Approximation Algorithms for Discounted Graph Search with Norm Objectives

Svenja M. Griesbach
Department of Computer Science, RWTH Aachen University, Germany

Felix Hommelsheim
Department of Computer Science, University of Cologne, Germany

Max Klimm
Institute of Mathematics, Technische Universität Berlin, Germany

Abstract

We introduce a unified framework for classical search and routing problems, including pathwise search, expanding search, the minimum spanning tree problem, and the traveling salesperson problem. The framework is based on two parameters. The first is a discount factor $\alpha \in [0, 1]$: the first traversal of an edge incurs its full cost, whereas each subsequent traversal incurs only an α-fraction of this cost. For a path starting at a designated root vertex, the α-latency of a vertex is the discounted cost accumulated until the vertex is first visited. The second parameter is a norm parameter $p \geq 1$. The objective is to find a root-starting path that visits all vertices and minimizes the p-norm of the resulting vector of α-latencies.

The model interpolates between several well-studied objectives. For $p = 1$ and $\alpha = 1$, it recovers pathwise search; for $p = 1$ and $\alpha = 0$, it recovers expanding search. As p tends to infinity, the objective converges to a makespan-type criterion. At the endpoints $\alpha = 1$ and $\alpha = 0$, this limiting objective corresponds to TSP-type and MST-type behavior, respectively. For $p = 1$, we give polynomial-time constant-factor approximation algorithms for all $\alpha \in [0, 1]$, matching the best known guarantees for expanding search at $\alpha = 0$ and pathwise search at $\alpha = 1$. For general $p \geq 1$, we obtain a randomized constant-factor approximation algorithm and a derandomized pseudo-polynomial-time algorithm with the same guarantee.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation Algorithm, Expanding Search, Pathwise Search, Search Problem, Graph Exploration, Traveling Repairperson Problem, Minimum Latency Problem

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2026.27

Category APPROX

Related Version *Full Version*: `https://arxiv.org/abs/2607.11301`

Funding The work of the first author is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) by the grant Ho 3831/9-1 (project ID: 514505843). The work of the second author is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) by the grant HO 7562/2-1 (project ID: 573939419). The work of the third author is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany´s Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/1, EXC-2046/2, project ID: 390685689).

Acknowledgements We thank Kevin Schewior for fruitful discussions.

1 Introduction

Pathwise search, expanding search, the traveling salesperson problem, and the minimum spanning tree problem are four fundamental models for exploring or connecting a network. At first glance, these problems optimize rather different objectives: pathwise and expanding

search minimize sums of discovery times, while the traveling salesperson problem and the minimum spanning tree problem minimize the time until all vertices are reached or connected. In this paper, we study a common framework that unifies these problems through two parameters. The first parameter determines how repeated traversals of edges are charged, and the second parameter determines how the individual vertex latencies are aggregated.

We are given an undirected graph $G = (V, E)$ with non-negative edge cost $c_e \in \mathbb{N}$, and a designated start vertex s. A solution is a traversal of the graph, i.e., a path starting in s that may visit edges more than once and eventually visits all vertices. For a vertex v, its latency is the time at which v is visited for the first time. Classical search problems ask for a traversal that minimizes the sum of these latencies.

The *pathwise search problem* asks for such a traversal when every traversal of an edge e requires c_e time units. Thus, the latency of a vertex is equal to the total cost of the prefix of the path until the vertex is first visited, and the goal is to minimize the sum of the latencies of all vertices. The problem, also known as the *traveling repairperson problem*, captures situations in which all traversals of an edge take the same amount of time. Hence, it has been used as a model for the movement of a repairperson in a road network or for disk heads on a hard drive. The problem is NP-hard even on weighted trees (Sitters [27]), so much research has focused on approximation algorithms. An algorithm is a ϱ-approximation algorithm if for any instance it runs in polynomial time and the cost of the solution output by the algorithm is at most $\varrho \cdot \textsc{Opt}$, where Opt denotes the cost of an optimum solution to the respective instance. The factor ϱ is called the approximation ratio or guarantee. The best currently known approximation algorithm for general graphs with unit weights for all vertices yields a 3.59-approximation (Chaudhuri et al. [15]).

The *expanding search problem* asks for a sequence of edges with the property that every prefix of edges is a connected subgraph and, without loss of generality, a tree containing s. The latency of a vertex is the total cost of the edges added until the vertex first appears in the sequence, and the goal is again to minimize the sum of the latencies of all vertices. It captures situations in which a tree network needs to be installed, and the cost of an edge is interpreted as the time needed to establish a connection between its end vertices. Hence, it has been used as a model for clearing paths in an area devastated by disasters or for mining. The problem is NP-hard (Averbakh and Pereira [9]) and the best currently known approximation algorithm for general graphs yields a 5.44-approximation (Griesbach et al. [22]).

We propose and study a general model of graph exploration that we term the *discounted graph search problem*. As in pathwise search, a solution is a traversal π starting in s that may visit edges more than once and eventually visits all vertices. The first parameter of the model is a discount factor $\alpha \in [0, 1]$. While the first traversal of an edge e in the sequence requires c_e time units, every further traversal of the same edge requires only $\alpha \cdot c_e$ time units. We call the cost of a path where repeated traversals are discounted in this way the *α-cost* of the path. Given a traversal π, the *α-latency* $C_{\alpha,v}(\pi)$ of a vertex v is defined as the α-cost of the smallest prefix of π that visits v.

The second parameter is a norm parameter $p \geq 1$, which determines how the individual α-latencies are aggregated. For a traversal π, its p-norm α-latency is

$$\|\mathbf{C}_\alpha(\pi)\|_p = \left(\sum_{v \in V} C_{\alpha,v}(\pi)^p \right)^{1/p}.$$

The goal is to compute a traversal π minimizing this quantity.

This two-parameter model contains several classical problems as special cases or limiting cases. For $p = 1$ and $\alpha = 1$, every traversal of an edge is charged its full cost, and the objective is the sum of the first-visit times of all vertices. Thus, we recover the pathwise search problem. For $p = 1$ and $\alpha = 0$, repeated traversals of already used edges are free. Therefore, only the cost of newly added edges contributes to the discovery time of vertices, and we recover the expanding search problem.

The parameter p interpolates between sum-of-latencies objectives and makespan-type objectives. As $p \to \infty$, the p-norm objective converges to the maximum α-latency of any vertex. For $\alpha = 1$, this limiting objective asks for a shortest traversal starting at s that visits all vertices, since the maximum latency is exactly the time when the last vertex is first reached. This is the path version of the traveling salesperson problem. For $\alpha = 0$, the maximum α-latency is the total cost of the distinct edges that have been introduced by the time all vertices are reached. Minimizing this quantity is therefore equivalent to finding a minimum-cost connected subgraph spanning all vertices, and hence to the minimum spanning tree problem. Intermediate values of $\alpha \in (0, 1)$ capture situations where the first traversal of an edge is more time-consuming than later traversals, for example, because of additional delays due to pathfinding or clearing a road, while further traversals still require a non-negligible amount of time. Intermediate values of p capture settings in which one wants to balance the average discovery time of vertices with the time until the last vertices are reached. The parameter p can also be interpreted from a fairness perspective. For $p = 1$, the objective minimizes the total latency and thus corresponds to a social-welfare objective, whereas larger values of p put increasing emphasis on vertices with large latency; in the limit $p \to \infty$, the objective becomes an egalitarian objective that minimizes the worst latency. The special case of $p = 2$ and $\alpha = 1$ has been studied as the *traveling firefighter problem* in the literature [16]. The authors motivate this particular choice of objective by firefighters looking for an order in which to tackle wildfires whose damage grows quadratically with the elapsed time.

1.1 Our Results

We first note that the discounted graph search problem is computationally hard already in the case $p = 1$. In particular, the hardness of pathwise and expanding search carries over to the endpoints $\alpha = 1$ and $\alpha = 0$, and we also show that the problem remains hard for every fixed intermediate value $\alpha \in (0, 1)$. The proof can be found in the full version of the paper.

▶ **Theorem 1.** *For $p = 1$ and every constant $\alpha \in [0, 1]$, there exists a constant $\varepsilon > 0$ such that there is no polynomial-time $(1 + \varepsilon)$-approximation algorithm for the discounted graph search problem with discount factor α, unless* P = NP.

We therefore focus on approximation algorithms. We first consider the case $p = 1$, where the objective is the total α-latency. For this case, we obtain a polynomial-time approximation algorithm with approximation ratio

$$\varrho_1(\alpha) \coloneqq \begin{cases} 2\mathrm{e} & \text{if } \alpha = 0, \\ 2\frac{\alpha}{(1+\alpha)W(\alpha/\mathrm{e})} & \text{otherwise,} \end{cases}$$

where $W : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is the Lambert-W function, which assigns $x \geq 0$ the unique value $w \geq 0$ such that $w\mathrm{e}^w = x$. This gives the following result.

