necessarily 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 *min-latency* version of group Steiner tree; not all vertices need lie in less than half the scenarios. Another major issue is that moreover we do not want our performance to depend on the size of the probabilities, in case some of them are exponentially small, so we cannot just hope to reduce the measure of the remaining scenarios by 50%. Finally, we need to charge our cost directly against the optimal decision tree. All these issues can be resolved to give an $O(\log^2 n \cdot \log m)$ approximation for Isolation (and hence for AdapTSP) with *n* nodes and *m* scenarios. The Isolation algorithm also involves an interesting combination of ideas from the group Steiner [14, 5] and minimum latency [2, 6, 11] problems—it uses a greedy strategy that is greedy with respect to two different criteria, namely both the probability measure and the number of scenarios. This idea is formalized in our definition of the *partial latency* group Steiner (LPGST) problem, and its algorithm (see Section 4). In Appendix A, we show that both AdapTSP and Isolation are $\Omega(\log^{2-\varepsilon} n)$ 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.

For the special case of the optimal decision tree problem, we show that we can use the min-sum set cover problem [12] instead of the min-latency version of group Steiner tree; this avoids an $O(\log^2 n)$ loss in the approximation guarantee, and hence gives us an optimal $O(\log m)$-approximation for the optimal decision tree problem. 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 Chakravarthy et al. [4].

Finally, we consider the adaptive traveling repairman problem (AdapTRP), which has the same input as AdapTSP, but we want to minimize the expected average response time of the vertices in the materialized demand set. Note that we cannot first isolate the scenario and then visit all its nodes, since a long isolation tour may negatively impact the response times. However, we show that our techniques for AdapTSP are robust, and can be used to get a similar approximation algorithm for the AdapTRP.

A brief outline of the paper layout: the results on AdapTSP and Isolation appear in Section 3, the algorithm for partial latency group Steiner appears in Section 4, and the hardness result in Appendix A. We show how to use the AdapTSP and Isolation results to get improved algorithms for optimal decision tree in Section 5. The details for the adaptive traveling repairman problem are in Section 6.

1.1 Other Related Work

The optimal decision tree problem has been studied earlier by many authors, with algorithms and hardness results being shown by [13, 21, 26, 24, 1, 9, 4, 3, 18]; as mentioned above, the algorithms in these papers give $O(\log m)$-approximations only when either the probabilities or costs (or both) are polynomially-bounded. Table 1 in Guillory and Bilmes [18] gives a good summary of known approximation ratios; in general, the best approximation guarantees are $O\left(\log \frac{1}{p_{\min}}\right)$ and $O\left(\log(m \frac{c_{max}}{c_{min}})\right)$. The $O(\log m)$-approximation for arbitrary costs and probabilities solves an open problem from these papers. The AdapTSP problem has not been studied earlier, to the best of our knowledge.

There are many results on adaptive optimization. E.g., [16] considered an adaptive set-cover problem; they give 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. This was improved in [27] to $O(\log^2 n \log m)$, and subsequently to the best-possible $O(\log n)$-approximation by [25], also using a greedy algorithm. [7] considered adaptive strategies for stochastic routing problems using a greedy-like algorithm where they solved an LP at each step. [17] studied adaptive transmission in noisy channels (that have multiple states) to maximize the utility (i.e., the difference between success-probability and probing-cost). In all these problems, the adaptivity-gap is large, as is the case for the problems we considered here, and hence the solutions need to be inherently adaptive.

The AdapTSP problem is related to universal TSP [23, 19] and *a priori* TSP [22, 30, 31] only in spirit—in both the universal and *a priori* TSP problems, we seek a master tour which we 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 Notation

Throughout this paper, we deal with demand distributions over vertex-subsets that are specified explicitly. I.e. demand distribution $\mathscr{D}$ 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 all adaptive strategies.

Definition 1 (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 P_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 P_{S_i}, and the expected cost of T (under distribution $\mathscr{D}$) is then $\sum_{i=1}^{m} p_i \cdot C_i$. For example in AdapTSP, cost C_i corresponds to the length of path P_{S_i}.
Given a metric (V,d) and root $r \in V$, an *r-tour* is any tour that begins and ends at r.

3 The Adaptive TSP and Isolation Problems

The input to *adaptive TSP* is a metric (V,d) with root $r \in V$ and a demand distribution $\mathscr{D}$ over subsets of vertices. The crucial aspect in this model is that 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 $\mathscr{D}$.

We assume that the distribution $\mathscr{D}$ is specified *explicitly* with a support-size of m. The more general setting would be to consider black-box access to distribution $\mathscr{D}$: however, as shown in [28], AdapTSP under general black-box distributions admits no $o(n)$ approximation if the algorithm is restricted to polynomially many samples. This is the reason we consider an explicit demands model for $\mathscr{D}$. Moreover, AdapTSP under explicit demands still contains interesting special cases such as optimal decision tree problem. One could also consider AdapTSP under independent demand distributions; note that in this case there is a trivial solution: visit all vertices having non-zero probability along an approximately minimum TSP tour. In terms of Definition 1, we have:

Definition 2 (The AdapTSP Problem) *Given metric (V,d), root r and demand distribution $\mathscr{D}$, the goal in AdapTSP is to compute a decision tree T in metric (V,d) with the added conditions that:*

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

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

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

Definition 3 (The Isolation Problem) *Given metric (V,d), root r and demand distribution $\mathscr{D}$, the goal in Isolation is to compute a decision tree T in metric (V,d) with the added conditions that:*

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

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

Note the only difference between this definition and the one for AdapTSP is that the tree path P_{S_i} need not contain all vertices of S_i, and the paths for different scenarios should end at distinct leaf-nodes. We first prove the following simple fact relating these two objectives.

Lemma 3 *An α-approximation algorithm for Isolation implies an $\left(\alpha + \frac{3}{2}\right)$ approximation for AdapTSP.*

Proof: We first claim that any feasible solution T to AdapTSP is also feasible for Isolation. For this it suffices to show that $P_{S_i} \neq P_{S_j}$ for any two scenarios $i, j \in [m]$ with $i \neq j$. Suppose (for a contradiction) that paths $P_{S_i} = P_{S_j} = \pi$ 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 P_{S_i} must take the *yes* child, whereas path P_{S_j} must take the *no* child. This contradicts the assumption $P_{S_i} = P_{S_j} = \pi$. Thus any solution to AdapTSP is also feasible for Isolation; moreover the expected cost remains the same. Hence the optimal value of Isolation is at most that of AdapTSP.

Now, using an α-approximation for Isolation, we obtain a strategy 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 strategy for AdapTSP:

1. Run T' to determine the realized scenario $k \in [m]$, and return to r.

2. Traverse a $\frac{3}{2}$-approximate TSP tour [8] 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. Hence we obtain a strategy that has expected cost at most $(\alpha + \frac{3}{2})\mathsf{Opt}$, as claimed. ■

In the next few subsections, we give an $O(\log^2 n \log m)$-approximation algorithm for Isolation, which by this lemma implies the same result for AdapTSP. In Appendix A we complement this result by showing that AdapTSP is at least as hard to approximate as the well-known Group Steiner Tree problem, for which the best approximation ratio is $O(\log^2 n \log m)$.

