Improved Approximation Algorithms for the Expanding Search Problem

Svenja M. Griesbach* Felix Hommelsheim†

Max Klimm* Kevin Schewior‡

January 12, 2023

Abstract

A searcher faces a graph with edge lengths and vertex weights, initially having explored only a given starting vertex. In each step, the searcher adds an edge to the solution that connects an unexplored vertex to an explored vertex. This requires an amount of time equal to the edge length. The goal is to minimize the vertex-weighted sum of the exploration times over all vertices. We show that this problem is hard to approximate and provide algorithms with improved approximation guarantees. For the case that all vertices have unit weight, we provide a $2e$-approximation. For the general case, we give a $(5e/2+\varepsilon)$-approximation for any $\varepsilon > 0$. Previously, for both cases only an 8-approximation was known. Finally, we provide a PTAS for the case of a Euclidean graph.

1 Introduction

A vital issue faced by disaster-relief teams sent to regions devastated by natural or man-made catastrophes is to decide where to search for buried or isolated people. The fundamental issues behind these decisions are that, in emergency situations, technical means for probing and for clearing areas are often limited, there is no full knowledge concerning the whereabouts of potential survivors, and rescue operations are time-critical since the chances of survival decrease

*Technische Universität Berlin, Institute for Mathematics, Email: {griesbach,klimm}@math.tu-berlin.de. Supported by Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy, Berlin Mathematics Research Center (grant EXC-2046/1, Project 390685689) and HYPATIA.SCIENCE (Department of Mathematics and Computer Science, University of Cologne).

†Universität Bremen, Faculty of Mathematics and Computer Science, Email: fhommels@uni-bremen.de.

‡University of Southern Denmark, Department of Mathematics and Computer Science, Email: kevs@sdu.dk. Supported in part by the Independent Research Fund Denmark, Natural Sciences (grant DFF-0135-00018B).

with the time needed for rescue; see also Averbakh and Pereira [14]. Mathematically, we model this problem using an undirected graph with edge lengths. The vertices of the graph correspond to different locations in the disaster area, and the edges between them correspond to possible connections between the locations. The length of an edge corresponds to the time that is needed to clear the connection. Clearing a connection may mean to clear a road connection of rubble or explosives, or to dig in snow, dirt, or debris for survivors. There is a single rescue team initially located at a designated root vertex. Based on experience, the rescue team has knowledge about the number of survivors that is located at the different locations. The goal is to minimize the average time at which the survivors are reached.

A solution to the problem is given by a sequence of edges to clear until all vertices (with non-zero weight) can be reached. Once an edge is cleared, it can be traveled along in negligible time by the rescue team, so that only the time needed to clear edges is considered. A search problem of this kind is called *expanding search problem* (ESP) since the set of vertices accessible by the rescue team expands in every step. This is in contrast to *pathwise search problems* where the actual movement of the searcher is modeled and traversing an edge always requires time equal to the length of the edge, no matter whether it is the first traversal or not.

Generally speaking, expanding search problems are a suitable model when the time needed to traverse an edge for the first time is significantly higher than the time needed to traverse this edge any time after the first time, and, thus, the time needed for further movements can be neglected. Further applications of expanding search problems are in mining where the time needed to dig a new tunnel is much higher than moving via already dug tunnels to previously explored locations (Alpern and Lidbetter [4]) and when securing an area from a hidden explosive where the time needed to move within a safe region can be neglected compared to the time needed to secure a new area (Angelopoulos et al. [6]).

Our contribution. In this work, we provide approximation algorithms with improved approximation guarantees for ESP. We first give an approximation algorithm with an approximation guarantee of $2e \approx 5.44$ for the unweighted case where all vertices have the same weight (Theorem 1). For the general case with arbitrary vertex weights, we provide an approximation algorithm with an approximation guarantee of $5e/2 + \varepsilon \approx 6.80$ for any $\varepsilon > 0$ (Theorem 2). For both variants, the best approximation algorithm was an 8-approximation due to Hermans et al. [29]. The result for the unweighted case is obtained by concatenating k-minimum spanning trees (k-MSTs) for varying values of k and of exponentially increasing length. Using the probabilistic method on lengths with random factor finally yields an additional factor of e. This technique has been used for pathwise search problems [17, 27]; we here adapt it to the case of expanding search. For the weighted case, instead of k-MSTs, we use the quota version of the k-MST problem. As noted in [31], approximation algorithms such as the 5/2-approximation for the unrooted version of k-MST [12] that rely on the prize-collecting Steiner-tree algorithm in [28] can be turned into approximation algorithms of the quota version. We then solve the quota version for a polynomial number of quotas (thereby losing the factor of $1+\varepsilon$) and use these solutions to construct a sequence of spanning trees of exponentially increasing length. Concatenating these solutions yields the claimed factor.

We then give a polynomial-time approximation scheme (PTAS) for the case of a Euclidean graph (Theorem 3). For this result, we use a decomposition approach by Sitters [40] for the pathwise search problem. A fundamental difference between pathwise and expanding search, which surfaces here, is the following: Pathwise search is memoryless in the sense that points outside any subinstance are irrelevant for solving it whereas, in expanding search, such points may be used as Steiner points even for solving the subinstance. We address this difficulty by keeping such points in the subinstance with zero weight. In order to obtain a PTAS for the subproblem, we further adapt techniques developed by Arora [9] for the Euclidean traveling salesperson problem.

Finally, we show that there is no PTAS for ESP unless $\mathsf{P} = \mathsf{NP}$ (Theorem 4). The proof follows a similar idea as the hardness proof for the travelling repairperson problem suggested in [16]. Previously, it was only known that the expanding search problem was NP-hard [14], but the reduction did not allow to show hardness of approximation.

Further related work. The unweighted pathwise search problem where all vertices have unit weight is also known as the *traveling repairperson problem.* Sahni and Gonzales [38] showed that the problem cannot be efficiently approximated within a constant factor unless $\mathsf{P} = \mathsf{NP}$ on complete non-metric graphs when the searcher is required to take a Hamiltonian tour. Afrati et al. [1] considered the problem in metric spaces and give an algorithm with quadratic runtime when the metric is induced by a path. This can be improved to linear runtime as shown by García et al. [25]. Minieka [34] proposed a polynomial algorithm for the case that the metric is induced by an unweighted tree. Sitters [39] showed that the problem is NP-hard when the metric is induced by a tree with edge lengths 0 and 1.

The first approximation algorithm of the metric traveling repairperson problem is due to Blum et al. [16] who gave a 144-approximation. After a series of improvements [7, 8, 10, 27, 32], the best factor so far is a 3.59-approximation for general metrics [17], and a polynomial-time approximation scheme for trees [40] and on the Euclidean plane [40]. Further variants of the problem have been studied both in terms of exact solution methods and in terms of competitive algorithms, among others settings with directed edges [21, 22, 35], with processing times and time windows [43], with profits at vertices [19], with multiple searchers [18, 20, 33], and online variants [36]. The vertex-weighted version of the problem is often referred to as *the* pathwise search problem. It has been shown to be NP-hard in metric graphs by Trummel and Weisinger [42] and was further studied in [32]. The approximation schemes in [40] apply to the weighted case as well.