▶ **Theorem 2.** *For $p = 1$ there is a polynomial-time $\varrho_1(\alpha)$-approximation algorithm for the discounted graph search problem.*

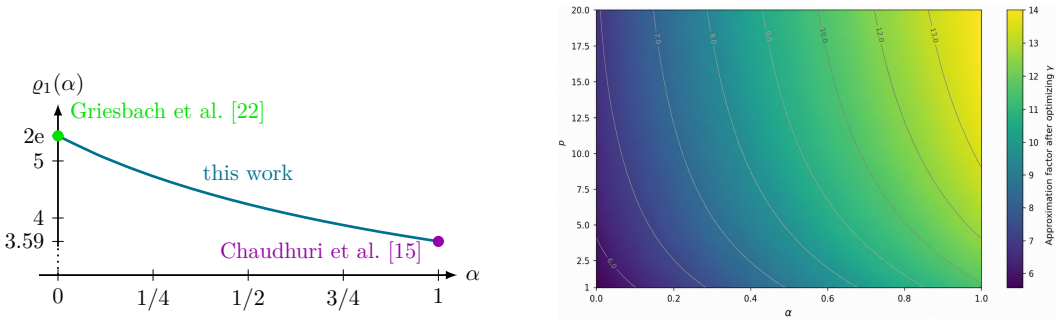

Figure 1 Approximation ratios $\varrho(\alpha)$ obtained in this work: (a) for $p = 1$ as a function of α; (b) as a function of $p > 1$ and α.

For $\alpha = 0$, the approximation factor is 2e, which matches the best currently known approximation guarantee for the expanding search problem (Griesbach et al. [22]). For $\alpha = 1$, the approximation factor is 3.59, which matches the best currently known approximation guarantee for the pathwise search problem in general graphs with unit vertex weights (Chaudhuri et al. [15]). For arbitrary values of $\alpha \in [0, 1]$, the approximation ratio interpolates smoothly between these two values; see Figure 1.

Like the algorithms of Chaudhuri et al. [15] and Griesbach et al. [22], our algorithm is based on a sequence of certain trees of exponentially growing cost. The trees that we consider are related to the concept of *good* trees used by Chaudhuri et al. [15] in their 3.59-approximation for the pathwise search problem. Roughly speaking, a tree is called *good* if it contains a certain number of vertices and its cost is bounded by the cost of an appropriate path visiting the same number of vertices. We extend this concept to general values of α. Specifically, we define good k-trees for α and show that such trees can be computed in polynomial time for the values of k required by our algorithm. For $\alpha = 1$, this recovers the trees used by Chaudhuri et al. [15]. For $\alpha = 0$, the good k-trees correspond to k-MSTs, and the resulting bound matches the best known approximation guarantee for expanding search.

We then turn to the more general case $p \geq 1$, where the objective is the p-norm of the vertices' α-latencies. For $p > 1$, the approach for the case $p = 1$ cannot be applied directly. The main difficulty is that the objective is no longer linear in the latencies, and hence the auxiliary-graph construction used for $p = 1$ no longer directly captures the contribution of each additional tree. Moreover, the good-tree machinery from the $p = 1$ case does not provide suitable trees for every value of k.

We therefore use a randomized construction that is closer in spirit to the geometric tree sequences used for $p = 1$, but relies only on 2-approximate k-MSTs. For every $k \in [n]$, we compute such a tree, choose a random geometric offset, and concatenate trees whose costs grow geometrically. Analyzing the p-th moments of the resulting vertex latencies gives the following guarantee.

▶ **Theorem 3.** *For every $p \geq 1$, there is a polynomial-time randomized algorithm for the discounted graph search problem with approximation factor*

$$\varrho_p(\alpha) = \min_{\gamma > 1} \left\{ \frac{2(1+\alpha)}{\gamma - 1} \left[\frac{\gamma^{2p} - \gamma^p}{p \ln \gamma} \right]^{1/p} \right\}.$$

The approximation factor from Theorem 3 is depicted in Figure 1. For fixed α, the optimized factor tends to $8(1 + \alpha)$ as $p \to \infty$. Over the range $\alpha \in [0, 1]$, the smallest value is attained at $\alpha = 0$ and $p = 1$, where the factor is 2e, while the largest value is approached for $\alpha = 1$ and $p \to \infty$, where the factor tends to 16.

For the traveling firefighter problem, corresponding to the cases $\alpha = 1$ and $p = 2$, this yields an approximation factor of ≈ 11.28. This is worse by a factor of 2 than the approximation factor of ≈ 5.65 claimed by Farhadi et al. [16]. However, as we discuss in Section 1.2, we do not believe that this discrepancy can be resolved straightforwardly.

Finally, we derandomize the p-norm algorithm using an auxiliary graph, as in the case $p = 1$. The difference is that for $p > 1$ the state must also record the accumulated cost of the trees traversed so far, since this quantity determines the nonlinear contribution to the p-norm objective. This yields a two-dimensional, time-expanded auxiliary graph and the following deterministic guarantee.

▶ **Theorem 4.** *For every $p \geq 1$, there is a deterministic pseudo-polynomial-time algorithm for the discounted graph search problem with approximation factor $\varrho_p(\alpha)$ defined in Theorem 3. Moreover, if the tree costs are integral and polynomially bounded, then the algorithm runs in polynomial time.*

1.2 Related Work

The pathwise search problem is also known as the *traveling repairperson problem*. This problem was shown to be NP-hard on general graphs by Sahni and Gonzalez [26], and to be NP-hard even on weighted trees by Sitters [27]. In the weighted setting, each vertex has a non-negative weight and the task is to minimize the weighted sum of latencies. The problem can be solved efficiently on unweighted trees and paths [1, 17, 25]. The first approximation algorithm for this problem was devised by Blum et al. [12] who gave a 144-approximation for general graphs and an 8-approximation for weighted trees. Goemans and Kleinberg [20] improved the approximation ratio for general graphs to 21.55 and the one for weighted trees to 3.59. In particular, they show that a β-approximation for the rooted k-MST problem of finding a shortest tree containing a given root and $k-1$ further vertices implies an approximation ratio of 3.59β for the traveling repairperson problem. Thus, the improved k-MST approximations of $\beta = 3$ by Garg [18], of $\beta = 2.5$ by Arya and Ramesh [8], of $\beta = 2 + \varepsilon$ by Arora and Karakostas [7], and of $\beta = 2$ by Garg [19] immediately yield better approximations of 10.77, 8.98, $7.18 + \varepsilon$, and 7.18, respectively. In addition, the work of Arora [5] implies an improved approximation ratio of $3.59 + \varepsilon$ for the Euclidean case. Archer et al. [4] showed that the approximation ratio of $7.18 + \varepsilon$ can be obtained with fewer calls to the k-MST subroutine. Chaudhuri et al. [15] improved the approximation ratio for general graphs to match the ratio of 3.59 for weighted trees. Archer and Blasiak [3] further improved the approximation ratio for weighted trees to 3.03. Arora and Karakostas [6] gave a quasi-polynomial-time approximation scheme (QPTAS) both for weighted trees and for Euclidean instances. Sitters [28] gave a polynomial-time approximation scheme (PTAS) for Euclidean instances, weighted trees, and planar graphs.

The expanding search problem has been shown to be NP-hard by Averbakh and Pereira [9]. Alpern and Lidbetter [2] proposed an algorithm for the expanding search problem on weighted trees. The first approximation algorithm was devised by Hermans et al. [23] who gave an 8-approximation. The approximation ratio was improved by Griesbach et al. [22] to $2\mathrm{e} \approx 5.44$.

Farhadi et al. [16] study the L_p-TSP, where the objective is to minimize the p-norm of the vector of visit times; this problem interpolates between the traveling repairperson problem for $p = 1$ and the path variant of TSP for $p \to \infty$ and corresponds to our case of $\alpha = 1$. They claimed to have a universal 8-approximation that is valid for all p simultaneously. If correct, this would improve over a 16-approximation by Golovin et al. [21]. Farhadi et al. [16] further claimed a 5.65-approximation for $p = 2$. However, we believe that both their results have a

substantial gap. Their algorithms rely on the ability to compute good k-trees for all values $k \in [n]$; Lemma 3.1 in their paper cites Chaudhuri et al. [15] for the claim that this can be done in polynomial time. However, this is not what Chaudhuri et al. [15] show. They only show "how to obtain such k-trees for some values of k" and "that it actually suffices to use the few k-trees that we found" [15, p. 3]. The algorithms of Farhadi et al. [16], on the other hand, crucially require good k-trees for *all* values of k to be computable in polynomial time. It seems to be an open problem whether this can be done. A positive answer to this open problem appears to require new techniques, as Garg [19, p. 1] writes that "we cannot argue that the cost of the tree picked is at most the cost of the best k-stroll. As was shown in [15] such an argument can be made for certain values of k by exploiting the same slack in the Goemans-Williamson argument". Our algorithms for $p > 1$ avoid this issue by computing 2-approximate k-MSTs for all values of k instead of good k-trees; the former can be done with the algorithm of Garg [19]. This is exactly why we incur an additional loss of a factor of 2. The discussion of Garg cited above makes it plausible that this additional factor of 2 cannot be avoided with current techniques. This route was also taken by Golovin et al. [21], who use 2-approximate k-MSTs for all values of k instead of good k-trees. It is interesting to note that our approximation guarantees for any $p \geq 1$ and $\alpha \in [0, 1]$ are strictly below 16 and approach this value when $\alpha = 1$ and $p \to \infty$.