3.1 Approximation Algorithm for the Isolation Problem

Recall that an instance of Isolation 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 $\{p_i\}_{i=1}^{m}$. At a high level, our approach is a simple divide-and-conquer based one. The algorithm for Isolation is recursive: given an instance, it first develops a strategy to generate several sub-instances from this instance, each given by some proper subset $M \subseteq [m]$ of scenarios, with associated probabilities $\{q_i\}_{i \in M}$ where $\sum_{i \in M} q_i = 1$. (We refer to such a sub-instance of the original Isolation instance as $\langle M, \{q_i\}_{i \in M} \rangle$.) We then recursively build an isolation strategy within each sub-instance. The real question is: *How do we generate these sub-instances so that we can charge these to the cost of the best decision tree?*

A useful subroutine to address this and solve Isolation will be the *partial latency group Steiner* (LPGST) problem, where we are given 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 $h \leq g$. A group $i \in [g]$ is said to be *covered* (or visited) by r-tour τ if any vertex in X_i is visited, and the *arrival time* of group i is the length of the shortest prefix of τ that contains an X_i-vertex. The arrival times of all uncovered groups are set to be the tour-length. The weighted sum of arrival times of all groups is termed *latency* of the tour, i.e.,

$$\text{latency}(\tau) = \textstyle\sum_{i \text{ covered}} w_i \cdot \text{arrival time}_\tau(X_i) + \sum_{i \text{ uncovered}} w_i \cdot \text{length}(\tau). \tag{3.1}$$

Note that this objective function generalizes the standard min-latency objective, where we have to cover *all* groups. In Section 4 we prove the following result:

Theorem 4 *There is an $(O(\log^2 n), 4)$-bicriteria approximation algorithm for* LPGST. *I.e., the tour output by the algorithm visits at least $\frac{h}{4}$ groups and has latency at most $O(\log^2 n)$ times the optimal latency of a tour that visits h groups.*

In this subsection, we present the algorithm for Isolation assuming this result for LPGST. Let us begin by showing the partitioning algorithm.

The Partitioning Algorithm The algorithm Partition (given below as Algorithm 1) finds an r-tour τ in the metric such that after observing the demands on τ, the number of scenarios that are consistent with these observations is only a constant fraction of the total. E.g., if there was a vertex v such that $\approx 50\%$ of the scenarios contained it, then visiting vertex v would reduce the candidate scenarios by $\approx 50\%$, irrespective of the observation at v, giving us a tangible notion of progress. However, each vertex may give a very unbalanced partition or such a vertex may be too expensive to visit, so we may have to visit multiple vertices—this is the basic idea in algorithm Partition.

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 $D_v := \begin{cases} F_v & \text{if } |F_v| \le \frac{g}{2} \\ M \setminus F_v & \text{if } |F_v| > \frac{g}{2} \end{cases}$

2: **for each** $i \in M$, set $X_i \leftarrow \{v \in V \mid i \in D_v\}$. Note that either the presence of a demand at some vertex v reduces the number of still-possible scenarios by half, or the absence of demand does so. To handle this asymmetry, this step takes scenarios $\{S_i\}_{i\in M}$ and "flips" some vertices to get X_i.

3: **run** the LPGST algorithm (Theorem 4) with metric (V, d) with root r, groups $\{X_i\}_{i\in M}$ with weights $\{q_i\}_{i\in M}$, and target $|M| - 1$.
  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 $\{P_k\}_{k=1}^t$.

The Isolation Algorithm To follow a divide-and-conquer strategy, the final algorithm is fairly natural given the partitioning scheme. The algorithm IsoAlg (given as Algorithm 2) proceeds in several phases. In each phase, it maintains a candidate set M of scenarios such that the realized scenario lies in M. Upon observing demands along the tour produced by algorithm Partition (in Step 2), a new set $M' \subseteq M$ containing the realized scenario is identified such that the number of candidate scenarios reduces by a constant factor (i.e. $|M'| \le \frac{7}{8} \cdot |M|$); then IsoAlg recurses on scenarios M'. After $O(\log m)$ such phases 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** $q'_j := \sum_{i\in P_k} q_i$ **for all** $j \in 1 \ldots t$.

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.

Remark on possible simplification. We use the LPGST problem as a subroutine in solving Isolation. The reason behind this is that we measure progress in the recursive scheme IsoAlg as the *number* of candidate scenarios, whereas in computing the objective we need to use the *probabilities* of scenarios. A seemingly simpler approach is to use in place of LPGST, a 'density' Group Steiner Tree problem, with profits being either (A) uniform (corresponding to measuring number of scenarios) or (B) the scenario probabilities. An $O(\log^2 n)$-approximation for such a density Group Steiner problem was achieved in [5], so this could be used directly instead of the LPGST result in Section 4. However there are non-trivial issues with these potential simplifications-

A. If we were to use unit-profits, then it is no longer possible to charge the length of the solution (which covers at least constant fraction of the groups) to the optimal adaptive solution. This is because the expected cost of the optimal solution is weighted by the scenarios' probabilities (see Claim 5).

B. If we were to use probabilities as profits, then we could charge the solution (that covers constant probability of groups) to the optimal adaptive solution. But in this case, the progress in a single phase would only reduce the *probability mass* on candidate scenarios by a constant factor; so the number of phases can only be bounded by $O(\log \frac{1}{p_{min}})$ where p_{min} is the minimum probability of any scenario. This would still work if we could preprocess the instance to ensure poly-bounded probabilities. However, as shown by the following instance, this preprocessing appears quite non-trivial even in the special case of the optimal decision tree problem. There is a star-metric on vertices $V = \{r\} \bigcup \{v_i\}_{i=1}^{n-1}$ centered at r, and $d(r, v_i) = 2^i$ (for all $i \in [n-1]$). There are n scenarios: $S_i = \{v_i\}$ for $i \in [n-1]$ and $S_n = \emptyset$, with probabilities $p_i = \frac{1}{2^i}$ (for $i \in [n-1]$) and $p_n = \frac{1}{2^{n-1}}$.

This is also the reason why our final algorithm interleaves both greedy objectives: number and probabilities of scenarios. See Section 4 for more details.

3.2 The Analysis for Algorithm IsoAlg

We now prove the performance guarantee and correctness of Algorithm IsoAlg. By the construction in algorithm IsoAlg, it follows that the realized scenario is identified after $O(\log m)$ recursive calls to IsoAlg. To complete the analysis, we need two additional ideas: (1) relating the LPGST and Isolation objective-values in each call to IsoAlg (which bounds the cost in a single sub-instance), and (2) establishing a subadditivity property of Isolation, that bounds the expected cost across all 'parallel' sub-instances (i.e. those in the same phase).

For any instance $\mathscr{I}$ of the isolation problem, let $\mathsf{IsoTime}^*(\mathscr{I})$ denote its optimal value.

Claim 5 *For any instance $\mathscr{I} = \langle M, \{q_i\}_{i\in M}\rangle$, the optimal value of the* LPGST *instance considered in Step 3 of Algorithm* $\mathsf{Partition}(\mathscr{I})$ *is at most* $\mathsf{IsoTime}^*(\mathscr{I})$*—i.e., the* LPGST *instance costs at most the isolation instance.*

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

Consider an r-tour σ that starts at the root r of T and moves from each node v to its unique child that corresponds to $M \setminus D_v$, until it reaches a leaf-node l, when it returns to r. Let the vertices in this tour σ be $r, u_1, u_2, \cdots, u_j, r$ (where u_j is the last vertex visited in T before leaf l). Since T is a feasible decision tree for the isolation instance, the leaf l is labeled by a unique scenario $a \in M$ such that when T is run under demands S_a, it traces path from the root to leaf node l. Moreover, every scenario $b \in M \setminus \{a\}$ gives rise to a root-leaf path that diverges from the root-l path (i.e. σ). From the way we constructed σ, the scenarios that diverged from σ were precisely $\cup_{k=1}^{j} D_{u_k}$, and hence $\cup_{k=1}^{j} D_{u_k} = M \setminus \{a\}$.