The expanding search problem has received considerably less attention in the literature than the pathwise problem. It has been shown to be NP-hard by Averbakh and Pereira [14]. Alpern and Gal [4] gave a polynomial-time algorithm for the case when the graph is a tree and give heuristics for general graphs. Averbakh et al. [13] considered a generalization of the problem with multiple searchers when the underlying graph is a path; Tan et al. [41] considered multiple searchers in a tree network. The first constant-factor approximation for general metrics is the 8-approximation due to Hermans et al. [29], based on an exact algorithm on trees [4]. Angelopouloas et al. [6] studied the expanding search ratio of a graph defined as the maximum ratio of time to reach a vertex by a search algorithm and by a shortest path, and they show that this ratio is NP-hard to compute.

The pathwise and expanding search problems appear naturally as strategies of the seeker in a two-player zero sum game between hider and seeker where the hider chooses a vertex that maximizes the expected search time whereas the seeker aims to minimize the search time. Gal [23] computed the value (i.e., the unique search time in an equilibrium of the game) of the pathwise search game on a tree; Alpern and Lidbetter [4] computed this value for the expanding search game; see also [5] for approximations of this value for general graphs. For more details on search games, we refer to [2, 3, 24, 30].

2 Preliminaries

We consider a connected undirected graph $G = (V, E)$ with $|V| = n$ and a designated root vertex $r \in V$. Every vertex $v \in V$ has a weight $w_v \in \mathbb{N}_{\geq 0}$, and we denote by $V^* \subseteq V$ the set of vertices with $w_v > 0$. Every edge $e \in E$ has a length $\ell_e \in \mathbb{N}_{\geq 0}$. We consider an agent that is initially located at the root and performs an *expanding search pattern* σ. Such a pattern is given by a sequence of edges $\sigma = (e_1, \dots, e_m)$ for some $m \leq n-1$ such that $r \in e_1$ and the set $\{e_1, \dots, e_i\}$ forms a tree in G for all $i \in \{1, \dots, m\}$. For a vertex $v \in V^* \setminus \{r\}$, let $k_v(\sigma) = \inf\big\{i \in \{1, \dots, m\} : v \in e_i\big\}$ be the index of the first edge that contains v and set $k_r(\sigma) = 0$. We then call $L_v(\sigma) = \sum_{i=1}^{k_v(\sigma)} \ell_{e_i}$ the *latency* of the vertex $v \in V^*$ under expanding search pattern σ. Our goal is to find an expanding search pattern σ that minimizes the *total latency* $L(\sigma) = \sum_{v \in V^*} w_v L_v(\sigma)$. Note that vertices v with $w_v = 0$ do not appear in the objective function, and, hence, do not need to be visited. When the pattern σ is clear from context, we drop the dependency on σ and simply write L, L_v, and k_v. The *length* $\ell(\sigma)$ of a search pattern σ is given by the sum of edge costs, i.e. $\ell(\sigma) = \sum_{e \in \sigma} \ell_e$. Finally, for two expanding search patterns $\sigma = (e_1, \dots, e_m), \sigma' = (e'_1, \dots, e'_{m'})$, we denote by $\sigma + \sigma'$ their concatenation, i.e., the subsequence of $(e_1, \dots, e_m, e'_1, \dots, e'_{m'})$ in which any edge closing a cycle is skipped.

3 The unweighted case

In this section, we prove the following theorem.

Theorem 1. *There is a polynomial-time* $2e$*-approximation algorithm for the unweighted expanding search problem.*

In the unweighted case, each edge $e \in E$ has an arbitrary length $\ell_e \in \mathbb{N}_{>0}$ but $w_v = 1$ for all $v \in V$. Our algorithm is an adaptation of the approximation algorithm of Chaudhuri et al. [17] for the Traveling Repairperson Problem where the task is to find a path in an undirected graph with edge lengths that minimizes the sum of latencies of the vertices.

Like the approximation algorithm of Chaudhuri et al. [17], our approximation algorithm is based on the approximate solutions of several k-minimum spanning tree (k-MST) instances, but the way how these approximate solutions are combined into an approximate solution of the original problem differs. In the rooted version of the k-MST problem, we are given an unweighted connected graph $G = (V, E)$ with a designated root vertex $r \in V$ and non-negative edge lengths $\ell_e \in \mathbb{N}_{\geq 0}$, $e \in E$. Let $\mathcal{T}_k$ denote the set of all trees $T = (V_T, E_T)$ that are subgraphs of G with $|V_T| = k$ and $r \in V_T$. The k-MST problem asks to compute a tree $T \in \mathcal{T}_k$ that minimizes $\ell(T) = \sum_{e \in E_T} \ell_e$. This problem is NP-complete; see Ravi et al. [37].

For all $k \in \{1, \dots, n\}$, we solve this problem approximately with the 2-approximation algorithm of Garg [26] and obtain n trees $T_1, T_2, \dots, T_n$, where $T_1 = (\{r\}, \emptyset)$ is the empty tree consisting of the root vertex only. Next we construct an auxiliary weighted directed graph $H = (V_H, A_H)$ with $V_H = \{1, 2, \dots, n\}$ and $A_H = \{(i, j) \in V_H^2 : i < j\}$. We construct H in such a way that any $(1, n)$-path P corresponds to some expanding search pattern σ_P such that we can upper-bound $L(\sigma_P)$ by $c(P)$. For this, a vertex $j \in V_H$ models the exploration of tree T_j. The cost $c_{i,j}$ of an edge from $i \in V_H$ to $j \in V_H$ models the delay in the exploration of the vertices not explored in T_i due to the exploration of T_j. Like this we obtain an upper bound on the latency of the vertices by assuming that these vertices will only be explored after T_j has been fully explored. Traversing edge (i, j) the exploration of $n - i$ vertices is delayed by $\ell(T_j)$; we thus set $c_{i,j} \coloneqq (n - i)\ell(T_j)$. To obtain the approximate expanding search pattern, we compute a shortest $(1, n)$-path $P = (n_0, n_1, \dots, n_l)$ in H with $n_0 = 1$, $n_l = n$, and some $l \in \mathbb{N}$. Hence, the expanding search pattern consists of l phases. In phase $j \in \{1, \dots, l\}$, we explore all edges $e \in E[T_{n_j}]$ with $|e \cap (\bigcup_{i=0}^{j-1} V[T_{n_i}])| < 2$ in an order such that the subgraph of explored vertices is always connected. In that fashion, when phase j is finished all vertices in $V[T_{n_j}]$ have been explored, and the total length of edges used in phase j is at most $\ell(T_{n_j})$. Since $n_l = n$, all vertices have been explored when the algorithm terminates.

Formally, the algorithm is given as follows:

1) For all $k \in \{1, \dots, n\}$ solve the k-MST problem with the 2-approximation algorithm of Garg [26] and obtain n trees $T_1, \dots, T_n$.

2) Construct an auxiliary weighted directed graph $H = (V_H, A_H)$ with $V_H = \{1, 2, \dots, n\}$, $A_H = \{(i, j) \in V_H^2 : i < j\}$, and $c_{i,j} \coloneqq (n - i)\ell(T_j)$.