Norm-based interpolation also appears in ordered optimization problems such as ordered k-median, which interpolates between k-median and k-center [13]. In scheduling and load balancing, one often minimizes the L_p-norm of the machine-load vector, which similarly interpolates between average-load objectives and makespan minimization; see, e.g., the work of Awerbuch et al. [10] and subsequent work on all-norm and minimum-norm load balancing [11, 14, 24]. While these problems differ from ours, they illustrate the role of p-norm objectives as a natural way to interpolate between sum-type and bottleneck-type criteria.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with $|V| = n + 1$ and a designated start vertex s and let $\alpha \in [0, 1]$ be the discount factor. Each edge $e \in E$ has a non-negative cost $c_e \in \mathbb{N}$. We assume that $c_e > 0$ for every edge incident to s since otherwise these edges can be contracted. We call a sequence of edges $\pi = (e_1, \ldots, e_k)$ a *path* if for all $1 \leq i < k$ the end vertex of e_i and the start vertex of e_{i+1} coincide and the sequence starts in s. If the path also ends in s, we call it a *tour*. The set of all tours is denoted by Π. For two paths $\pi = (e_1, \ldots, e_k)$ and $\pi' = (e'_1, \ldots, e'_l)$ where the end vertex of π coincides with the start vertex of π', we write $\pi + \pi' = (e_1, \ldots, e_k, e'_1, \ldots, e'_l)$ for the concatenation of those two paths.

For a sequence of edges π, we define the *α-cost* $c_\alpha(\pi)$ of π as the sum of all edges in π, where for the second and further traversals of an edge e its cost c_e is multiplied by α. We call a sequence π a *tree* if the set of edges in π spans a tree in G that contains s. Since a tree contains no edge more than once, the α-cost of a tree is independent of α, and we simply write $c(\pi)$. The following lemma states the simple fact that every tree can be transformed into a tour by increasing the α-cost by at most a factor of $1 + \alpha$. We state it here for future reference; its proof can be found in the full version of the paper.

▶ **Proposition 5.** *For a tree T, there exists a tour $\pi(T)$ that visits all vertices in T and has α-cost at most $c_\alpha(\pi(T)) \leq (1 + \alpha)c(T)$.*

For a vertex $v \in V$ visited by a tour $\pi \in \Pi$, we denote by π_v the prefix of π until v is visited for the first time. The *α-latency* $C_{\alpha,v}(\pi)$ of vertex v in π is defined as $C_{\alpha,v}(\pi) := c_\alpha(\pi_v)$ where we set $C_{\alpha,v}(\pi) = \infty$ if v is not visited by π at all. Note that $C_{\alpha,s}(\pi) = 0$. The *total*

α-*latency* $C_\alpha(\pi)$ of π is then defined as $C_\alpha(\pi) \coloneqq \sum_{v \in V} C_{\alpha,v}(\pi)$. In particular, we have $C_\alpha(\pi) = \infty$ if there is a vertex $v \in V$ that is never visited by π. The set of feasible solutions for the *discounted graph search problem* is the set Π. Further, a tour $\pi^* \in \Pi$ is optimal for the discounted graph search problem with $\alpha \in [0,1]$ if $C_\alpha(\pi^*) \leq C_\alpha(\pi)$ for all tours $\pi \in \Pi$.

A k-*tree*, k-*path*, or k-*tour* is a tree, path, or tour, respectively, that contains s and visits at least $k+1$ distinct vertices of V. For a k-tour $\pi \in \Pi$, we denote by π_k the prefix of π until the k-th distinct vertex of $V \setminus \{s\}$ is visited for the first time. A k-path π^* is *optimal* with respect to α if $c_\alpha(\pi^*) \leq c_\alpha(\pi)$ for all k-paths π. If α is clear from context, we simply say that π^* is optimal. Further, a k-tree T is called *good* if its cost is at most $c(T) \leq \frac{2}{1+\alpha} c_\alpha(\pi^*)$, where π^* is an optimal k-path.

3 Algorithms for the 1-Norm of α-Latencies

Recall that $\varrho(\alpha) = 2\frac{\alpha}{(1+\alpha)W(\alpha/\mathrm{e})}$ when $\alpha \in (0,1]$ and $\varrho(0) = 2\mathrm{e}$. We prove the following result.

▶ **Theorem 2.** *For $p = 1$ there is a polynomial-time $\varrho_1(\alpha)$-approximation algorithm for the discounted graph search problem.*

The intuitive idea behind the algorithm is to concatenate a specific subset of *good trees*. We construct an initial set of good trees in Section 3.1 using an algorithm by Chaudhuri et al. [15]. They introduced this algorithm for the computation of good k-trees for a specific subset of values of k and in the special case of $\alpha = 1$. A more thorough analysis of their approach yields that the obtained trees are indeed good trees for arbitrary choices of $\alpha \in [0,1]$. Since the algorithm does not return good k-trees for all values of $k \in [n]_0$, we use so-called phantom trees for the remaining values of k; the details of this construction can be found in the full version of the paper. Next, we introduce the concatenating algorithm in Section 3.2, which takes as input a set of good trees and returns a feasible solution to the discounted graph search problem. More precisely, it constructs an auxiliary graph H whose vertices correspond to the good trees. Within this auxiliary graph, the algorithm computes a shortest path and concatenates the corresponding trees to the desired tour. The algorithm is thus well-defined only if the input contains only real trees. We temporarily ignore this constraint and show in Section 3.3 that, under this relaxation, the returned (phantom) tour has an approximation guarantee of $\varrho_1(\alpha)$. The analysis builds on the construction of a randomized path in the auxiliary graph that visits exponentially growing trees. Then, we show that the same guarantee can be obtained by a path that visits only real trees. This allows us to restrict the input to good real trees while maintaining the approximation guarantee. Combining these results yields a well-defined algorithm that returns the desired solution. The running-time analysis then concludes the proof of Theorem 2.

3.1 Finding Good Trees

To obtain good k-trees, we use an algorithm by Chaudhuri et al. [15]. They study the pathwise search problem, i.e., the discounted graph search problem with $p = \alpha = 1$, for which they introduce a primal–dual algorithm that computes good k-trees for a subset of values of k. With a more thorough analysis, we can show that these trees are indeed good trees for all choices of $\alpha \in [0,1]$.

A subtle issue is that the primal–dual procedure is not monotone in the vertex-budget parameter. In contrast to what is sometimes assumed in related Lagrangian approaches for k-MST [7], increasing the parameter need not produce a larger tree, even before the final pruning step. This is because the parameter can change the order in which moats merge and become inactive. An explicit example can be found in the full version of the paper.

A detailed explanation of how the framework of Chaudhuri et al. [15] needs to be adapted to compute good k-trees for all values of $k \in [n]$ can be found in the full version of the paper. For this purpose, it is important to distinguish between values of k for which a good k-tree can be computed and those for which this is not the case. In the latter case, we define a phantom tree, which is not an actual tree but a placeholder whose costs are interpolated from the costs of the closest two real k-trees. We later argue that these phantom trees are not used by our approximation algorithm, so their inclusion as placeholders does not introduce any issues down the line.

The main result needed here is the following lemma; its proof can be found in the full version of the paper.