Now consider the r-tour σ as a potential solution to the LPGST instance in Step 3. Since $|\cup_{k=1}^{j} D_{u_k}| = |M| - 1$, the definition of the X_i's says that this path hits $|M| - 1$ of these sets X_i, so it is indeed feasible. Also, the definition of the isolation cost implies that the latency of this tour σ is at most the isolation cost $\mathsf{IsoTime}(T) = \mathsf{IsoTime}^*(\mathscr{I})$. ■

If we use a $(\rho_{\mathsf{LPGST}}, 4)$-bicriteria LPGST algorithm, we get the following claim:

Claim 6 *The latency of tour τ returned by Algorithm* Partition *is at most $\rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i\in M}\rangle)$. Furthermore, the resulting partition $\{P_k\}_{k=1}^t$ has each $|P_k| \le \frac{7}{8}|M|$ for each $k \in [t]$, when $|M| \ge 2$.*

Proof: By Claim 5, the optimal value of the LPGST instance in Step 3 of algorithm Partition is at most $\mathsf{IsoTime}^*(\langle M, \{q_i\}_{i\in M}\rangle)$; now the approximation guarantee from Theorem 4 implies that the latency of the solution tour τ is at most ρ_{LPGST} 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 LPGST algorithm in Step (3) of algorithm Partition; and $\{P_k\}_{k=1}^t$ the resulting partition. Claim 5 and Theorem 4 imply that the number of groups covered by τ is $|\cup_{k=1}^{t-1} D_{v_k}| \ge \frac{|M|-1}{4} \ge \frac{|M|}{8}$ (when $|M| \ge 2$). By definition of the sets D_v, it holds that $|D_v| \le |M|/2$ for all $v \in V$; moreover, since all but the last part P_k is a subset of some D_v, it holds that $|P_k| \le \frac{|M|}{2}$ for $1 \le k \le t-1$. Moreover, the set P_t has size $|P_t| = |M \setminus (\cup_{j<t} D_{v_j})| \le \frac{7}{8}|M|$. ■

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

Claim 7 *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 $2\rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i\in M}\rangle)$.*

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:

- $1 \le k^* \le t-1$. Tour returns directly to r from the first vertex v_k (for $1 \le k \le t-1$) such that the realized scenario lies in D_{v_k}. Note that 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 $\{D_{v_k} \mid 1 \le k \le t-1\}$, and it is inferred to be in $M \setminus (\cup_{j<t} D_{v_j})$, 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 for group X_i in tour τ; recall that this is the length of the shortest prefix of τ until it hits an X_i-vertex, and is set to the entire tour length if τ does not ever hit X_i. The construction of partition $\{P_k\}_{k=1}^t$ from τ implies that

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

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. Hence the expected length traversed is at most:

$$2 \cdot \sum_{k=1}^t \sum_{i\in P_k} q_i \cdot \left(\sum_{j=1}^k d(v_{j-1}, v_j)\right) = 2 \cdot \sum_{i\in M} q_i \cdot \alpha_i = 2 \cdot \mathsf{latency}(\tau),$$

and by Claim 6, this is at most $2 \cdot \rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i\in M}\rangle)$. ■

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

Claim 8 *For any instance $\langle M, \{q_i\}_{i\in M}\rangle$ and any partition $\{P_k\}_{k=1}^t$ of M,*

$$\sum_{k=1}^t q'_k \cdot \mathsf{IsoTime}^*(\langle P_k, \{\tfrac{q_i}{q'_k}\}_{i\in P_k}\rangle) \le \mathsf{IsoTime}^*(\langle M, \{q_i\}_{i\in M}\rangle), \tag{3.2}$$

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 $\mathscr{J}_0 := \langle M, \{q_i\}_{i\in M}\rangle$. For each $k \in [t]$, consider instance $\mathscr{J}_k := \langle P_k, \{\frac{q_i}{q'_k}\}_{i\in P_k}\rangle$; one feasible adaptive strategy for instance $\mathscr{J}_k$ 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 strategy for $\mathscr{J}_k$ is $\sum_{i\in P_k} \frac{q_i}{q'_k} \cdot d(\pi_i)$ where π_i denotes the tour traced by T under scenario $i \in P_k$. Hence $\mathsf{Opt}(\mathscr{J}_k) \le \sum_{i\in P_k} \frac{q_i}{q'_k} \cdot d(\pi_i)$. Summing over all parts $k \in [t]$, we get

$$\sum_{k=1}^{t} q'_k \cdot \mathsf{Opt}(\mathscr{J}_k) \le \sum_{k=1}^{t} q'_k \cdot \sum_{i\in P_k} \frac{q_i}{q'_k} \cdot d(\pi_i) = \sum_{i\in M} q_i \cdot d(\pi_i) = \mathsf{Opt}(\mathscr{J}_0), \tag{3.3}$$

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

Given the above claims, we finally bound the expected cost of the strategy given by our algorithm.

Theorem 9 *The expected length of the strategy given by* $\mathsf{IsoAlg}\langle M, \{q_i\}_{i\in M}\rangle$ *is at most:*

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

Proof: We prove this by induction on $|M|$. The base case of $|M| = 1$ is trivial, since zero length is traversed, and hence we consider $|M| \ge 2$. Let instance $\mathscr{I}_0 := \langle M, \{q_i\}_{i\in M}\rangle$. For $k \in [t]$, consider the sub-instance $\mathscr{I}_k := \langle P_k, \{\frac{q_i}{q'_k}\}_{i\in P_k}\rangle$, where $q'_k = \sum_{i\in P_k} q_i$. By the inductive hypothesis, for any $k \in [t]$, the expected length of $\mathsf{IsoAlg}(\mathscr{I}_k)$ is at most $2\rho_{\mathsf{LPGST}} \cdot \log_{8/7} |P_k| \cdot \mathsf{IsoTime}^*(\mathscr{I}_k) \le 2\rho_{\mathsf{LPGST}} \cdot (\log_{8/7} |M| - 1) \cdot \mathsf{IsoTime}^*(\mathscr{I}_k)$, since $|P_k| \le \frac{7}{8}|M|$ (from Claim 6 as $|M| \ge 2$).

By Claim 7, the expected length traversed in Step 4 of $\mathsf{IsoAlg}(\mathscr{I}_0)$ is at most $2\rho_{\mathsf{LPGST}} \cdot \mathsf{IsoTime}^*(\mathscr{I}_0)$. The probability of recursing on $\mathscr{I}_k$ is exactly q'_k for each $k \in [t]$, hence the expected length of $\mathsf{IsoAlg}(\mathscr{I}_0)$ is at most:

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

where the third inequality uses Claim 8. ■

Applying Theorem 9 on the original $\mathsf{Isolation}$ instance gives an $O(\rho_{\mathsf{LPGST}} \cdot \log m)$-approximately optimal algorithm; using Theorem 4 implies $\rho_{\mathsf{LPGST}} = O(\log^2 n)$ and the following:

Theorem 10 *There is an* $O(\log^2 n \cdot \log m)$ *approximation for* $\mathsf{Isolation}$ *with n vertices and m scenarios. Hence, there is an* $O(\log^2 n \cdot \log m)$ *approximation for* $\mathsf{AdapTSP}$.