3) Compute a shortest $(1, n)$-path $P = (n_0, n_1, \dots, n_l)$ with $n_0 = 1$ and $n_l = n$ in H.

4) For each phase $j \in \{1, \dots, l\}$ explore all unexplored vertices of $V[T_{n_j}]$ in any feasible order using the edge set of $E[T_{n_j}]$.

Let σ_{ALG} be the expanding search pattern given by this algorithm. Let v_i be the ith vertex explored according to σ_{ALG} and let $j(i) \in \{1, \dots, l\}$ be such that $n_{j(i)-1} < i \le n_{j(i)}$. Then we define $\pi(i) \coloneqq \sum_{k=0}^{j(i)} \ell(T_{n_k})$. The following lemma gives an upper bound on the latency for each individual vertex.

Lemma 1. *The latency of v_i in σ_{ALG} can be bounded by $L_{v_i}(\sigma_{\text{ALG}}) \le \pi(i)$.*

Proof. Since $n_{j(i)-1} < i \le n_{j(i)}$, vertex v_i will be explored in or before phase $j(i)$. To give an upper bound on the latency $L_{v_i}(\sigma_{\text{ALG}})$ of v_i in the expanding search pattern σ_{ALG}, note that v_i is explored at the latest if all trees T_{n_k} are nested by inclusion and v_i is visited at the very end of the tree that explores it first. Also note that the cost in any phase k is at most $c(T_{n_k})$. Thus, the latency can be bounded from above by

$$L_{v_i}(\sigma_{\text{ALG}}) \le \sum_{k=0}^{j(i)} \ell(T_{n_k}) = \pi(i).$$

This completes the proof. □

The following lemma bounds the total latency $L(\sigma_{\text{ALG}})$. Similar lemmas have been proven in related settings by Goemans and Kleinberg [27] and Chaudhuri et al. [17].

Lemma 2. *For the total latency of the algorithm, we have $L(\sigma_{\text{ALG}}) \le z$ where z is the length of a shortest $(1, n)$-path in H.*

Proof. Let $P = (n_0, \dots, n_l)$ be a shortest $(1, n)$-path in H. Its cost is equal to

$$c(P) = \sum_{j=1}^{l} (n - n_{j-1})\ell(T_{n_j}).$$

Next, consider σ_{ALG} and recall from Lemma 1 that we can bound the latency of the i-th vertex v_i in σ_{ALG} by $\pi(i)$. Taking the sum over all vertices, we obtain

$$\begin{aligned} L(\sigma_{\text{ALG}}) \le \sum_{i=1}^{n} \pi(i) = \sum_{i=1}^{n} \sum_{k=0}^{j(i)} \ell(T_{n_k}) = \sum_{j=1}^{l} (n_j - n_{j-1}) \sum_{k=0}^{j} \ell(T_{n_k}) \\ = \sum_{j=1}^{l} (n - n_{j-1})\ell(T_{n_j}) = c(P). \end{aligned}$$

Thus, the total latency is bounded from above by the cost of the path $P = (n_0, n_1, \dots, n_l)$ in H, as claimed. □

We now claim that the cost of the shortest path P from $n_0 = 1$ to $n_l = n$ in H is at most $2e$ times the total latency along the optimal sequence. To prove this claim, we consider a randomized sequence of exponentially growing subtrees and show that the path along their corresponding vertices in H has the desired properties.

Lemma 3. *Let σ^* be an optimal expanding search pattern with total latency $L^* := L(\sigma^*)$. Then, the shortest $(1, n)$-path in H has cost at most $2eL^*$.*

Proof. Our goal is to construct a path in H and compare its cost to the total latency of the optimal expanding search pattern. To do so, let L_i^* denote the latency of the ith vertex explored by the optimal expanding search pattern. Observe that no expanding search pattern can explore the ith vertex with latency less than the length of an optimal i-MST. Hence, this is also true for the optimal expanding search pattern. Since we use the 2-approximation algorithm by Garg [26] we obtain that $L_i^* \geq \frac{1}{2}\ell(T_i)$. To show the result, let $\gamma > 1$ and $b \in [1, \gamma)$ be two parameters to be optimized later. Let $\omega \in \mathbb{Z}$ be the smallest number such that $\ell(T_n) \leq 2b\gamma^\omega$, and let $\alpha \in \mathbb{Z}$ be the largest number such that $\min_{e \in E} \ell_e > \gamma^\alpha$. Then, for all $j \in \{\alpha, \dots, \omega\}$, let

$$n_j = \max\bigl\{k \in \{1, \dots, n\} : \ell(T_k) \leq 2b\gamma^j\bigr\},$$

i.e., n_j is the largest number of vertices that can be visited by one of the i-MSTs $T_1, T_2, \dots, T_n$ computed with the 2-approximation algorithm of Garg [26] such that the length of the tree is bounded from above by $2b\gamma^j$. Note that these values are well-defined since $\ell(T_1) = 0$ and that by the choices of $\alpha, \omega \in \mathbb{Z}$, we have $n_\alpha = 1$ and $n_\omega = n$. Consider the sequence $n_\alpha, n_{\alpha+1}, \dots, n_\omega$ and note that the sequence is non-decreasing. Without loss of generality, we may even assume that this sequence is strictly increasing, since otherwise we can consider the inclusion-wise largest strictly increasing subsequence of $n_\alpha, n_{\alpha+1}, \dots, n_\omega$.

First, we show a simple approach that implies an upper bound of $8L^*$ for the shortest $(1, n)$-path in H. Afterwards we will improve this upper bound to $2eL^*$.

Let v_i be the ith vertex explored according to σ_{ALG} and again let $j(i) \in \{\alpha, \dots, \omega\}$ be such that $n_{j(i)-1} < i \leq n_{j(i)}$. The cost of the path $P = (n_\alpha, n_{\alpha+1}, \dots, n_\omega)$ in H can be expressed as

$$c(P) = \sum_{k=\alpha}^{\omega} (n - n_{k-1})\ell(T_{n_k}) = \sum_{i=1}^{n} \sum_{k=\alpha}^{j(i)} \ell(T_{n_k}) = \sum_{i=1}^{n} \pi(i),$$

where we set $n_{\alpha-1} := 1$ and $\pi(i)$ is an upper bound on $L_{v_i}(\sigma_{\text{ALG}})$. Since $\ell(T_{n_{j(i)}}) \leq 2b\gamma^s =: M$ for $s = \lceil \log_\gamma(\ell(T_{n_j})/b) \rceil$ we can give an upper bound on $\pi(i)$ by

$$\pi(i) = \sum_{k=\alpha}^{j(i)} \ell(T_{n_k}) \leq \sum_{i=0}^{\infty} \frac{M}{\gamma^i} = \frac{\gamma}{\gamma-1} M.$$

By plugging in $M \leq \gamma \ell(T_i)$ we obtain $\pi(i) \leq \frac{\gamma}{\gamma-1} M \leq \frac{\gamma^2}{\gamma-1} \ell(T_i)$. Since we use a 2-approximation to compute the i-MSTs we have $\ell(T_i) \leq 2L_i^*$, which yields $\pi(i) \leq \frac{\gamma^2}{\gamma-1} \ell(T_i) \leq 2\frac{\gamma^2}{\gamma-1} L_i^*$. Thus, we obtain

$$c(P) = \sum_{i=1}^{n} \pi(i) \leq \sum_{i=1}^{n} 2\frac{\gamma^2}{\gamma-1} L_i^* = 2\frac{\gamma^2}{\gamma-1} L^*.$$

The factor is minimized for $\gamma = 2$, which yields the upper bound of $8L^*$.