▶ **Lemma 6.** *There is a polynomial-time algorithm that computes a set $\mathcal{T}$ containing, for each value $k \in [n]_0$, a real or phantom tree $T_k \in \mathcal{T}$. Each such T_k is a good k-tree.*

3.2 The Concatenating Algorithm

We introduce the *concatenating algorithm*, which takes as input a set of good k-trees, selects a subset of these, and concatenates them to obtain a solution for the discounted graph search problem. To this end, let $\mathcal{I}$ be a set of good k-trees for a subset of values $k \in [n]_0$ and let $I \subseteq [n]_0$ be such that $k \in I$ if and only if $\mathcal{I}$ contains a good k-tree denoted by T_k. Then the concatenating algorithm with input $\mathcal{I}$ is defined as follows:

1. Construct a directed auxiliary graph $H = (V_H, A_H)$ with vertices $V_H = I$, edges $A_H = \{(i,j) : i < j\}$, and edge lengths $\ell_{i,j} = \big[(1+\alpha)n - \alpha j - i\big] c(T_j)$.
2. Compute a shortest 0–n-path $P^* = (n_0, n_1, \dots, n_l)$ with $n_0 = 0$ and $n_l = n$ in H.
3. Start with the empty sequence $\pi_{\text{ALG}} = (\cdot)$.
   For each phase $j = 1, \dots, l$, construct the tour $\pi(T_{n_j})$ according to Proposition 5. Let π_{f} and π_{b} be the tour obtained from traversing the sequence $\pi(T_{n_j})$ in forward or backward direction, respectively. Let $V_j \coloneqq V_{n_j} \setminus \big(\bigcup_{i=1}^{j-1} V_{n_i}\big)$ be the set of vertices that are contained in T_{n_j} but in no previous tree T_{n_i} for $1 \le i < j$. Let $\pi_{\min} \in \{\pi_{\text{f}}, \pi_{\text{b}}\}$ be such that $\sum_{v \in V_j} C_{\alpha,v}(\pi_{\text{ALG}} + \pi_{\min})$ is minimized. Set $\pi_{\text{ALG}} \leftarrow \pi_{\text{ALG}} + \pi_{\min}$.

Note that the concatenating algorithm is only well-defined under the following two assumptions. First, the input must contain a good 0-tree and a good n-tree. Second, all trees $T_{n_0}, \dots, T_{n_l}$ on the shortest path must be real trees, because otherwise Step 3 is not well defined. The second constraint is trivially fulfilled if the input $\mathcal{I}$ only contains real trees.

3.3 The Concatenating Algorithm with Phantom Trees

We want to apply the concatenating algorithm to the set $\mathcal{T}$ of good trees whose existence is guaranteed by Lemma 6. However, this set may contain phantom trees, so the concatenating algorithm is not necessarily well-defined on this input. We ignore this issue for now by assuming that all phantom trees are real trees and can therefore be transformed into tours. Under this assumption, we analyze the approximation ratio of the resulting tour π_{ALG} and then show how to obtain the same guarantee when restricting to real trees.

To this end, let $\mathcal{T}$ be the set of good k-trees whose existence is guaranteed by Lemma 6. Note that $\mathcal{T}$ contains a good (real or phantom) k-tree *for all* values of $k \in [n]_0$ and that the trees T_0 and T_n are real trees. Before applying the concatenating algorithm to $\mathcal{T}$, we slightly adjust the cost of the phantom trees $\mathcal{P} \subset \mathcal{T}$. More precisely, for a phantom tree $T_k \in \mathcal{P}$, let $k_l < k$ be maximal and $k < k_r$ be minimal such that $T_{k_l}, T_{k_r} \in \mathcal{T}$ are two real trees. We redefine the cost of T_k to

$$c(T_k) \coloneqq (1-\mu)c(T_{k_l}) + \mu\, c(T_{k_r}) \quad \text{with} \quad \mu \coloneqq \frac{k - k_l}{k_r - k_l}. \tag{1}$$

Since the phantom tree T_k was originally constructed by a linear interpolation of two real trees, this redefinition of the cost can only decrease its cost. See also [15] for a more detailed discussion. Hence, the phantom tree T_k remains a good k-tree. We denote the set $\mathcal{T}$ with the adjusted costs of the phantom trees by $\mathcal{T}'$. We analyze the approximation guarantee of the tour π_{ALG} obtained by running the concatenating algorithm on input $\mathcal{T}'$. In this section, we prove the following lemma.

▶ **Lemma 7.** *If all trees in $\mathcal{T}'$ were real trees, the tour π_{ALG} obtained from the concatenating algorithm with input $\mathcal{T}'$ is a feasible solution for the discounted graph search problem with approximation guarantee $\varrho_1(\alpha)$.*

For the remainder of this section, we denote by P^* the shortest 0–n-path in H computed in Step 2 of the concatenating algorithm on input $\mathcal{T}'$. Since each vertex $i \in V_H$ corresponds to a unique tree $T_i \in \mathcal{T}'$, we often refer to the vertices on P^* as the chosen trees. Furthermore, we extend the procedure of Step 3 to an arbitrary 0–n-path $P = (n_0, n_1, \dots, n_{l'})$ in H. More precisely, we introduce a randomized variant of Step 3 that runs as follows. We start with the empty sequence $\pi_P = (\cdot)$. For each phase $j = 1, \dots, l'$, we construct the tour $\pi(T_{n_j})$ according to Proposition 5. Then we pick a traversal direction (forward or backward) of the tour $\pi(T_{n_j})$ uniformly at random and set $\pi_P \leftarrow \pi_P + \pi(T_{n_j})$.

We proceed to analyze the sequence π_{ALG}. First, observe that P^* contains vertex n. Thus, the corresponding good n-tree T_n is traversed and appended to π_{ALG} in Step 3. This ensures that the returned tour π_{ALG} is a feasible solution to the discounted graph search problem with finite total α-latency, i.e., π_{ALG} visits all vertices. It remains to analyze the approximation factor obtained by π_{ALG}. To this end, we first show that for any given 0–n-path P in H, the expected total α-latency $\mathbb{E}[C_\alpha(\pi_P)]$ of π_P is at most the length $\ell(P)$ of P in H. Next, we use a probabilistic argument to prove the existence of a 0–n-path in H whose length is at most $\varrho_1(\alpha)$ times the total α-latency of an optimal tour. Finally, we explain how these two results imply that the concatenating algorithm on input $\mathcal{T}'$ indeed computes a tour with an approximation ratio of $\varrho_1(\alpha)$.

In that direction, we introduce the function ψ_P such that $\psi_P(k)$ gives an upper bound on the latency of the k-th distinct vertex in $V \setminus \{s\}$ visited by the tour constructed from path P in H. More formally, for a fixed 0–n-path $P = (n_0, n_1, \dots, n_l)$ in H, the function $\psi_P : [n]_0 \to \mathbb{R}_{\geq 0}$ is defined by

$$\psi_P(k) \coloneqq \begin{cases} 0 & \text{if } k = 0 \\ c(T_{n_j}) + (1+\alpha)\sum_{i=0}^{j-1} c(T_{n_i}) & \text{if } n_{j-1} < k \leq n_j, \text{for some } j \in \{1, \dots, l\}. \end{cases}$$

This definition is well-defined as $n_0 = 0$ and $n_l = n$. We obtain the following lemma. Its proof can be found in the full version of the paper.

▶ **Lemma 8.** *Let π_P be the randomized tour obtained from a 0–n-path P in H and let $v \in V$ be the k-th distinct vertex of $V \setminus \{s\}$ in π_P for some $k \in [n]_0$. Then we have $\mathbb{E}[C_{\alpha,v}(\pi_P)] \leq \psi_P(k)$.*

For a fixed 0–n-path P in H, Lemma 8 yields $\mathbb{E}[C_\alpha(\pi_P)] = \sum_{v \in V} \mathbb{E}[C_{\alpha,v}(\pi_P)] \leq \sum_{k=0}^{n} \psi_P(k)$. Recall that in Step 3 of the concatenating algorithm, we pick the traversal direction of each subtour such that $\sum_{v \in V_j} C_{\alpha,v}(\pi_{\text{ALG}} + \pi_{\min})$ is minimized, where $V_j \coloneqq V_{n_j} \setminus \left(\bigcup_{i=1}^{j-1} V_{n_i}\right)$. Hence, for the tour π_{ALG} we obtain the deterministic bound

$$C_\alpha(\pi_{\text{ALG}}) \leq \sum_{v \in V} C_{\alpha,v}(\pi_{P^*}) \leq \sum_{k=0}^{n} \psi_{P^*}(k). \tag{2}$$

Next, we use the following identity, which expresses the length of a path P in H in terms of ψ_P. Its proof can be found in the full version of the paper.

▶ **Lemma 9.** *For a 0–n-path P in H we have $\sum_{k=0}^{n} \psi_P(k) = \ell(P)$.*

The following lemma is the technically most challenging part of the analysis, as it compares the length of a 0–n-path P in H against the total α-latency of the optimal solution. The proof uses similar ideas as in [15] and [22]. We consider a randomized sequence of good k-trees with exponentially increasing costs where $\gamma \in (1, \infty)$ is a factor determining the rate of the exponential growth. Then the expected length of that path can be bounded by a constant factor times the optimal total α-latency. A probabilistic argument then yields that this bound is also attained by the shortest 0–n-path P^* in H. Specifically, we obtain the following bound. Its proof can be found in the full version of the paper.