A comment about the optimality of these results: we show in Appendix A that $\mathsf{AdapTSP}$ (and by Lemma 3, $\mathsf{Isolation}$ as well) is as hard to approximate as the group Steiner problem, and hence any improvement here will make progress on that problem too.

4 Solving Group Steiner Variants useful for Isolation

The input to the well-studied *group Steiner tree problem* (GST) consists of metric (V, d) with root $r \in V$ and groups $\{X_i \subseteq V\}_{i=1}^{g}$; the goal is to compute a minimum length r-tour covering all group, where a group $i \in [g]$ is said to be covered if any vertex from X_i is visited by the tour. The best approximation guarantee known for GST is $O(\log^2 n \cdot \log g)$ [14], and there is an $\Omega(\log^{2-\varepsilon} n)$ hardness of approximation [20]. Next, we study two variants of group Steiner tree that are useful in obtaining an approximation algorithm for $\mathsf{Isolation}$.

- In the *latency* group Steiner tree problem, in addition to the GST input we are also given non-negative weights for each group, and the objective is to compute an r-tour covering all groups that minimizes the *sum of the weighted arrival times*, where the arrival time at a group is the distance along the tour from the root r to the first vertex in that group.

- In the group Steiner *orienteering* problem, in addition to the GST input there are profits on groups and a length bound B; the goal is to compute an r-tour of length at most B covering maximum possible profit.

It is somewhat intuitive that the latency group Steiner problem is related to Isolation: in order to isolate the scenarios we need to hit at least one vertex from at least $g-1$ scenarios, and we need to do this with small expected latency. However, we need an adaptive (rather than a linear) strategy, and we need to isolate scenarios rather than just hit them; nonetheless, this problem will be useful as a subroutine, and we show the reduction in Section 3.1. It is easy to show (following [2, 6, 11]) that an approximation algorithm for the orienteering objective leads to an approximation for the latency objective. Hence we first give an algorithm for group Steiner orienteering. Then we show how this algorithm for group Steiner orienteering implies an algorithm for a generalization of latency group Steiner (namely LPGST), that is particularly convenient in the algorithm for Isolation.

4.1 Group Steiner Orienteering

The *group Steiner orienteering* (GSO) problem takes as input, a metric (V,d) with 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 is to compute an r-tour of length at most B that collects the maximum possible profit (from the groups that are visited by the tour).

In this section, we show that the deterministic algorithm for group Steiner tree [5] can be used within a standard greedy framework to obtain a tour slightly longer than the length bound B, and which gets a constant fraction of the profit obtained by the best length-B tour. In particular, an algorithm for GSO is said to be an (a,b)-bicriteria approximation if on any instance of the problem, it outputs a solution of length at most $b \cdot B$ that obtains at least $\frac{1}{a}$ times the best profit obtained by any r-tour of length B.

Theorem 11 *There is a $(4, O(\log^2 n))$-bicriteria approximation algorithm for* GSO*, where n is the number of vertices in the metric.*

Proof: Let Opt denote the optimal profit of the given GSO instance. We first preprocess the metric to only include vertices within distance $B/2$ of the root r: note that since the optimal tour cannot visit any excluded vertex, the optimal profit remains unchanged by this. Note that now the profit achieved by visiting any remaining vertex is at most Opt. Also every group is covered by some vertex (otherwise the group can be dropped), i.e. the sum of profits achieved by visiting all vertices is at least $\sum_{i=1}^{g} \phi_i$. Hence there is some vertex with profit at least $\frac{1}{n}\sum_{i=1}^{g} \phi_i$, and we have $\mathsf{Opt} \geq \frac{1}{n}\sum_{i=1}^{g} \phi_i$.

Our algorithm for GSO follows a standard greedy approach (see eg. Garg [15]), and is given as Algorithm 3. In the following, let $\alpha := O(\log^2 n)$, where the precise constant comes from the analysis.

We now prove that Algorithm 3 achieves a $(4, 4\alpha+2)$ bicriteria approximation. By the description of the algorithm, we iterate as long as the total length of edges in S is bounded by αB; since the increase in length of S in any iteration is at most $(\alpha+1)\cdot B$ (recall that every vertex in A is at distance at most B from the root r), the final length $d(S) \leq (2\alpha+1)\cdot B$, and the Euler tour costs at most twice that. This proves the bound on the length; to prove Theorem 11, it now suffices to show that the final subgraph S gets profit at least $\frac{\mathsf{Opt}}{4}$. Moreover, at any iteration, let $p(S)$ denote the profit of S, and $d(S)$ its length. Since $d(S) > \alpha B$ upon termination, it suffices to show the following invariant for subgraph S over the

Algorithm 3 Algorithm for GSO.

1: **initialize** solution $S \leftarrow \emptyset$. **Mark** all groups as uncovered.
2: **while** length of S 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: **solve** the following LP for the residual GSO to obtain solution (x,y):

$$\begin{aligned} \max \quad & \textstyle\sum_{i=1}^{g} \widetilde{\phi}_i \cdot y_i \\ \text{s.t.} \quad & x(\delta(S)) \geq y_i \quad \forall S \subseteq V : r \notin S,\ X_i \subseteq S;\ \ \forall i \in [g] \\ & y_i \leq 1 \qquad \forall i \in [g] \\ & \textstyle\sum_e d_e \cdot x_e \leq B \\ & x, y \geq 0 \end{aligned}$$

5: As described in [5] (using probabilistic tree embedding), obtain a spanning tree T in metric (V,d) and capacities x_T on edges of T such that: $\sum_{e\in T} d_e \cdot x_T(e) \leq O(\log n) \cdot \sum_e d_e \cdot x_e \leq O(\log n) \cdot B$, and these capacities x_T support at least y_i flow from the root r to the group X_i for each $i \in [g]$.
6: **round** down each $x_T(e)$ to an integral multiple of $\frac{1}{n^3}$.
7: For each group $i \in [g]$, let y'_i be the maximum flow value from r to the group X_i under these capacities x_T, and let $f_i^T(e)$ (for all $e \in T$) denote the flow variables realizing this flow.
8: Using x_T and f_i^Ts, run the deterministic algorithm for 'density group Steiner' ([5, Section 3.2]) to obtain subtree A rooted at r that covers groups $\widehat{A}$ such that:

$$\frac{d(A)}{\sum_{i\in\widehat{A}} \widetilde{\phi}_i} \leq \alpha \cdot \frac{B}{\sum_{i=1}^{g} \widetilde{\phi}_i \cdot y'_i}, \qquad \text{where } d(A) := \textstyle\sum_{e\in A} d_e \text{ and } \alpha = O(\log^2 n)$$

9: **if** $d(A) \leq \alpha B$ then $S' \leftarrow S \cup A$.
10: **if** $d(A) > \alpha B$ then: (i) partition tree A into at most $2 \cdot \frac{d(A)}{\alpha B}$ subtrees, each of length at most αB; let A' denote the subtree containing maximum profit. (ii) Set $S' \leftarrow S \cup A' \cup \{f\}$ where f is the shortest edge from r to A'.
11: **Set** $S \leftarrow S'$. Mark all groups visited in S as covered.
12: **end while**
13: **output** an Euler tour of the subgraph S.

iterations of the algorithm:

$$p(S) \geq \min\left\{ \frac{\mathsf{Opt}}{4}, \frac{\mathsf{Opt}}{2\alpha B} \cdot d(S) \right\} \tag{4.4}$$

At the beginning, inequality (4.4) holds trivially since $d(S) = 0$ for $S = \emptyset$. Consider any iteration where $p(S) < \mathsf{Opt}/4$ at the beginning (otherwise the claim is trivial); the invariant now ensures that $d(S) < \alpha B/2$ and hence we proceed further with the steps of the algorithm. Moreover, the optimal value of the LP solved in this iteration $\sum_{i=1}^{g} \widetilde{\phi}_i \cdot y_i \geq \mathsf{Opt} - p(S) \geq \frac{3}{4} \cdot \mathsf{Opt}$. After the pruning step (that rounds down capacities) we reduce the capacity of each edge in T by at most $\frac{1}{n^3}$. Since any cut in the tree T has at most n edges, the capacity of any cut decreases by at most $\frac{1}{n^2}$ after Step 6; and by the max-flow min-cut theorem, $y'_i \geq y_i - \frac{1}{n^2}$ for each $i \in [g]$ (in Step 7). Furthermore, since edge capacities are integer multiples of $\frac{1}{n^3}$, so are all the flow variables f_i^T corresponding to a maximum r-X_i flow for all $i \in [g]$.