However, by choosing b and γ more carefully we obtain the upper bound of $2eL^*$ as follows. Towards using the probabilistic method, let $b = \gamma^U$ where U is a random variable distributed uniformly in $[0, 1)$. The parameter γ will be determined at the very end. Again consider the path $P = (n_\alpha, n_{\alpha+1}, \ldots, n_\omega)$ in H. We compute its expected cost.

Let σ^* be an optimal expanding search pattern and $v_1^*, \ldots, v_n^*$ with $r = v_1^*$ be the vertices as they are explored by σ^*. Further let $i \in \{1, \ldots, n\}$ be arbitrary and let $j \in \{\alpha, \ldots, \omega\}$ and $d \in [1, \gamma)$ be such that $L_i^* = d\gamma^j$. Note that this is possible by our choices of α and ω. We distinguish two cases:

First case: $d \leq b$**.** Since there exists a tree containing the root with length at most $d\gamma^j \leq b\gamma^j$ that contains at least i vertices, we know that our 2-approximation of i-MST with length at most $2b\gamma^j$ visits at least i vertices, i.e., $n_j \geq i$. This allows us to bound $\pi(i)$ from above by

$$\pi(i) \leq \sum_{k=\alpha}^{j} \ell(T_{n_k}) \leq \sum_{k=\alpha}^{j} 2b\gamma^k < \sum_{k=-\infty}^{j} 2b\gamma^k = 2b\frac{\gamma^{j+1}}{\gamma-1} = 2b\gamma^j \frac{\gamma}{\gamma-1}.$$

Second case: $d > b$**.** We have that $d\gamma^j < \gamma^{j+1}$ since $d < \gamma$. This implies that there is a tree containing the root with length at most γ^{j+1} containing at least i vertices. Analogously to the argumentation in the first case, we obtain $n_{j+1} \geq i$. This allows us to bound $\pi(i)$ from above by

$$\pi(i) \leq \sum_{k=\alpha}^{j+1} \ell(T_{n_k}) \leq \sum_{k=\alpha}^{j+1} 2b\gamma^k < \sum_{k=-\infty}^{j+1} 2b\gamma^k = 2b\frac{\gamma^{j+2}}{\gamma-1} = 2b\gamma^{j+1} \frac{\gamma}{\gamma-1}.$$

In the first case, $U \in [\log_\gamma d, 1]$ and in the second case, $U \in [0, \log_\gamma d)$.

Taking the expectation over U, we obtain

$$\begin{aligned}\mathbb{E}_U\big[\pi(i)\big] &\le \int_{\log_\gamma d}^{1} 2b\gamma^j \frac{\gamma}{\gamma-1}\,\mathrm{d}U + \int_0^{\log_\gamma d} 2b\gamma^{j+1}\frac{\gamma}{\gamma-1}\,\mathrm{d}U\\ &= 2\gamma^j\frac{\gamma}{\gamma-1}\left[\int_{\log_\gamma d}^{1}\gamma^U\,\mathrm{d}U + \gamma\int_0^{\log_\gamma d}\gamma^U\,\mathrm{d}U\right]\\ &= 2\gamma^j\frac{\gamma}{\gamma-1}\left[\frac{\gamma-d}{\ln\gamma}+\gamma\frac{d-1}{\ln\gamma}\right]\\ &= 2\gamma^j d\frac{\gamma}{\ln\gamma}\\ &= 2L_i^*\frac{\gamma}{\ln\gamma}.\end{aligned}$$

Therefore, using $c(P)=\sum_{i=1}^n \pi(i)$, we obtain

$$\frac{\mathbb{E}[c(P)]}{L^*} = \frac{\mathbb{E}[\sum_{i=1}^n \pi(i)]}{L^*} = \frac{\sum_{i=1}^n \mathbb{E}\big[\pi(i)\big]}{\sum_{i=1}^n L_i^*} \le \frac{2\frac{\gamma}{\ln\gamma}\sum_{i=1}^n L_i^*}{\sum_{i=1}^n L_i^*} = \frac{2\gamma}{\ln\gamma}.$$

This quantity is minimized for $\gamma = e$ for which the approximation ratio turns out to be $2e \approx 5.44$. Hence, the randomized path P has expected cost at most $2eL^*$. Therefore, the cost of a shortest $(1,n)$-path in H can also be bounded by the same value. □

Hence, Lemma 2 and Lemma 3 imply Theorem 1.

4 The weighted case

In this section, we consider the general case of the expanding search problem where the weights w_v, $v \in V \setminus \{r\}$, are arbitrary and $w_r = 0$ and prove the following theorem.

Theorem 2. *For every $\varepsilon > 0$, there is a polynomial-time $(5e/2+\varepsilon)$-approximation algorithm for the expanding search problem.*

The approximation algorithm that we devise in this section is based on the approximate solution of several quota versions of the prize-collecting Steiner tree problem. In this problem, we are given a connected undirected graph $G=(V,E)$ with designated root vertex $r \in V$, non-negative edge lengths $\ell_e \in \mathbb{R}_{\ge 0}, e \in E$, vertex weights $w_v \in \mathbb{N}$, $v \in V \setminus \{r\}$, and a quota $q \in \{0,1,\dots,W\}$, where $W := \sum_{v\in V} w_v$. The task is to find a subgraph that is a tree $T=(V_T,E_T)$ such that $r \in V_T$ and $\sum_{v\in V_T} w_v \ge q$ minimizing $\ell(T) := \sum_{e\in E_T}\ell_e$.

We argue that this problem admits a 5/2-approximation.