▶ **Lemma 10.** *Let $\pi^* \in \Pi$ be an optimal solution for the discounted graph search problem. Then there exists a 0–n-path P in H such that $\ell(P) \leq 2\frac{\gamma+\alpha}{(1+\alpha)\ln\gamma} C_\alpha(\pi^*)$ for any $\gamma \in (1, \infty)$.*

Lemma 10 yields an approximation ratio parametrized in $\gamma \in (1, \infty)$. To obtain the smallest approximation ratio possible, we optimize over γ. To this end, we denote by $W : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ the Lambert-W function that assigns $x \geq 0$ the unique value $w \geq 0$ such that $we^w = x$. The proof of the following result can be found in the full version of the paper.

▶ **Lemma 11.** *For $\gamma \in (1, \infty)$, let h be defined as $h(\gamma) \coloneqq 2\frac{\gamma+\alpha}{(1+\alpha)\ln(\gamma)}$. Then h is minimized for $\gamma^* = e^{1+W(\alpha/e)}$. In particular, we have $\varrho_1(\alpha) = h(\gamma^*)$.*

Combining all previous results yields the proof for Lemma 7; this proof can be found in the full version of the paper.

We have shown so far that the concatenating algorithm with input $\mathcal{T}'$ (the set of trees guaranteed by Lemma 6 with adjusted costs for phantom trees) computes a solution to the discounted graph search problem with approximation guarantee $\varrho_1(\alpha)$. However, we obtained this result only under the assumption that all trees were real trees, i.e., that they could be transformed into tours. It remains to show that we can obtain the same approximation guarantee when using only real trees as input to the concatenating algorithm. The issue with phantom trees as input arises only if phantom trees lie on the shortest 0–n-path P^* computed by the concatenating algorithm, since these trees are the ones traversed to obtain the final tour π_{ALG}. As a first step towards proving Theorem 2, we thus show that whenever the shortest 0–n-path P^* on input $\mathcal{T}'$ contains a phantom tree, there exists another 0–n-path with the same length that visits only real trees. The proof of the lemma follows the argument of [4, 15] and can be found in the full version of the paper.

▶ **Lemma 12.** *Let H be the auxiliary graph constructed by the concatenating algorithm on input $\mathcal{T}'$. Then there exists a shortest 0–n-path P in H that does not visit any vertex whose corresponding tree is a phantom tree.*

By Lemma 12, applying the concatenating algorithm to the set $\mathcal{R} \subseteq \mathcal{T}$ of real trees in $\mathcal{T}$ ensures that the algorithm is well-defined while maintaining an approximation guarantee of $\varrho_1(\alpha)$. We can now prove Theorem 2; the proof can be found in the full version of the paper.

4 Algorithms for the p-Norm of α-Latencies

We now turn to the case of arbitrary $p \geq 1$. The case $p = 1$ was treated in the previous section, where the linearity of the objective allowed us to use an auxiliary graph whose edge lengths directly represent the additional latency caused by adding a tree. For $p > 1$, this linearity is lost, and the same construction no longer applies directly.

At a high level, we would still like to use sequences of trees of geometrically increasing cost. However, two difficulties arise. First, as pointed out by Garg [19], it is not possible, for every $k \in [n]$, to use the primal–dual algorithm to obtain a k-MST with cost no more than the optimum k-stroll. In particular, this issue occurs for those values of k for which the primal–dual procedure does not directly compute a k-MST. Second, when $p > 1$, the objective is nonlinear in the vertices' α-latencies. Thus, an analogue of the auxiliary-graph cost from the previous section has to keep track not only of how many vertices have already been visited, but also of the accumulated cost of the trees used so far.

We subdivide this section into two parts. First, we present a randomized algorithm based on a geometric scaling argument. Then, we show how to derandomize it using a time-expanded auxiliary graph; the resulting running time is pseudo-polynomial in the edge costs.

4.1 A Randomized Algorithm

To avoid the difficulties with the auxiliary-graph construction for the moment, we first give a randomized algorithm with the desired approximation guarantee.

▶ **Theorem 3.** *For every $p \geq 1$, there is a polynomial-time randomized algorithm for the discounted graph search problem with approximation factor*

$$\varrho_p(\alpha) = \min_{\gamma > 1} \left\{ \frac{2(1+\alpha)}{\gamma - 1} \left[\frac{\gamma^{2p} - \gamma^p}{p \ln \gamma} \right]^{1/p} \right\}.$$

The approximation ratio given in the above theorem is depicted in Section 1.1. It ranges from 2e for $\alpha = 0$ and $p = 1$ to a limiting value of 16 for $\alpha = 1$ as $p \to \infty$. Note that the value 2e matches the bound for $\alpha = 0$ and $p = 1$ from the previous section.

Instead of computing a *good* k-tree, we use Garg's 2-approximation [19] to compute a 2-approximate k-MST for every $k \in [n]$. Recall that such a tree contains s and visits k additional vertices. Let these trees be denoted by $T_1, T_2, \ldots, T_n$.

We next define a randomized sequence of trees, similar to the construction in the previous section. Fix some $\gamma > 1$, which will be optimized later. Let $b = \gamma^U$, where U is drawn uniformly at random from $[0, 1]$. For all relevant values of r, define $n_r = \max\{k \in [n]_0 : c(T_k) \leq 2b\gamma^r\}$. Here, $c(T_k)$ denotes only the total length of the tree and, in particular, does not depend on p.

Let $P = (n_0, n_1, \ldots, n_l)$ be the sequence of values obtained by the above procedure, where $n_0 = 0$. As in the previous section, we concatenate the sequence of trees induced by P to obtain a tour, which we denote by π_P. We also define a function ψ_P such that $\psi_P(k)$ gives an upper bound on the α-latency of the k-th distinct vertex in $V \setminus \{s\}$ visited by π_P. Formally, the function $\psi_P : [n]_0 \to \mathbb{R}_{\geq 0}$ is defined by

$$\psi_P(k) \coloneqq \begin{cases} 0 & \text{if } k = 0, \\ (1+\alpha) \sum_{i=0}^{j} c(T_{n_i}) & \text{if } n_{j-1} < k \leq n_j, \text{for some } j \in \{1, \ldots, l\}. \end{cases}$$

We obtain the following lemma.

▶ **Lemma 13.** *Let π_P be the randomized tour obtained from P, and let $v \in V$ be the k-th distinct vertex of $V \setminus \{s\}$ in π_P, for some $k \in [n]_0$. Then $C_{\alpha,v}(\pi_P) \leq \psi_P(k)$.*

Proof. This follows directly from the definition of ψ_P. In the worst case, all trees are nested and vertex k is visited at the very end of the last tree containing it. Moreover, the cost of the tour induced by tree T_{n_i} is bounded by $(1+\alpha)c(T_{n_i})$. ◀

Lemma 13 and the randomized sequence of trees imply Theorem 3.

Proof of Theorem 3. Let π^* be an optimal tour for the p-norm α-latency objective. Let v_i^* be the i-th vertex visited by π^*. Then

$$\mathrm{OPT} = \left(\sum_{i \in [n]} \left(C_{\alpha, v_i^*}(\pi^*) \right)^p \right)^{1/p}.$$

Fix some arbitrary $i \in [n]$. Let r be such that $C_{\alpha, v_i^*}(\pi^*) = d\gamma^r$ for some $d \in [1, \gamma)$. We consider two cases.

First case: $d \le b$. In this case, there is a tree T_{n_r} with cost at most $2b\gamma^r$ and $n_r \ge i$. Hence,

$$\psi_P(i) \le (1+\alpha) \sum_{k=1}^{r} c(T_{n_k}) \le (1+\alpha) \sum_{k=1}^{r} 2b\gamma^k < (1+\alpha) \sum_{k=-\infty}^{r} 2b\gamma^k = (1+\alpha) 2b\gamma^r \frac{\gamma}{\gamma - 1}.$$

Second case: $d > b$. In this case, the same argument applied to the next threshold gives

$$\psi_P(i) \le (1+\alpha) \sum_{k=1}^{r+1} c(T_{n_k}) \le (1+\alpha) \sum_{k=1}^{r+1} 2b\gamma^k < (1+\alpha) \sum_{k=-\infty}^{r+1} 2b\gamma^k = (1+\alpha) 2b\gamma^{r+1} \frac{\gamma}{\gamma - 1}.$$

Since U is chosen uniformly at random in $[0, 1]$ and $b = \gamma^U$, we can average over the two cases and obtain

$$\mathbb{E}_U[\psi_P(i)^p] \le \int_{\log_\gamma d}^{1} \left((1+\alpha) 2b\gamma^r \frac{\gamma}{\gamma - 1} \right)^p \mathrm{d}U + \int_{0}^{\log_\gamma d} \left((1+\alpha) 2b\gamma^{r+1} \frac{\gamma}{\gamma - 1} \right)^p \mathrm{d}U.$$