(This latter condition seems mysterious but is needed for technical reasons by the algorithm in [5] which we use in Step 8). Also note that this rounding down does not change the fractional profits much, since

$$\sum_{i=1}^{g} \widetilde{\phi}_i \cdot y'_i \geq \sum_{i=1}^{g} \widetilde{\phi}_i \cdot y_i - \frac{1}{n^2}\sum_{i=1}^{g} \widetilde{\phi}_i \geq \frac{3}{4}\cdot \mathsf{Opt} - \frac{1}{n^2}\sum_{i=1}^{g} \phi_i \geq \frac{3}{4}\cdot \mathsf{Opt} - \frac{\mathsf{Opt}}{n} \geq \frac{\mathsf{Opt}}{2} \tag{4.5}$$

where the second last inequality follows from $\frac{1}{n}\sum_{i=1}^{g}\phi_i \leq \mathsf{Opt}$ (by the preprocessing). By the properties of the algorithm of Charikar et al. [5], in Step (8) we get a subtree A with $d(A)/p(A) \leq 2\alpha\frac{B}{\mathsf{Opt}}$. We finish by handling the two possible cases (Steps 9 and 10):

- If $d(A) \leq \alpha B$, then $p(S') = p(S) + p(A) \geq \frac{\mathsf{Opt}}{2\alpha B}\cdot d(S) + \frac{\mathsf{Opt}}{2\alpha B}\cdot d(A) = \frac{\mathsf{Opt}}{2\alpha B}\cdot d(S')$.
- If $d(A) > \alpha B$, then by averaging, we have $p(A') \geq \frac{\alpha B}{2\cdot d(A)}p(A) \geq \frac{\mathsf{Opt}}{4}$; so $p(S') \geq \frac{\mathsf{Opt}}{4}$.

In either case the subgraph S' satisfies inequality (4.4), and since $S \leftarrow S'$ at the end of the iteration, the invariant holds for the subgraph S as well. ■

This shows that we have a deterministic algorithm for GSO that outputs a tour whose length violates the budget B by a factor of $\alpha = O(\log^2 n)$, but which achieves a constant fraction of the best profit possible. In the next section, we will use it to get an algorithm for a version of the min-latency group Steiner problem.

4.2 Partial Latency Group Steiner

In the *partial latency group Steiner* (LPGST) problem, we are given 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 $h \leq g$. A group $i \in [g]$ is said to be *covered* (or visited) by r-tour τ if any vertex in X_i is visited, and the *arrival time* of such a group i is the length of the shortest prefix of τ that contains an X_i-vertex. The arrival times of all uncovered groups are set to be the tour-length. The weighted sum of arrival times of all groups is termed *latency* of the tour, i.e.,

$$\text{latency}(\tau) = \textstyle\sum_{i \text{ covered}} w_i \cdot \text{arrival time}_\tau(X_i) + \sum_{i \text{ uncovered}} w_i \cdot \text{length}(\tau). \tag{4.6}$$

The objective in LPGST is to compute a minimum latency r-tour that covers at least h groups. (Note that setting the target h to g would give us the standard latency group Steiner problem.) In this section, we present an $(O(\log^2 n), 4)$-bicriteria approximation algorithm for LPGST, i.e., the solution tour visits at least $\frac{h}{4}$ groups and has latency at most $O(\log^2 n)$ times the optimal latency of a tour that visits h groups.

In the following discussion, fix any optimal tour ζ^* for the given instance of LPGST: let Lat^* denote the latency and Opt^* the length of ζ^*. Let $\beta := \frac{5}{4}$, and $\rho = O(\log^2 n)$ be such that the algorithm of Theorem 11 is a $(4,\rho)$ bicriteria approximation for GSO. Algorithm 4 is the approximation algorithm for LPGST.

Claim 12 *The tour τ in Step 8 has length $\Theta(\rho)\cdot \mathsf{Opt}^*$ and latency $O(\rho)\cdot \mathsf{Lat}^*$.*

Proof: The length of each $\tau^{(i)}$ is at most $\rho \cdot \beta^i$ (by the guarantees of Theorem 11), and hence the length of τ is at most $\rho\sum_{i\leq l}\beta^i = O(\rho)\beta^{l-1} = O(\rho)\mathsf{Opt}^*$. Moreover, the increase in Step 8 ensures that $d(\tau) \geq \rho \cdot \mathsf{Opt}^*$. (This increase of the tour length is important, since the contribution of any uncovered group to the latency of the tour τ is the length of τ. Hence, if the original length of τ was much smaller than $\rho \cdot \mathsf{Opt}^*$, we want to ensure that uncovered groups still pay $\rho \cdot \mathsf{Opt}^*$ in the following analysis.)

The proof for bounding the latency is identical to the analysis for the *minimum latency/traveling repairman problem* [6, 11]. We include it here for completeness. Recall optimal solution ζ^* to the LPGST instance, where $d(\zeta^*) = \mathsf{Opt}^* \in (\beta^{l-1}, \beta^l]$. For each $i \in [l]$, let N_i^* denote the total weight of groups visited in ζ^* by time β^i; note that N_l^* equals the total weight of the groups covered by ζ^*.

Algorithm 4 Algorithm for LPGST

1: **guess** an integer l such that $\beta^{l-1} < \mathsf{length}^* \leq \beta^l$.
2: **mark** all groups as *uncovered*.
3: **for** $i = 1 \ldots l$ **do**
4: **run** the GSO algorithm (Theorem 11) on the instance with groups $\{X_i\}_{i=1}^g$, root r, length bound β^{i+1}, and profits:
$$p_i := \begin{cases} 0 & \text{for each covered group } i \in [g] \\ w_i & \text{for each uncovered group } i \in [g] \end{cases}$$
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 the r-tours found in the above iterations. If $d(\tau) < \rho \cdot \beta^l$ then then increase its length to $\rho \cdot \beta^l$ (this may only increase the latencies of groups).
9: **run** the GSO algorithm on the instance with groups $\{X_i\}_{i=1}^g$, root r, length bound β^l, and *unit profit* for each group. Let σ denote the resulting r-tour.
10: **output** tour $\pi := \tau \circ \sigma$ as solution to the LPGST instance.

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. The latency of tour τ is upper bounded by $T := \frac{\rho}{\beta-1} \sum_{i=0}^l \beta^{i+2} \cdot (W - N_i)$. Note also that the latency of optimal tour ζ^* is $\mathsf{Lat}^* \geq \frac{\beta-1}{\beta} \sum_{i=0}^l \beta^i (W - N_i^*)$.