Lemma 4. *For the quota version of the prize-collecting Steiner tree problem, a 5/2-approximation can be computed in polynomial time.*

Proof. We solve the quota version of the prize-collecting Steiner tree problem by running an approximation algorithm for the k-MST problem. To obtain a polynomial-time algorithm, we make use of the following construction due to Johnson et al. [31]. For each vertex v, we add $2nw_v$ many vertices and connect these to the original vertex v with edges of cost 0. The multiplication with $2n$ is necessary to handle vertices of weight 0. Afterwards, we set k to $2nq$ where n is the number of vertices in the original instance and q is the quota of the original instance. Now running a k-MST algorithm on the obtained sequence requires pseudo-polynomial time. However, Johnson et al. [31] noticed that for many approximation algorithms used for approximating the k-MST problem the running time can be reduced to polynomial time. This holds in particular for algorithms relying on the approximation algorithm for prize-collecting Steiner tree by Goemans and Williamson [28]. This algorithm in a first step actually joins all vertices connected by edges of length 0 to a single component, so that it would yield a polynomial running time on instances for which the above construction is used. Specifically, Johnson et al. [31] note that the 5/2-approximation of Arya and Ramesh [12] for the unrooted version of the k-MST problem yields also a polynomial 5/2-approximation for the unrooted quota version of the prize-collecting Steiner tree problem. However, as already noted by Garg [26] for the unweighted case, an algorithm for the unrooted version can be turned into an algorithm for the rooted version by attaching a large-enough weight to the root. Specifically, adding an artificial weight of W to the root and changing the quota to $W + q$ restricts the set of feasible solutions to those containing the root. □

To approximate ESP, fix $\varepsilon > 0$. We solve the quota problem for the quotas

$$W - W(1+\varepsilon)^{-i} \text{ for all } i \in \{0, \dots, \omega\},$$

where we let

$$\omega = \left\lceil \frac{\log W}{\log(1+\varepsilon)} \right\rceil.$$

Note that, for fixed ε, the number ω is polynomial in the encoding length of the input. In this way, we obtain $\omega+1$ trees $T_0, T_1, \dots, T_\omega$. By construction, tree T_0 has to collect a total weight of 0, so T_0 is the tree $T_0 = (\{r\}, \emptyset)$ consisting only of the root vertex. By the choice of ω, the tree T_ω has to collect a total weight of $W - W(1+\varepsilon)^{-\omega} > W - 1$. This implies that the tree T_ω collects all weight since the weights are integers. We then construct a directed auxiliary graph $H = (V_H, A_H)$ with vertex set $V_H = \{0, \dots, \omega\}$ and arc set $A_H = \{(i,j) : i < j\}$. We set the cost of arc (i,j) equal to $c_{i,j} = W(1+\varepsilon)^{-i}\ell(T_j)$. Next, we compute a shortest $(0,\omega)$-path $P = (n_0, \dots, n_l)$ with $n_0 = 0$ and $n_l = \omega$ for some $l \in \mathbb{N}$. Again as in the unweighted case, we construct from this path an expanding search pattern with l phases. In phase $j \in \{1, \dots, l\}$, we explore all edges in $e \in E[T_{n_j}]$ with $|e \cap (\bigcup_{i=0}^{j-1} V[T_{n_i}])| < 2$ in an order such that the subgraph of explored vertices is connected at all times. In that fashion, when phase j is finished, all vertices in $V[T_{n_j}]$ have been explored. Since $n_l = \omega$ and T_ω collects the total weight W, all vertices v with $w_v > 0$ have been explored when the algorithm terminates.

Formally, the algorithm is given as follows:

1) For all $i \in \{0, 1, \ldots, \omega\}$ solve the quota version of prize-collecting Steiner tree with quota $q = W - W(1+\varepsilon)^{-i}$ with the 5/2-approximation algorithm of Lemma 4 and obtain $\omega + 1$ trees $T_0, T_1, \ldots, T_\omega$.

2) Construct an auxiliary weighted directed graph $H = (V_H, A_H)$ with $V_H = \{0, 1, \ldots, \omega\}$, $A_H = \{(i, j) \in V_H^2 : i < j\}$, and $c_{i,j} := W(1+\varepsilon)^{-i}\ell(T_j)$.

3) Compute a shortest $(0, \omega)$-path $P = (n_0, n_1, \ldots, n_l)$ with $n_0 = 0$ and $n_l = \omega$ in H.

4) For each phase $j \in \{1, \ldots, l\}$ explore all unexplored vertices of $V[T_{n_j}]$ in any feasible order using the edge set of $E[T_{n_j}]$.

Let σ_{ALG} be the expanding search pattern given by this algorithm. Similar to the previous section, we define π of some quota. Let $q \in [0, W]$ be arbitrary and let $j(q) \in \{1, \ldots, l\}$ be chosen such that

$$W - W(1+\varepsilon)^{-n_{j(q)-1}} \leq q < W - W(1+\varepsilon)^{-n_{j(q)}}.$$

Then we define $\pi(q) := \sum_{i=0}^{j(q)} \ell(T_{n_i})$. The following lemma gives an upper bound on the latency for each quota.

Lemma 5. *The latency of quota $q \in [0, W]$ in σ_{ALG} can be bounded by $L_q(\sigma_{\text{ALG}}) \leq \pi(q)$.*

Proof. Since $W - W(1+\varepsilon)^{-n_{j(q)-1}} \leq q < W - W(1+\varepsilon)^{-n_{j(q)}}$, the algorithm has explored a total weight of q not later than in phase $j(q)$. Analogously to the unweighted case, quota q is explored at the latest if all trees T_{n_k} are nested by inclusion and all vertices are visited at the very end of the tree that explores it first. Also note that the cost in any phase k is at most $c(T_{n_k})$. Thus, the latency can be bounded from above by

$$L_q(\sigma_{\text{ALG}}) \leq \sum_{i=0}^{j(q)} \ell(T_{n_i}) = \pi(q).$$

This completes the proof. □

Using this lemma we can now give an upper bound on $L(\sigma_{\text{ALG}})$.

Lemma 6. *For the total latency of the algorithm, we have $L(\sigma_{\text{ALG}}) \leq z$ where z is the cost of a shortest $(0, \omega)$-path in H.*

Proof. The proof of this lemma is very similar to the one of Lemma 2. Let $P = (n_0, \dots, n_l)$ be a shortest $(0, \omega)$-path in H. Its cost is equal to

$$c(P) = \sum_{k=1}^{l} W(1+\varepsilon)^{-n_{k-1}} \ell(T_{n_k}).$$

Next, consider σ_{ALG} and recall from Lemma 5 that we can bound the latency of quota q in σ_{ALG} by $\pi(q)$. Then, we obtain

$$\begin{aligned}
L(\sigma_{\text{ALG}}) &\le \int_0^W \pi(q)\,\mathrm{d}q \\
&= \int_0^W \sum_{i=0}^{j(q)} \ell(T_{n_i})\,\mathrm{d}q \\
&= \quad \sum_{k=1}^{l} \Big[\big(W - W(1+\varepsilon)^{-n_k}\big) - \big(W - W(1+\varepsilon)^{-n_{k-1}}\big) \Big] \sum_{i=0}^{k} \ell(T_{n_i}) \\
&\quad + W(1+\varepsilon)^{-n_l} \sum_{i=0}^{l} \ell(T_{n_i}) \\
&= \quad \sum_{k=1}^{l} W(1+\varepsilon)^{-n_{k-1}} \sum_{i=0}^{k} \ell(T_{n_i}) - \sum_{k=1}^{l} W(1+\varepsilon)^{-n_k} \sum_{i=0}^{k} \ell(T_{n_i}) \\
&\quad + W(1+\varepsilon)^{-n_l} \sum_{i=0}^{l} \ell(T_{n_i}) \\
&= W(1+\varepsilon)^0 \ell(T_{n_1}) + \sum_{k=2}^{l} W(1+\varepsilon)^{-n_{k-1}} \ell(T_{n_k}) \\
&= \sum_{k=1}^{l} W(1+\varepsilon)^{-n_{k-1}} \ell(T_{n_k}) \\
&= c(P).
\end{aligned}$$

Thus, the total latency of σ_{ALG} is bounded from above by the cost of the path $(n_0, n_1, \dots, n_l)$ in H, as claimed. □

We proceed to bound the cost of a shortest path in relation to the total latency of the optimal expanding search pattern.