We now compute the right-hand side. Let $a \coloneqq \log_\gamma d$. Since $b = \gamma^U$, we have

$$\begin{aligned}
&\int_{\log_\gamma d}^{1} \left((1+\alpha) 2b\gamma^r \frac{\gamma}{\gamma - 1} \right)^p \mathrm{d}U + \int_{0}^{\log_\gamma d} \left((1+\alpha) 2b\gamma^{r+1} \frac{\gamma}{\gamma - 1} \right)^p \mathrm{d}U \\
&= \left((1+\alpha) 2 \frac{\gamma}{\gamma - 1} \right)^p \left[\gamma^{rp} \int_a^1 \gamma^{pU} \, \mathrm{d}U + \gamma^{(r+1)p} \int_0^a \gamma^{pU} \, \mathrm{d}U \right].
\end{aligned}$$

Using

$$\int \gamma^{pU} \, \mathrm{d}U = \frac{\gamma^{pU}}{p \ln \gamma},$$

we obtain

$$\begin{aligned}
&\left((1+\alpha) 2 \frac{\gamma}{\gamma - 1} \right)^p \left[\gamma^{rp} \int_a^1 \gamma^{pU} \, \mathrm{d}U + \gamma^{(r+1)p} \int_0^a \gamma^{pU} \, \mathrm{d}U \right] \\
&= \left((1+\alpha) 2 \frac{\gamma}{\gamma - 1} \right)^p \frac{\gamma^{rp}(\gamma^p - \gamma^{pa}) + \gamma^{(r+1)p}(\gamma^{pa} - 1)}{p \ln \gamma}.
\end{aligned}$$

Since $a = \log_\gamma d$, we have $\gamma^{pa} = d^p$. Hence,

$$\begin{aligned}
\mathbb{E}_U[\psi_P(i)^p] &\le \left((1+\alpha) 2 \frac{\gamma}{\gamma - 1} \right)^p \frac{\gamma^{rp}(\gamma^p - d^p) + \gamma^{(r+1)p}(d^p - 1)}{p \ln \gamma} \\
&= \left((1+\alpha) 2 \frac{\gamma}{\gamma - 1} \right)^p \frac{d^p \gamma^{rp}(\gamma^p - 1)}{p \ln \gamma} \\
&= (1+\alpha)^p \frac{2^p}{p \ln \gamma} \left[\left(\frac{d\gamma^{r+2}}{\gamma - 1} \right)^p - \left(\frac{d\gamma^{r+1}}{\gamma - 1} \right)^p \right].
\end{aligned}$$

The contribution of v_i^* to the optimum is

$$\left(C_{\alpha,v_i^*}(\pi^*)\right)^p = (d\gamma^r)^p.$$

Therefore, the previous bound implies

$$\frac{\mathbb{E}_U[\psi_P(i)^p]}{(d\gamma^r)^p} \leq (1+\alpha)^p \frac{2^p}{p\ln\gamma}\left[\left(\frac{\gamma^2}{\gamma-1}\right)^p - \left(\frac{\gamma}{\gamma-1}\right)^p\right] = (1+\alpha)^p \frac{2^p}{p\ln\gamma}\cdot\frac{\gamma^{2p}-\gamma^p}{(\gamma-1)^p} =: \eta. \tag{3}$$

In particular, since $C_{\alpha,v_i^*}(\pi^*) = d\gamma^r$, we have

$$\mathbb{E}_U[\psi_P(i)^p] \leq \eta \cdot \left(C_{\alpha,v_i^*}(\pi^*)\right)^p. \tag{4}$$

We now use this bound to compare the expected p-norm α-latency of the randomized tour with the optimum. By the definition of ψ_P, for every realization of U we have

$$C_{\alpha,v_i^*}(\pi_P) \leq \psi_P(i) \qquad \text{for all } i \in [n].$$

Hence,

$$\mathbb{E}_U\left[\left(\sum_{i\in[n]} C_{\alpha,v_i^*}(\pi_P)^p\right)^{1/p}\right] \leq \mathbb{E}_U\left[\left(\sum_{i\in[n]} \psi_P(i)^p\right)^{1/p}\right].$$

We next use Jensen's inequality. Recall that if f is concave and X is a nonnegative random variable, then

$$\mathbb{E}[f(X)] \leq f(\mathbb{E}[X]).$$

In our setting, we apply this with

$$f(x) = x^{1/p} \qquad \text{and} \qquad X = \sum_{i\in[n]} \psi_P(i)^p.$$

Since $p \geq 1$, the function $f(x) = x^{1/p}$ is concave on $\mathbb{R}_{\geq 0}$. Moreover, $X \geq 0$ for every realization of the random choice of U. Therefore, Jensen's inequality gives

$$\mathbb{E}_U\left[\left(\sum_{i\in[n]} \psi_P(i)^p\right)^{1/p}\right] \leq \left(\mathbb{E}_U\left[\sum_{i\in[n]} \psi_P(i)^p\right]\right)^{1/p} = \left(\sum_{i\in[n]} \mathbb{E}_U[\psi_P(i)^p]\right)^{1/p}.$$

Using (4), we obtain

$$\begin{aligned}
\mathbb{E}_U\left[\left(\sum_{i\in[n]} C_{\alpha,v_i^*}(\pi_P)^p\right)^{1/p}\right] &\leq \left(\sum_{i\in[n]} \eta\cdot\left(C_{\alpha,v_i^*}(\pi^*)\right)^p\right)^{1/p} \\
&= \eta^{1/p}\left(\sum_{i\in[n]} \left(C_{\alpha,v_i^*}(\pi^*)\right)^p\right)^{1/p} \\
&= \eta^{1/p}\cdot \mathrm{OPT}.
\end{aligned}$$

It remains to simplify $\eta^{1/p}$. By (3),

$$\begin{aligned}\eta^{1/p} &= (1+\alpha)\left[\frac{2^p}{p\ln\gamma}\cdot\frac{\gamma^{2p}-\gamma^p}{(\gamma-1)^p}\right]^{1/p}\\ &= (1+\alpha)\frac{2}{(p\ln\gamma)^{1/p}}\cdot\frac{(\gamma^{2p}-\gamma^p)^{1/p}}{\gamma-1}\\ &= (1+\alpha)\frac{2}{\gamma-1}\left[\frac{\gamma^{2p}-\gamma^p}{p\ln\gamma}\right]^{1/p}.\end{aligned}$$

Thus, for every fixed $\gamma > 1$, the randomized algorithm has expected approximation factor

$$(1+\alpha)\frac{2}{\gamma-1}\left[\frac{\gamma^{2p}-\gamma^p}{p\ln\gamma}\right]^{1/p}.$$

Optimizing over $\gamma > 1$ gives the claimed bound. ◀

4.2 Derandomization via a Time-Expanded Auxiliary Graph

We now describe how to derandomize the randomized algorithm from the previous subsection. The idea is similar to the derandomization used for the case $p = 1$: we construct an auxiliary graph and compare the length of a shortest path in this graph to the expected length of a suitable randomized path.

For $p = 1$, the auxiliary graph only needs to keep track of how many vertices have already been visited. For $p > 1$, however, the objective is nonlinear in the latencies. Thus, the additional cost of traversing a tree depends not only on the number of vertices visited so far, but also on the latency accumulated so far. We therefore use a two-dimensional auxiliary graph. One dimension keeps track of the number of vertices that have already been visited, and the other dimension keeps track of the total cost of the trees traversed so far.

▶ **Theorem 4.** *For every $p \geq 1$, there is a deterministic pseudo-polynomial-time algorithm for the discounted graph search problem with approximation factor $\varrho_p(\alpha)$ defined in Theorem 3. Moreover, if the tree costs are integral and polynomially bounded, then the algorithm runs in polynomial time.*

Let $T_1, \ldots, T_n$ be the 2-approximate k-MST trees used in the previous subsection, and let $c(T_k)$ denote the cost of tree T_k. We set T_0 to be the trivial tree containing only s, with $c(T_0) = 0$. Let

$$Z \coloneqq \left\{\sum_{k\in S} c(T_k) : S \subseteq [n]\right\}.$$

Thus, Z is the set of all values that can occur as the accumulated cost of a set of traversed trees.