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 β^i length prefix of tour ζ^* corresponds to a feasible solution to this GSO instance. Theorem 11 implies that the profit obtained in $\tau^{(i)}$, i.e. $N_i - N_{i-1} \geq \frac{1}{4} \cdot (N_i^* - N_{i-1})$, i.e. $W - N_i \leq \frac{3}{4} \cdot (W - N_{i-1}) + \frac{1}{4} \cdot (W - N_i^*)$. Using this,

$$\begin{aligned}
(\beta - 1)\frac{T}{\rho} &= \sum_{i=0}^{l} \beta^{i+2} \cdot (W - N_i) \\
&\leq \beta^2 \cdot W + \frac{1}{4} \sum_{i=1}^{l} \beta^{i+2} (W - N_i^*) + \frac{3}{4} \sum_{i=1}^{l} \beta^{i+2} (W - N_{i-1}) \\
&\leq O(1) \cdot \mathsf{Lat}^* + \frac{3\beta}{4} \sum_{i=0}^{l-1} \beta^{i+2} (W - N_i) \\
&\leq O(1) \cdot \mathsf{Lat}^* + \frac{3\beta}{4} \cdot (\beta - 1) \frac{T}{\rho}
\end{aligned}$$

Since $\beta = \frac{5}{4}$, this implies $T = O(\rho) \cdot \mathsf{Lat}^*$, giving the claim. ■

Claim 13 *The tour σ in Step 9 covers at least $\frac{h}{4}$ groups and has length $O(\rho) \cdot \mathsf{Opt}^*$.*

Proof: Since we know that the optimal tour ζ^* has length at most β^l and covers at least h groups, it is a feasible solution to the GSO instance defined in Step 9. Hence the guarantees on the GSO algorithm $\mathscr{A}$ from Theorem 11 ensure that the tour σ has length at most $\rho \beta^l = O(\rho) \mathsf{Opt}^*$. Moreover, the fact that we set unit profits for each group ensure that the solution returned visits at least $h/4$ groups. ■

Theorem 14 *Tour $\pi = \tau \cdot \sigma$ covers at least $\frac{h}{4}$ groups and has latency $O(\rho) \cdot \mathsf{Lat}^*$.*

Proof: Since π visits all the vertices in σ (and maybe some more), Claim 13 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 obtained in Step 8—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 12 implies that the latency of tour τ, $\sum_{i=1}^{g} w_i \cdot \alpha_i = O(\rho) \cdot \mathsf{Lat}^*$. 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 and under $\pi = \tau \cdot \sigma$ are both $\Theta(\rho) \cdot \mathsf{Opt}^*$: since length of π is $O(1)$ times length of τ (here we use the increase in length of τ from Claim 12). 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 12 completes the proof. ■

Proof of Theorem 4: By Theorem 14 above, the tour π covers $h/4$ groups, and has latency $O(\rho) \cdot \mathsf{Lat}^*$. But $\rho = O(\log^2 n)$ by Theorem 11, which completes the proof of Theorem 4. ■

Remark: The above arguments also give an $O(\log^2 n)$-approximation algorithm for the *minimum latency group Steiner* problem, which is the same as the LPGST problem when the target $h = g$. Note that the objective is to minimize the sum of weighted arrival times where *every* group has to be visited. Indeed, the algorithm is simpler, since for this problem we can just repeat Step 4 until *all* groups are covered, instead of stopping after l iterations. A proof identical to that in Claim 12 implies an $O(\log^2 n)$ approximation. We claim this is the best possible unless we improve the known approximation for the standard group Steiner problem, which is a long-standing open problem. Indeed, it is shown in [28] that an α-approximation to latency group Steiner implies an $O(\alpha \cdot \log g)$-approximation to group Steiner tree; thus improving this $O(\log^2 n)$-approximation for latency group Steiner would also improve the best known bound for the standard group Steiner tree problem.

5 Optimal Decision Tree Problem

In the *optimal decision tree problem* [24, 1, 3], we are given a set of m diseases with associated probabilities $\{p_i\}_{i=1}^m$ that sum to 1, and a collection $\{T_j\}_{j=1}^n$ of n binary tests with non-negative costs $\{c_j\}_{j=1}^n$. Each test $j \in [n]$ is a subset T_j of the diseases that correspond to passing the test; so performing test j distinguishes between diseases T_j and $[m] \setminus T_j$.

Definition 4 *A test strategy S is a binary tree where each internal node is labeled by a test, and each leaf node is labeled by a disease such that:*

- *For each disease $i \in [m]$ define a path π_i in S from the root node to some leaf as follows. At any internal node, if i passes the test then π_i follows the right branch; if it fails the test then π_i follows the left branch.*
- *For each $i \in [m]$, the path π_i ends at a leaf labeled disease i.*

The cost *L_i of a disease $i \in [m]$ is the sum of test-costs along path π_i; and the cost of the test strategy S is $\sum_{i=1}^m p_i \cdot L_i$. The objective in the optimal decision tree problem is to compute a test strategy of minimum cost.*

Observe that the optimal decision tree problem is a special case of Isolation: given an instance of optimal decision tree (as above), 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 is $\{j \in [n] \mid i \in T_j\}$. It is easy to see that this Isolation instance corresponds exactly to the optimal decision tree instance. Based on this, it suffices to focus on Isolation in weighted star-metrics, and we obtain an improved bound as follows.

Theorem 15 *Group Steiner orienteering on weighted star metrics has a $(1-\frac{1}{e})$ approximation algorithm.*

Proof: Assume, without loss of generality, that the root r is the center of the star. Consider an instance of group Steiner orienteering (GSO) on weighted star-metric (V,d) with center r and leaves $[n]$, g groups $\{X_i \subseteq [n]\}_{i=1}^{g}$ with profits $\{v_i\}_{i=1}^{g}$, and length bound B. If for each $j \in [n]$, define set $S_j := \{i \in [g] \mid j \in X_i\}$ 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 $\sum_{j\in K} c_j \le B/2$ that maximizes $f(K) := \sum\{v_i \mid i \in \cup_{j\in K} S_j\}$. But the latter problem is an instance of monotone submodular function maximization over a knapsack constraint ($\sum_{j\in K} c_j \le B/2$), for which a $1-\frac{1}{e}$ approximation algorithm is known [32]. ■

Since the proofs in Section 4.2 use the results for GSO as a black-box, we also get

Corollary 16 *There is an $\big(O(1),O(1)\big)$ bicriteria approximation for* LPGST *on weighted star metrics.*

Setting $\rho_{\mathsf{LPGST}} = O(1)$ (from Corollary 16) in the analysis of Section 3.2, we obtain an $O(\rho_{\mathsf{LPGST}} \cdot \log m) = O(\log m)$ approximation for Isolation on weighted star-metrics.

Theorem 17 *There is an $O(\log m)$-approximation algorithm for the optimal decision tree problem.*

Multiway tests. Chakravarthy et al. [3] considered the optimal decision tree problem when the outcomes of tests are multiway (not just binary), and gave an $O(\log m)$-approximation under unit probabilities and costs. We observe that our algorithm can be easily extended to this problem with non-uniform probabilities and costs. 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 $[m]$, and performing test j determines which of the parts the realized disease lies in. Firstly note that this problem is also a special case of Isolation. 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 distance from each other. The demand scenarios are defined naturally: for each $i \in [m]$, scenario i is $\{(j,k) \mid i \in T_j^k\}$. Clearly this Isolation instance is equivalent to the (multiway) decision tree instance. Since the resulting metric is still a weighted star (we only made vertex copies), Corollary 16 and the algorithm of Section 3.2 imply an $O(\log m)$-approximation for this multiway decision tree problem.