Lemma 7. *Let σ^* be an optimal expanding search pattern with total latency $L^* := L(\sigma^*)$. Then, the shortest $(0, \omega)$-path in H has cost at most $\frac{5}{2}(1+\varepsilon)eL^*$.*

Proof. First, we give a lower bound on L^*. For this purpose, let $q \in [0, W]$ be arbitrary, and let $\lambda^*(q)$ denote the length of the optimal solution to the instance of the quota version of rooted prize-collecting Steiner tree with quota q. Note that there are only finitely many trees T that are subgraphs of G and contain r, so λ^* is a piece-wise constant function. The optimal expanding search pattern cannot achieve a total weight of q with a latency smaller than $\lambda^*(q)$. Therefore,

$$L^* \geq \int_0^W \lambda^*(q)\,\mathrm{d}q.$$

To show that the cost of the shortest $(0,\omega)$-path in H is bounded from above by $\frac{5}{2}(1+\varepsilon)eL^*$, we construct a random path and compute its expected cost. Let $\gamma > 1$ be a parameter whose value will be determined later, and let $b = \gamma^U$ where U is a random variable uniformly drawn from $[0,1)$. We set $m \in \mathbb{N}$ to be the smallest number such that $\ell(T_\omega) \leq \frac{5}{2}b\gamma^m$. For $j \in \{0,\dots,m\}$, let

$$n_j := \max\left\{k \in \{0,\dots,\omega\} : \ell(T_k) \leq \frac{5}{2}b\gamma^j\right\}.$$

These values are well-defined as $\ell(T_0) = 0$. Note that the sequence $n_0, n_1, \dots, n_m$ is non-decreasing. Without loss of generality, we may even assume that this sequence is strictly increasing, since otherwise we can consider the inclusion-wise largest strictly increasing subsequence of $n_0, n_1, \dots, n_m$. In the following, we compute the cost of the path $P = (n_0, n_1, \dots, n_m)$. Let $i \in \{0,\dots,m\}$ be such that

$$W - W(1+\varepsilon)^{-i} \leq q < W - W(1+\varepsilon)^{-(i+1)}.$$

Recall that $\pi(q) = \sum_{j=0}^{i+1} \ell(T_{n_j})$ is an upper bound on the latency of quota q, i.e., the sum of the lengths of all trees that lie on path P up to the first tree that collects a quota of size q all by itself.

For a quota $q \in [0,W]$ to be collected we find it convenient to denote by $\bar{q} := W - q$ the quota left aside. Let $\bar{q} \in [W(1+\varepsilon)^{-\omega}, W]$ be arbitrary, and let $j \in \{0,\dots,m\}$ and $d \in [1,\gamma)$ be such that $\lambda^*(W-\bar{q}) = d\gamma^j$. We distinguish two cases regarding the relation between b and d.

First case: $d \leq b$**.** Since $\lambda^*(q) = \lambda^*(W-\bar{q}) = d\gamma^j$, there is a tree of cost $d\gamma^j$ containing the root that explores a total weight of at least $W-\bar{q}$. When computing the $(5/2)$-approximation for the quota version of the prize-collecting Steiner tree problem with quota $W - W(1+\varepsilon)^{-i}$, we obtain a tree T_i with length

$$\ell(T_i) \leq \frac{5}{2}\lambda^*\big(W - W(1+\varepsilon)^{-i}\big) \leq \frac{5}{2}\lambda^*(W-\bar{q}) = \frac{5}{2}d\gamma^j,$$

where we obtain the first inequality from the fact that we use a $(5/2)$-approximation and the second inequality from λ^* being non-decreasing. Since $\ell(T_i) \leq \frac{5}{2}d\gamma^j \leq \frac{5}{2}b\gamma^j$, we have that $n_j \geq i$. Using that $\pi(q)$ is non-decreasing and that $W - W(1+\varepsilon)^{-i} \geq W - (1+\varepsilon)\bar{q}$, we obtain

$$\pi\big(W-(1+\varepsilon)\bar{q}\big) \leq \sum_{k=0}^{j} \ell(T_{n_k}) \leq \sum_{k=0}^{j} \frac{5}{2}b\gamma^k = \frac{5}{2}b\frac{\gamma^{j+1}-1}{\gamma-1} \leq \frac{5}{2}b\gamma^j\frac{\gamma}{\gamma-1},$$

where we use that $\gamma > 1$.

Second case: $d > b$**.** Analogously to the first case we obtain $\ell(T_i) \le \frac{5}{2} d\gamma^j$. However, with $1 \le b$ and $d < \gamma$ we have $d < b\gamma$ which yields

$$\ell(T_i) \le \frac{5}{2} d\gamma^j < \frac{5}{2} b\gamma^{j+1}.$$

Hence, we have that $n_{j+1} \ge i$. Using that $\pi(q)$ is non-decreasing and that $W - W(1+\varepsilon)^{-i} \ge W - (1+\varepsilon)\bar{q}$, we obtain

$$\pi\big(W - (1+\varepsilon)\bar{q}\big) \le \sum_{k=0}^{j+1} \ell(T_{n_k}) \le \sum_{k=0}^{j+1} \frac{5}{2} b\gamma^k = \frac{5}{2} b \frac{\gamma^{j+2}-1}{\gamma-1} \le \frac{5}{2} b\gamma^{j+1} \frac{\gamma}{\gamma-1},$$

where we again use that $\gamma > 1$.

Note that we are in the first case when $U \in [\log_\gamma d, 1]$ and in the second case when $U \in [0, \log_\gamma d)$. Taking the expectation over U, we obtain

$$\begin{aligned}
\mathbb{E}_U\big[\pi\big(W - (1+\varepsilon)\bar{q}\big)\big] &\le \int_{\log_\gamma d}^{1} \frac{5}{2} b\gamma^j \frac{\gamma}{\gamma-1}\, \mathrm{d}U + \int_0^{\log_\gamma d} \frac{5}{2} b\gamma^{j+1} \frac{\gamma}{\gamma-1}\, \mathrm{d}U \\
&= \frac{5}{2}\gamma^j \frac{\gamma}{\gamma-1} \left[\int_{\log_\gamma d}^{1} \gamma^U \,\mathrm{d}U + \gamma \int_0^{\log_\gamma d} \gamma^U \,\mathrm{d}U\right] \\
&= \frac{5}{2}\gamma^j \frac{\gamma}{\gamma-1} \left[\frac{\gamma - d}{\ln\gamma} + \gamma \frac{d-1}{\ln\gamma}\right] \\
&= \frac{5}{2}\gamma^j d \frac{\gamma}{\ln\gamma} \\
&= \frac{5}{2}\frac{\gamma}{\ln\gamma} \lambda^*(W - \bar{q}). \qquad (1)
\end{aligned}$$

Next, consider the case that $\bar{q} < W(1+\varepsilon)^{-\omega}$ is arbitrary. Let $j \in \{0, \dots, m\}$ and $d \in [1, \gamma)$ be such that $\lambda^*(W - \bar{q}) = d\gamma^j$. By the choice of ω, we have $W - \bar{q} > W - W(1+\epsilon)^{-\omega} > W - 1$, and, hence, $\lambda^*(W) = \lambda^*(W - \bar{q})$. We distinguish two cases regarding the relation between b and d.