We construct a directed auxiliary graph $H = (V_H, A_H)$ as follows: The vertex set is $V_H = \big([n]_0 \times Z\big) \cup \{\omega\}$, where ω is an additional terminal vertex. A vertex (i, z) represents the state in which i vertices of $V \setminus \{s\}$ have already been visited, and the total cost of the trees traversed so far is z. For every $i, j \in [n]_0$ with $i < j$ and every $z, z' \in Z$ with $z' = z + c(T_j)$, we add the directed edge $\big((i, z), (j, z')\big)$. The length of this edge is defined as

$$\ell_{(i,z),(j,z')} \coloneqq (n-i)\left(\big((1+\alpha)z'\big)^p - \big((1+\alpha)z\big)^p\right). \tag{5}$$

Finally, for every $z \in Z$, we add an edge $((n, z), \omega)$ of length zero. The algorithm computes a shortest path from $(0, 0)$ to ω in H. Let $P^* = \big((n_0, z_0), (n_1, z_1), \ldots, (n_l, z_l), \omega\big)$ be such a shortest path, where $n_0 = 0$, $z_0 = 0$, and $n_l = n$. The algorithm then concatenates the trees $T_{n_1}, T_{n_2}, \ldots, T_{n_l}$ in this order, using the tree-to-tour conversion from Proposition 5. We denote the resulting tour by π_{ALG}. We first relate the length of a path in H to the p-norm α-latency of the corresponding concatenated tour.

▶ **Lemma 14.** *Let $P = \big((n_0, z_0), (n_1, z_1), \ldots, (n_l, z_l), \omega\big)$ be any $(0, 0)$–ω path in H, where $n_0 = 0$, $z_0 = 0$, and $n_l = n$. Let π_P be the tour obtained by concatenating the trees $T_{n_1}, \ldots, T_{n_l}$. Then $\big(\sum_{v \in V} C_{\alpha,v}(\pi_P)^p\big)^{1/p} \leq \ell(P)^{1/p}$.*

Proof. For $h \in \{1, \ldots, l\}$, the transition from (n_{h-1}, z_{h-1}) to (n_h, z_h) corresponds to traversing tree T_{n_h}, and by construction

$$z_h = z_{h-1} + c(T_{n_h}).$$

After this traversal, the accumulated cost of the traversed trees is z_h. By the tree-to-tour conversion, the α-latency of every vertex that is first covered by T_{n_h} is at most

$$(1 + \alpha) z_h.$$

Hence, if $n_{h-1} < k \leq n_h$, then the k-th distinct vertex visited by π_P has α-latency at most $(1 + \alpha) z_h$.

Define

$$\psi_P(k) \coloneqq (1 + \alpha) z_h \qquad \text{if } n_{h-1} < k \leq n_h.$$

Then

$$C_{\alpha, v_k}(\pi_P) \leq \psi_P(k)$$

for the k-th distinct vertex v_k of $V \setminus \{s\}$ visited by π_P. Therefore,

$$\sum_{v \in V} C_{\alpha,v}(\pi_P)^p \leq \sum_{k=1}^{n} \psi_P(k)^p.$$

It remains to observe that the right-hand side is exactly the length of P. Indeed, using the definition of the edge lengths in (5), we get

$$\begin{aligned}
\ell(P) &= \sum_{h=1}^{l} (n - n_{h-1}) \left(\big((1 + \alpha) z_h\big)^p - \big((1 + \alpha) z_{h-1}\big)^p \right) \\
&= \sum_{h=1}^{l} (n_h - n_{h-1}) \big((1 + \alpha) z_h\big)^p \\
&= \sum_{k=1}^{n} \psi_P(k)^p.
\end{aligned}$$

The second equality follows by telescoping. Thus,

$$\sum_{v \in V} C_{\alpha,v}(\pi_P)^p \leq \ell(P),$$

which proves the claim. ◀

We next show that the auxiliary graph contains a path whose length is bounded by the same expression that appeared in the randomized analysis. For fixed $\gamma > 1$, define

$$\eta_\gamma \coloneqq (1+\alpha)^p \frac{2^p}{p \ln \gamma} \cdot \frac{\gamma^{2p} - \gamma^p}{(\gamma - 1)^p}.$$

▶ **Lemma 15.** *Let π^* be an optimal tour for the p-norm α-latency objective. Then, for every fixed $\gamma > 1$, there exists a $(0,0)$–ω path P in H such that*

$$\ell(P) \le \eta_\gamma \sum_{i \in [n]} \big(C_{\alpha, v_i^*}(\pi^*)\big)^p.$$

Proof. We use the randomized construction from the previous subsection. For a random choice of $U \in [0,1]$, let $P(U)$ be the corresponding sequence of trees. After removing consecutive repetitions, this sequence defines a path in the auxiliary graph H. Indeed, the first coordinate records the number of vertices covered, while the second coordinate records the accumulated cost of the selected trees.

By the definition of the auxiliary graph and the calculation in the proof of Lemma 14, we have

$$\ell(P(U)) = \sum_{i \in [n]} \psi_{P(U)}(i)^p.$$

From the randomized analysis, for every $i \in [n]$ we have

$$\mathbb{E}_U[\psi_{P(U)}(i)^p] \le \eta_\gamma \big(C_{\alpha, v_i^*}(\pi^*)\big)^p.$$

Taking the sum over all $i \in [n]$ and using linearity of expectation gives

$$\mathbb{E}_U[\ell(P(U))] = \mathbb{E}_U\left[\sum_{i \in [n]} \psi_{P(U)}(i)^p\right] = \sum_{i \in [n]} \mathbb{E}_U[\psi_{P(U)}(i)^p] \le \eta_\gamma \sum_{i \in [n]} \big(C_{\alpha, v_i^*}(\pi^*)\big)^p.$$

Therefore, there exists at least one realization of U such that

$$\ell(P(U)) \le \eta_\gamma \sum_{i \in [n]} \big(C_{\alpha, v_i^*}(\pi^*)\big)^p.$$

This proves the lemma. ◀

Using the above lemmas and analyzing the size of the time-expanded auxiliary graph, we prove Theorem 4.

Proof of Theorem 4. Let P^* be a shortest $(0,0)$–ω path in H, and let π_{ALG} be the tour obtained by concatenating the trees along P^*. By Lemma 14,

$$\left(\sum_{v \in V} C_{\alpha, v}(\pi_{\mathrm{ALG}})^p\right)^{1/p} \le \ell(P^*)^{1/p}.$$

Since P^* is a shortest path, for every $\gamma > 1$ and for the path P whose existence is guaranteed by Lemma 15, we have

$$\ell(P^*) \le \ell(P).$$

Thus,

$$\ell(P^*) \le \eta_\gamma \sum_{i\in[n]} \left(C_{\alpha,v_i^*}(\pi^*)\right)^p = \eta_\gamma \cdot \mathrm{OPT}^p.$$

Taking p-th roots gives

$$\left(\sum_{v\in V} C_{\alpha,v}(\pi_{\mathrm{ALG}})^p\right)^{1/p} \le \eta_\gamma^{1/p} \cdot \mathrm{OPT}.$$

Finally,

$$\eta_\gamma^{1/p} = (1+\alpha)\frac{2}{\gamma-1}\left[\frac{\gamma^{2p}-\gamma^p}{p\ln\gamma}\right]^{1/p}.$$

Since this holds for every $\gamma > 1$, we may minimize over $\gamma > 1$, which gives the claimed approximation factor.

It remains to discuss the running time. Let

$$N_H := |V_H| + |A_H|$$

denote the size of the auxiliary graph. All edge lengths in H are nonnegative, and hence a shortest $(0,0)$–ω path can be computed with Dijkstra's algorithm in time

$$O(N_H \log N_H).$$

In general, the set Z may be exponentially large, since it consists of all subset sums of the tree costs. Thus, the algorithm is pseudo-polynomial. If the tree costs are integral and polynomially bounded, then $|Z|$ is polynomially bounded as well. In this case, N_H is polynomially bounded, and the algorithm runs in polynomial time. ◀

5 Discussion

We studied the discounted graph search problem, a two-parameter framework that contains pathwise search, expanding search, TSP-type objectives, and MST-type objectives as special or limiting cases. The first parameter, $\alpha \in [0,1]$, determines how repeated traversals of edges are charged, while the second parameter, $p \ge 1$, determines how the individual vertex latencies are aggregated. For $p = 1$, we obtain a polynomial-time approximation algorithm whose guarantee interpolates between the currently best-known approximation ratios for expanding search and pathwise search. For general $p \ge 1$, we obtain a randomized constant-factor approximation and a deterministic pseudo-polynomial-time algorithm with the same approximation guarantee.

We stated and analyzed the problem for a uniform discount factor α. The results for $p = 1$ also extend to the case where every edge e has its own discount factor $\alpha_e \in [0,1]$. In this setting, the first traversal of edge e costs c_e, while every further traversal costs $\alpha_e c_e$. Let $\alpha_{\min} := \min_{e\in E} \alpha_e$ and $\alpha_{\max} := \max_{e\in E} \alpha_e$. Then the analysis yields the approximation guarantee

$$\varrho_1(\alpha) := \begin{cases} 2\mathrm{e} & \text{if } \alpha_{\max} = \alpha_{\min} = 0, \\ 2\dfrac{\alpha_{\max}}{(1+\alpha_{\min})W(\alpha_{\max}/\mathrm{e})} & \text{otherwise.} \end{cases}$$

The reason is that the analysis uses the largest possible discount factor when comparing the cost of trees to paths, while the conversion of trees into tours benefits from the smallest discount factor. Thus, the same proof goes through with α replaced by $\alpha_{\max}$ in the former part of the analysis and by $\alpha_{\min}$ in the latter.