6 Adaptive Traveling Repairman

We now extend the results on AdapTSP and Isolation to the adaptive traveling repairman problem (which is the latency version of adaptive TSP). The input to AdapTRP is the same as AdapTSP and Isolation: a metric (V,d) with root $r \in V$, a distribution $\mathscr{D}$ supported on m distinct subsets $\{S_i\}_{i=1}^{m}$ with respective probabilities $\{p_i\}_{i=1}^{m}$.

Definition 5 (The AdapTRP **Problem)** *Given metric (V,d), root r and demand distribution $\mathscr{D}$, the goal in* AdapTRP *is to compute a decision tree T in metric (V,d) with the added conditions that:*

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

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

The main difference while solving the latency problem is that we can *not* just isolate the realized scenario and then visit the vertices of the realized scenario. This is because all the unvisited vertices of the scenario have to pay the entire duration of isolation, and a large group isolated very late would incur a high latency objective. For example, consider an instance of AdapTRP (and Isolation) defined 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 Isolation solution clearly does not visit vertex v (it appears in all scenarios, hence provides no information). So if we first follow this strategy for Isolation, the arrival time for v is at least n (under scenario S_0); since S_0 occurs with $1 - o(1)$ probability, the resulting expected latency is $\Omega(n)$. However, the strategy that first visits v, and then vertices $\{u_1, \cdots, u_n\}$ has expected latency $O(\sqrt{n})$. On the other hand, it is easy to see that one can not ignore the Isolation aspect of the AdapTRP either.

Hence, in general we need to interleave the two goals of isolating a scenario and visiting more vertices to prevent the latency from blowing up. Like in the AdapTSP's isolation algorithm, the idea is to recursively create sub-instances of smaller size to isolate the realized scenario, while also visiting other vertices. For the recursion step, we again use a partitioning algorithm like in the isolation problem; however it is more involved in this case since we need to simultaneously isolate scenarios and visit high-probability vertices. We have two classes of vertices: S_H are those which occur in more than half the scenarios, and S_L are those which occur in fewer than half scenarios. The partitioning scheme is presented in Algorithm 5.

Algorithm 5 PartnLat($\langle M, \{q_i\}_{i \in M} \rangle$)

1: **define** $F_v := \{i \in M \mid v \in S_i\}$ for each $v \in V$.

2: **let** $S := \bigcup_{i \in M} S_i \subseteq V$, $S_L := \{u \in S \mid |F_u| \le \frac{|M|}{2}\}$, and $S_H := S \setminus S_L$.

3: **create** an instance of the *latency group Steiner* problem with the following groups and weights: for each scenario S_i (with $i \in M$),
- The *main* group of scenario i, group X_i has weight $|S_i \cap S_L| p_i$ and vertices $(S_L \cap S_i) \cup (S_H \setminus S_i)$.
- For each $v \in S_i \cap S_H$, group Y_i^v has weight p_i and vertices $\{v\} \cup (S_L \cap S_i) \cup (S_H \setminus S_i)$.

4: **run** the latency group Steiner algorithm (Subsection 4.2) on metric (V, d) with root r and groups as defined above. Note that the goal here is to cover *all* the groups.
  let $\tau' := \langle r, v_1, v_2, \cdots, v_{t'}, r \rangle$ be the r-tour returned.

5: **define** a partition of M as follows:
- scenario $i \in M$ is said to be *covered* by v_k if v_k is the first X_i-vertex on τ'.
- let v_{t-1} be the first vertex on τ' after which *fewer* than half the scenarios are uncovered.
- for $1 \le k \le t-1$, part P_k comprises of all scenarios covered by v_k; and the last part P_t consists of all uncovered scenarios after v_{t-1}.

6: **return** tour $\tau = r, v_1, v_2, \cdots, v_{t-1}, r$ and the partition $\{P_k\}_{k=1}^t$.

Given this partitioning scheme, Algorithm 6 for adaptive TRP is very similar to the one for isolation in AdapTSP— run the partition process and traverse that tour till *(i)* we visit a vertex in S_L which appears in the current instantiation, *(ii)* we visit a vertex in S_H which does not appear, or *(iii)* we have "thrown away" half the scenarios. In each case, we can be sure that more than half the scenarios have been discarded, and we can then return to the root and recurse (while also getting rid of vertices already visited from the scenarios).

6.1 The Analysis

The analysis for this algorithm is similar to that for the isolation problem for TSP (Section 3.2) and we follow the same outline. For any sub-instance $\mathscr{J}$ of AdapTRP, let $\mathsf{Opt}(\mathscr{J})$ denote its optimal value. Just as in the isolation case (Claim 8), it can be easily seen that the latency objective function is also sub-additive.

Algorithm 6 $\mathsf{AdapTRP}\langle M, \{q_i\}_{i\in M}\rangle$

If $|M| = 1$, visit the vertices in this scenario in (approximately) latency minimizing fashion, and quit.
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.
let $q'_j := \sum_{i\in P_k} q_i$ **for all** $j \in 1 \ldots t$.
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, set $k^* \leftarrow t$.
Update the scenarios in P_{k^*} by removing vertices we have already visited in τ until v_{k^*}.
run $\mathsf{AdapTRP}\langle P_{k^*}, \{\frac{q_i}{q'_{k^*}}\}_{i\in P_{k^*}}\rangle$ to recursively cover the realized scenario within P_{k^*}.

Claim 18 *For any sub-instance $\langle M, \{q_i\}_{i\in M}\rangle$ and any partition $\{P_k\}_{k=1}^t$ of M,*

$$\textstyle\sum_{k=1}^t q'_k \cdot \mathsf{Opt}(\langle P_k, \{\frac{q_i}{q'_k}\}_{i\in P_k}\rangle) \le \mathsf{Opt}(\langle M, \{q_i\}_{i\in M}\rangle), \tag{6.7}$$

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 latency cost of the group Steiner instance considered in Step 4 of Algorithm 5 is not too high. In what follows, let $\mathscr{G}$ denote the latency group Steiner instance considered.

Lemma 19 *For any instance $\mathscr{J} = \langle M, \{q_i\}_{i\in M}\rangle$ of* $\mathsf{AdapTRP}$, *the optimal value of the latency group Steiner instance considered in Step 4 of Algorithm* $\mathsf{PartnLat}(\mathscr{J})$ *is at most* $\mathsf{Opt}(\mathscr{J})$.

Proof: Let T be an optimal decision tree for the given $\mathsf{AdapTRP}$ instance $\mathscr{J}$. 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 $v \in S_L$, and moves to the *yes* child if $v \in S_H$. We claim that this tour is a feasible solution to $\mathscr{G}$, the latency group Steiner instance created.

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 to the scenario branched off from σ. If $v \in S_L$, then by the way we defined σ, it must be that v belongs to $S_i \cap S_L$, because otherwise the tree path to the scenario would not branch off σ at v. On the other hand, if $v \in S_H$, then it must be that $v \in S_H \setminus S_i$ (again from the way σ was defined). In either case, $v \in (S_i \cap S_L) \cup (S_H \setminus S_i)$, and visiting v covers *all* groups (i.e. $X_i, \{Y_i^v \mid v \in S_i \cap S_H\}$) associated with scenario i. Thus σ covers all groups of 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 $\mathsf{AdapTRP}$, σ must visit every vertex in S_a. Therefore σ covers all the groups associated with scenario a: clearly $\{Y_a^v \mid v \in S_a \cap S_H\}$ are covered; X_a is also covered unless $S_a \cap S_L = \emptyset$ (however in that case group X_a has zero weight).