First case: $d \le b$**.** Since $\lambda^*(W) = d\gamma^j$, there is a tree of cost $d\gamma^j$ containing the root that explores a total weight of at least $W - \bar{q}$. When computing the (5/2)-approximation for the quota version of the prize-collecting Steiner tree problem with quota $W - W(1+\varepsilon)^{-\omega}$, we obtain a tree T_ω with length

$$\ell(T_\omega) \le \frac{5}{2}\lambda^*(W) = \frac{5}{2} d\gamma^j,$$

where we obtain the inequality from the fact that we use a (5/2)-approximation. Since $\ell(T_\omega) \le \frac{5}{2} d\gamma^j \le \frac{5}{2} b\gamma^j$, we have that $n_j \ge \omega$, thus, $j = m$. We obtain

$$\pi\big(W\big) \le \sum_{k=0}^{m} \ell(T_{n_k}) \le \sum_{k=0}^{m} \frac{5}{2} b\gamma^k = \frac{5}{2} b \frac{\gamma^{m+1}-1}{\gamma-1} \le \frac{5}{2} b\gamma^m \frac{\gamma}{\gamma-1},$$

where we use that $\gamma > 1$.

Second case: $d > b$**.** Analogously to the first case we obtain $\ell(T_\omega) \leq \frac{5}{2} d\gamma^j$. However, with $1 \leq b$ and $d < \gamma$ we have $d < b\gamma$ which yields

$$\ell(T_\omega) \leq \frac{5}{2} d\gamma^j < \frac{5}{2} b\gamma^{j+1}.$$

Hence, we have that $n_{j+1} \geq \omega$, i.e., $j \geq m-1$. In any case, we obtain

$$\pi(W) \leq \sum_{k=0}^{m} \ell(T_{n_k}) \leq \sum_{k=0}^{m} \frac{5}{2} b\gamma^k = \frac{5}{2} b \frac{\gamma^{m+1}-1}{\gamma-1} \leq \frac{5}{2} b\gamma^m \frac{\gamma}{\gamma-1},$$

where we again use that $\gamma > 1$.

Again, we are in the first case when $U \in [\log_\gamma d, 1]$ and in the second case when $U \in [0, \log_\gamma d)$. Taking the expectation over U, we obtain

$$\begin{aligned}
\mathbb{E}_U\big[\pi\big(W\big)\big] &\leq \int_{\log_\gamma d}^{1} \frac{5}{2} b\gamma^j \frac{\gamma}{\gamma-1}\, \mathrm{d}U + \int_0^{\log_\gamma d} \frac{5}{2} b\gamma^{j+1} \frac{\gamma}{\gamma-1}\, \mathrm{d}U \\
&= \frac{5}{2}\gamma^j \frac{\gamma}{\gamma-1}\left[\int_{\log_\gamma d}^{1} \gamma^U\, \mathrm{d}U + \gamma \int_0^{\log_\gamma d} \gamma^U\, \mathrm{d}U\right] \\
&= \frac{5}{2}\gamma^j \frac{\gamma}{\gamma-1}\left[\frac{\gamma-d}{\ln\gamma} + \gamma\frac{d-1}{\ln\gamma}\right] \\
&= \frac{5}{2}\gamma^j d \frac{\gamma}{\ln\gamma} \\
&= \frac{5}{2}\frac{\gamma}{\ln\gamma}\lambda^*(W). \qquad (2)
\end{aligned}$$

For the expected cost of the $(0,\omega)$-path $P = (n_0, n_1, \ldots, n_m)$, we obtain

$$\mathbb{E}\Bigg[c(P)\Bigg] = \mathbb{E}\Bigg[\int_0^W \pi(q)\,\mathrm{d}q\Bigg] = \mathbb{E}\Bigg[\int_0^W \pi(W-\bar{q})\,\mathrm{d}\bar{q}\Bigg].$$

Since $\pi(q)$ is piece-wise constant, we can exchange the order of expectation and integral and obtain

$$\begin{aligned}
\mathbb{E}\big[c(P)\big] = &\int_0^W \mathbb{E}\big[\pi(W-\bar{q})\big]\,\mathrm{d}\bar{q} \\
= &-(1+\varepsilon)\int_{W(1+\varepsilon)^{-1}}^{W(1+\varepsilon)^{-\omega}} \mathbb{E}\big[\pi\big(W-(1+\varepsilon)\bar{q}\big)\big]\,\mathrm{d}\bar{q} \\
&-(1+\varepsilon)\int_{W(1+\varepsilon)^{-\omega}}^{0} \mathbb{E}\big[\pi\big(W-(1+\varepsilon)\bar{q}\big)\big]\,\mathrm{d}\bar{q} \\
\leq\ &\ (1+\varepsilon)\int_{W(1+\varepsilon)^{-\omega}}^{W(1+\varepsilon)^{-1}} \mathbb{E}\big[\pi\big(W-(1+\varepsilon)\bar{q}\big)\big]\,\mathrm{d}\bar{q} \\
&+ W(1+\varepsilon)^{-(\omega-1)}\mathbb{E}\big[\pi(W)\big],
\end{aligned}$$

where we further used the substitution rule for integrals and the fact that π is non-decreasing. Using (2) we further obtain

$$\begin{aligned}\mathbb{E}\big[c(P)\big] &\le \frac{5\gamma(1+\varepsilon)}{2\ln\gamma}\left[\int_{W(1+\varepsilon)^{-\omega}}^{W(1+\varepsilon)^{-1}} \lambda^*(W-\bar{q})\,\mathrm{d}\bar{q} + W(1+\varepsilon)^{-\omega}\mathbb{E}\big[\lambda^*(W)\big]\right] \\ &= \frac{5\gamma(1+\varepsilon)}{2\ln\gamma}\int_{0}^{W(1+\varepsilon)^{-1}} \lambda^*(W-\bar{q})\,\mathrm{d}\bar{q},\end{aligned}$$

where for the equation we used that $W(1+\varepsilon)^{-\omega} < 1$ and, hence, λ^* is constant on the interval $[W - W(1+\varepsilon)^{-\omega}, W]$. Finally, we obtain

$$\mathbb{E}\big[c(P)\big] \le \frac{5\gamma(1+\varepsilon)}{2\ln\gamma}\int_0^W \lambda^*(W-\bar{q})\,\mathrm{d}\bar{q} = \frac{5\gamma(1+\varepsilon)}{2\ln\gamma}\int_0^W \lambda^*(q)\,\mathrm{d}q \le \frac{5\gamma(1+\varepsilon)}{2\ln\gamma}L^*.$$

This shows that, in the expectation over U, P has an expected cost of at most $\frac{5\gamma(1+\varepsilon)}{2\ln\gamma}L^*$. Therefore we obtain the same bound on the cost of the shortest path. This term is minimized for $\gamma = e$, which implies the result. □

Lemma 6 and Lemma 7 together imply Theorem 2.