A similar generalization is possible when the cost of an edge depends on how often it has already been used. Suppose that for each edge e we are given a sequence $c_e^{(1)}, c_e^{(2)}, c_e^{(3)}, \ldots$ such that the k-th traversal of edge e costs $c_e^{(k)}$. If these sequences are non-increasing, i.e., $c_e^{(k)} \geq c_e^{(k+1)}$ for all $e \in E$ and $k \in \mathbb{N}$, then our analysis can again be extended. Define $\alpha_{\min} := \min_{e \in E} c_e^{(2)}/c_e^{(1)}$ and $\alpha_{\max} := \max_{e \in E} c_e^{(2)}/c_e^{(1)}$. Then the same approximation guarantee as above follows. Indeed, in the comparison to the optimum, it is sufficient to consider paths in which each edge is used at most twice. On the other hand, the algorithm may use edges more often, and for non-increasing traversal costs, every traversal after the second one can only become cheaper. Thus, replacing all costs $c_e^{(k)}$ for $k \geq 2$ by $c_e^{(2)}$ can only make the algorithm more expensive, while preserving the relevant comparison to the optimum.

By contrast, for arbitrary non-decreasing traversal-cost sequences, one cannot hope for a constant-factor approximation in full generality. For example, suppose that $c_e^{(1)} = 0$ and $c_e^{(k)} = 1$ for all $k \geq 2$ for every edge $e \in E$. Then there is a traversal with total latency zero if and only if the graph has a Hamiltonian path starting at s. Otherwise, every feasible traversal must repeat some edge and hence has positive latency. Since deciding the existence of such a Hamiltonian path is NP-complete, no constant-factor approximation can exist in this setting unless P = NP. It would be interesting to understand which structural assumptions on the sequences $c_e^{(1)}, c_e^{(2)}, c_e^{(3)}, \ldots$ still allow constant-factor approximations.

Finally, for the case $p > 1$, one can also compare our bounds to algorithms for the limiting case $p = \infty$. For any nonnegative latency vector $x \in \mathbb{R}^n_{\geq 0}$, we have $\|x\|_\infty \leq \|x\|_p \leq n^{1/p}\|x\|_\infty$. Thus, any β-approximation for the maximum-latency objective immediately gives a $\beta n^{1/p}$-approximation for the p-norm objective. At the endpoint $\alpha = 1$, the maximum-latency objective corresponds to a TSP-path-type problem, while at the endpoint $\alpha = 0$ it corresponds to the MST problem. Therefore, for large values of p, one can use approximation algorithms for these limiting problems to obtain alternative approximation guarantees. In particular, once $p = \Omega(\log n)$, the factor $n^{1/p}$ becomes constant. For even larger values of p, this factor approaches 1, and the guarantee approaches that of the corresponding maximum-latency approximation algorithm. This gives better bounds for sufficiently large p in regimes where the limiting TSP- or MST-type objective can be approximated more accurately than the general bound obtained from the randomized geometric construction.

References

1 Foto N. Afrati, Stavros S. Cosmadakis, Christos H. Papadimitriou, George Papageorgiou, and Nadia Papakostantinou. The complexity of the travelling repairman problem. *RAIRO – Theoretical Informatics and Applications*, 20(1):79–87, 1986. doi:10.1051/ita/1986200100791.

2 Steve Alpern and Thomas Lidbetter. Mining coal or finding terrorists: The expanding search paradigm. *Operations Research*, 61(2):265–279, 2013. doi:10.1287/opre.1120.1134.

3 Aaron Archer and Anna Blasiak. Improved approximation algorithms for the minimum latency problem via prize-collecting strolls. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 429–447, 2010. doi:10.1137/1.9781611973075.36.

4 Aaron Archer, Asaf Levin, and David P. Williamson. A faster, better approximation algorithm for the minimum latency problem. *SIAM Journal on Computing*, 37(5):1472–1498, 2008. doi:10.1137/07068151X.

5 Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

6 Sanjeev Arora and George Karakostas. Approximation schemes for minimum latency problems. *SIAM Journal on Computing*, 32(5):1317–1337, 2003. doi:10.1137/S009753970139965.

7 Sanjeev Arora and George Karakostas. A $2+\varepsilon$ approximation algorithm for the k-MST problem. *Mathematical Programming*, 107(3):491–504, 2006. doi:10.1007/S10107-005-0693-1.

8 Sunil Arya and H. Ramesh. A 2.5-factor approximation algorithm for the k-MST problem. *Information Processing Letters*, 65:117–118, 1998. doi:10.1016/S0020-0190(98)00010-6.

9 Igor Averbakh and Jordi Pereira. The flowtime network construction problem. *IIE Transactions*, 44(8):681–694, 2012. doi:10.1080/0740817X.2011.636792.

10 Baruch Awerbuch, Yossi Azar, Edward F Grove, Ming-Yang Kao, P Krishnan, and Jeffrey Scott Vitter. Load balancing in the L_p norm. In *Proceedings of IEEE 36th Symposium on Foundations of Computer Science*, pages 383–391. IEEE, 1995. doi:10.1109/SFCS.1995.492494.

11 Yossi Azar, Leah Epstein, Yossi Richter, and Gerhard J Woeginger. All-norm approximation algorithms. *Journal of Algorithms*, 52(2):120–133, 2004. doi:10.1016/j.jalgor.2004.02.003.

12 Avrim Blum, Prasad Chalasani, Don Coppersmith, William R. Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–171, 1994. doi:10.1145/195058.195125.

13 Deeparnab Chakrabarty and Chaitanya Swamy. Interpolating between k-median and k-center: Approximation algorithms for ordered k-median. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 29:1–29:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.29.

14 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*, pages 126–137, 2019. doi:10.1145/3313276.3316322.

15 Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, trees, and minimum latency tours. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 36–45, 2003. doi:10.1109/SFCS.2003.1238179.

16 Majid Farhadi, Alejandro Toriello, and Prasad Tetali. The traveling firefighter problem. In *Proceedings of the SIAM Conference on Applied and Computational Discrete Algorithms (ACDA)*, pages 205–216. SIAM, 2021. doi:10.1137/1.9781611976830.19.

17 Alfredo García, Pedro Jodrá, and Javier Tejel. A note on the travelling repairman problem. *Networks*, 40:27–31, 2002. doi:10.1002/net.10031.

18 Naveen Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 302–309, 1996. doi:10.1109/SFCS.1996.548489.

19 Naveen Garg. Saving an epsilon: A 2-approximation for the k-MST problem in graphs. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, pages 396–402, 2005. doi:10.1145/1060590.1060650.

20 Michel X. Goemans and Jon M. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82:111–124, 1998. doi:10.1007/BF01585867.

21 Daniel Golovin, Anupam Gupta, Amit Kumar, and Kanat Tangwongsan. All-norms and all-L_p-norms approximation algorithms. In *Proceedings of the 28th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 199–210, 2008. doi:10.4230/LIPIcs.FSTTCS.2008.1753.

22 Svenja M. Griesbach, Felix Hommelsheim, Max Klimm, and Kevin Schewior. Improved approximation algorithms for the expanding search problem. *SIAM Journal on Discrete Mathematics*, 40(1):349–373, 2026. doi:10.1137/25M1759902.

23 Ben Hermans, Roel Leus, and Jannik Matuschke. Exact and approximation algorithms for the expanding search problem. *INFORMS Journal on Computing*, 34(1):281–296, 2022. doi:10.1287/ijoc.2020.1047.

24 Sharat Ibrahimpur and Chaitanya Swamy. Minimum-norm load balancing is (almost) as easy as minimizing makespan. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 81–1, 2021. doi:10.4230/LIPIcs.ICALP.2021.81.

25 Edward Minieka. The delivery man problem on a tree network. *Annals of Operations Research*, 18:261–266, 1989. doi:10.1007/BF02097807.

26 Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976. doi:10.1145/321958.321975.

27 René Sitters. The minimum latency problem is NP-hard for weighted trees. In *Proceedings of the International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 230–239, 2002. doi:10.1007/3-540-47867-1_17.

28 René Sitters. Polynomial time approximation schemes for the traveling repairman and other minimum latency problems. *SIAM Journal on Computing*, 50(5):1580–1602, 2021. doi:10.1137/19M126918X.