We now bound the latency cost of tour σ for instance $\mathscr{G}$. 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 $v \in S_i \cap S_H$) the coverage time for group Y_i^v. The next claim shows that the latency of σ for instance $\mathscr{G}$ is at most $\mathsf{Opt}(\mathscr{J})$.

Claim 20 *The expected cost of the optimal solution for the* $\mathsf{AdapTRP}$ *instance $\mathscr{J}$ is at least $\sum_{i\in M} p_i \cdot |S_L \cap S_i| \cdot \alpha_i + \sum_{i\in M} p_i \cdot \beta_i^v$, which is the average latency of tour σ for the latency group Steiner instance $\mathscr{G}$.*

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). Since all but the last vertex in σ_i are from $(S_L \setminus S_i) \cup (S_H \cap S_i)$, by definition of

σ, the path traced in the decision-tree T when scenario i is realized agrees with σ_i. Hence under scenario S_i, the total arrival time (i.e. latency) for vertices $S_L \cap S_i$ is at least $|S_L \cap S_i| \cdot \alpha_i$. So the contribution of S_i towards the expected total latency in $\mathsf{Opt}(\mathscr{J})$ is at least $p_i \cdot |S_L \cap S_i| \cdot \alpha_i$.

Now consider some vertex $v \in S_i \cap S_H$; let σ_i^v denote the shortest prefix of σ containing an Y_i^v-vertex (note σ_i^v has length β_i^v). Again by the definition of σ, when scenario i is realized, vertex v is not visited before tracing path σ_i^v in the decision tree T. So the contribution of v (under scenario i) to expected total latency is at least $p_i \cdot \beta_i^v$. ■

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

It remains to bound the expected additional latency incurred in Step 4 of Algorithm 6 when a random scenario is realized.

Lemma 21 *At the end of Step 4 of* $\mathsf{AdapTRP}\langle M, \{q_i\}_{i\in M}\rangle$*, the realized scenario lies in* P_{k^*} *(as determined in Step 4). The expected additional latency incurred in this step is at most* $O(\log^2 n) \cdot \mathsf{Opt}(\langle M, \{q_i\}_{i\in M}\rangle)$.

Proof: Consider a particular scenario $i \in M$. If this is the scenario that has realized, the algorithm would traverse tour τ until it hits the vertex v_k such that $i \in P_{k^*}$. By the definition of the partition algorithm, this is the first point in τ where we visit a vertex in $(S_i \cap S_L) \cup (S_H \setminus S_i)$. We show below that the expected additional latency incurred by vertices in S_i in Step 4 of Algorithm 6 is at most twice the total (weighted average) latency of groups $X_i, \{Y_i^v \mid v \in S_i \cap S_H\}$ in solution τ to the latency group Steiner instance $\mathscr{G}$. To see this, observe that any vertex visited along the way before v_k would incur only the length till it was visited (correspondingly, the group Y_i^v would incur exactly the same latency in tour the solution for instance $\mathscr{G}$). All other vertices incur a latency of the entire prefix until v_k: this is exactly the latency incurred by the groups X_i and Y_i^v (for unvisited vertices v) in the tour σ for the latency group Steiner instance $\mathscr{G}$. Furthermore, the unvisited vertices would incur an additional latency in this step, since we go back to the root before recursing. Therefore, the total expected latency accrued in this step is at most twice the latency of τ w.r.t the group Steiner instance $\mathscr{G}$. The proof then follows from Lemma 19, and because the latency group Steiner problem admits an $O(\log^2 n)$ approximation algorithm (see Subsection 4.2). ■

Since in each level of the recursion we halve the number of possible scenarios, the depth of this is at most $\log |M|$. This combined with Lemma 21 and Claim 18, gives us the following result (by an identical proof as in Theorem 9).

Theorem 22 *There is an* $O(\log^2 n \cdot \log m)$*-approximation algorithm for* $\mathsf{AdapTRP}$.

There is still a logarithmic gap between our approximation bound, and the $\Omega(\log^{1-\varepsilon} n)$ hardness of approximation [28], for AdapTRP on tree-metrics. Closing this gap is an interesting open question.

References

[1] Micah Adler and Brent Heeringa. Approximating Optimal Binary Decision Trees. In *Proceedings of the 11th International Workshop on Approximation, Randomization and Combinatorial Optimization*, pages 1–9, 2008.

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

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

[4] V. T. Chakravarthy, V. Pandit, S. Roy, P. Awasthi, and M. Mohania. Decision Trees for Entity Identification: Approximation Algorithms and Hardness Results. In *Proceedings of the ACM-SIGMOD Symposium on Principles of Database Systems*, 2007.

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

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

[7] Shuchi Chawla and Tim Roughgarden. Single-source stochastic routing. In *Approximation, randomization and combinatorial optimization*, volume 4110 of *Lecture Notes in Comput. Sci.*, pages 82–94. Springer, Berlin, 2006.

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

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

[10] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:208–217, 2004.

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

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

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

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

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

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

[17] Sudipto Guha, Kamesh Munagala, and Saswati Sarkar. Information acquisition and exploitation in multichannel wireless networks. *CoRR*, abs/0804.1724, 2008.

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

[19] Anupam Gupta, Mohammad T. Hajiaghayi, and Harald Räcke. Oblivious network design. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 970–979, New York, NY, USA, 2006. ACM.

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

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

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

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

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

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

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

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

[28] Viswanath Nagarajan. *Approximation Algorithms for Sequencing Problems*. PhD thesis, Tepper School of Business, Carnegie Mellon University, 2009.

[29] Robert D. Nowak. The geometry of generalized binary search, 2009. arXiv.org:0910.4397.

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

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

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

A Hardness of Approximation for AdapTSP

We show that AdapTSP is at least as hard to approximate as group Steiner tree.

Theorem 23 *An α-approximation algorithm for* AdapTSP *implies an $\alpha + o(1)$ approximation algorithm for group Steiner tree. Hence* AdapTSP *is $\Omega(\log^{2-\varepsilon} n)$ hard to approximate even on tree metrics.*

Proof: This reduction is almost identical to the reduction [4] 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

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

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 $\mathsf{Opt} \le \mathsf{Opt}' \le \mathsf{Opt} + 1$ which would prove the theorem.

(A) $\mathsf{Opt} \le \mathsf{Opt}'$**.** Consider the optimal strategy for the AdapTSP instance; let σ denote the r-tour traversed by this strategy under scenario S_{g+1}. We now argue that σ is a feasible solution to the group Steiner tree instance. 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 strategy under scenario S_i is also σ, since the strategy 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 $S_i \supseteq X_i$.

(B) $\mathsf{Opt}' \le \mathsf{Opt} + 1$**.** Let τ denote an optimal r-tour for the given GST instance, so $d(\tau) = \mathsf{Opt}$. Consider the following strategy 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 strategy 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 strategy. We have $d(\pi_{g+1}) \le d(\tau) \le \mathsf{Opt}$, and $d(\pi_i) \le 2n \cdot \max_{u,v} d(u,v) \le L$ for all $i \in [g]$. Thus the resulting AdapTSP objective is at most:

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

Thus we have the desired reduction. ■