5 The Euclidean case

In this section, we show the following theorem.

Theorem 3. *On Euclidean graphs, there exists a PTAS for ESP.*

Our approach has three steps, corresponding to the three subsections of this section. The first two steps are reductions inspired by Sitters [40]. In the first step, we show a reduction from ESP to a problem called δ-bounded ESP, for some constant $\delta \in \mathbb{R}_+$, in the sense that a PTAS for δ-bounded ESP implies a PTAS for ESP. In the next step, we reduce the latter problem to another problem called κ-segmented ESP, for some constant $\kappa \in \mathbb{N}$, with weights in $\{0, 1\}$, in the same sense as before. Finally, we provide a PTAS for the latter problem in the Euclidean case using ideas by Arora [9] as well as Sitters [40].

We define the auxiliary subproblems as modifications of ESP. First, in δ-bounded ESP, the input comes with an additional delay parameter $L \ge 0$, and it is guaranteed that there exists a solution visiting all nonzero-weight vertices and completing by time δL, i.e. this solution has length δL (recall definition in Section 2). The objective is minimizing $C'(\sigma) = \sum_{v\in V^*} w_v C'_v(\sigma)$ where $C'_v(\sigma) = L + C_v(\sigma)$. Second, in κ-segmented ESP, the output needs to come with $\kappa + 1$ additional numbers $0 = t^{(0)} \le t^{(1)} \le \cdots \le t^{(\kappa)}$. For $v \in V$, its rounded search time is then $\bar{C}_v(\sigma) = \inf\{t^{(i)} : 0 \le i \le \kappa,\, C_v(\sigma) \le t^{(i)}\}$, and the objective is minimizing $\bar{C}(\sigma) = \sum_{v\in V^*} w_v \bar{C}_v(\sigma)$.

We assume $0 < \varepsilon \le 1$ and use $O_\varepsilon(f)$ to denote $O(f)$ when ε is a constant.

5.1 Reducing ESP to δ-Bounded ESP

In this subsection, we show the following lemma.

Lemma 8. *Consider any class $\mathcal{C}$ of metric spaces. There exists a constant δ such that, if there exists a PTAS for δ-bounded ESP on $\mathcal{C}$, then there exists a PTAS for ESP on $\mathcal{C}$.*

We follow the decomposition approach by Sitters [40] and adapt it to ESP at several places. To do so, we assume that a PTAS for δ-bounded ESP on $\mathcal{C}$, denoted $\mathrm{PTAS}_{\delta\text{-bd}}$ in the following, is given for a yet-to-be-determined value of δ. In the remainder of this subsection we describe, given any $\varepsilon > 0$, a polynomial-time algorithm for ESP on $\mathcal{C}$ based on this, and we show that it is a $(1+\varepsilon)$-approximation algorithm.

For some constant β, we need, in addition to $\mathrm{PTAS}_{\delta\text{-bd}}$, a polynomial-time β-approximation algorithm $\textsc{Approx}_\beta$ for ESP on $\mathcal{C}$ as a subroutine. We emphasize that any constant β is sufficient to obtain an approximation guarantee of $1+\varepsilon$ in polynomial time. Therefore, we can pick, e.g., the algorithm from Section 4.

In our algorithm, we apply $\textsc{Approx}_\beta$ to obtain an order of the vertices according to their search times in the solution, and we obtain a partition of the vertices by cutting this order at several places. We run $\mathrm{PTAS}_{\delta\text{-bd}}$ on the (carefully defined) emerging subinstances. We can, however, not simply concatenate all these solutions because any of these solutions may, despite its low cost, have large total length, which would delay the solutions of all later subinstances. We solve this issue by cutting the solution at a certain point and using the solution given by $\textsc{Approx}_\beta$ from then on—a solution with a length bound.

In the following, we present our algorithm which is given some $\varepsilon > 0$ as well as an instance I of ESP on $\mathcal{C}$. Our algorithm has five steps.

1) **Approximate:** Apply $\textsc{Approx}_\beta$ to the instance to obtain a solution σ_β.

2) **Partition:**

   - Define $\gamma := 3/\varepsilon$, $a := \beta\gamma/\varepsilon$, and pick b uniformly at random in $[0, a]$.
   - Define time points $t_i := e^{(i-2)a+b}$ for $i \in [q+1]$, where q is as small as possible such that $C_v(\sigma_\beta) < t_{q+1}$ for all $v \in V$.
   - For $i \in [q]$, let $V_i := \{v \in V : t_i \leq C_v(\sigma_\beta) < t_{i+1}\}$ and $U_i := V_1 \cup \cdots \cup V_i$.
   - For $i \in [q]$, define I_i to be an instance which is obtained from I by setting the weight of all vertices in $V \setminus V_i$ to zero. Note that I_i with delay parameter γt_i is an instance of (e^a/γ)-bounded ESP. Indeed, the prefix $\sigma_{\beta,i+1}$ of σ_β visiting U_{i+1} has total length at most $(e^a/\gamma)\gamma t_i = t_{i+1}$.

3) **Approximate subproblems:** For $i \in [q]$, apply $\mathrm{PTAS}_{\delta\text{-bd}}$ to I_i to obtain an $(1+\varepsilon)$-approximation $\sigma_{1+\varepsilon,i}$.

4) **Modify:** For each $i \in [q]$, define σ_i to be $\sigma'_{1+\varepsilon,i} + \sigma_{\beta,i+1}$ where:

   – $\sigma'_{1+\varepsilon,i}$ is the longest prefix of $\sigma_{1+\varepsilon,i}$ of length at most $(1 + {}^{e^a}/_{\varepsilon\gamma})\gamma t_i$.
   – $\sigma_{\beta,i+1}$ is the prefix of σ_β visiting U_{i+1}.

5) **Concatenate:** Return $\sigma_1 + \cdots + \sigma_q$.

We show Lemma 8 by establishing two lemmata on the above algorithm. We first prove that partitioning the instance into multiple instances of $({}^{e^a}/_\gamma)$-bounded ESP is only at the loss of a $1+\varepsilon$ factor in the achievable (total) objective-function value. To make this formal, we denote by σ^* an optimal solution for I and, for all $i \in [q]$, by σ_i^* an optimal solution for I_i.

Lemma 9. *It holds that*

$$\mathbb{E}\left[\sum_{i\in[q]} C'(\sigma_i^*)\right] \le (1+\varepsilon)C(\sigma^*).$$

Proof. First observe that σ^* is a solution to I_i, for any $i \in [q]$. Therefore,

$$\sum_{v\in V_i^*} w_v(\gamma t_i + C_v(\sigma_i^*)) \le \sum_{v\in V_i^*} w_v(\gamma t_i + C_v(\sigma^*)).$$

Summing over all $i \in [q]$ and taking expectation, we obtain

$$\begin{aligned}\mathbb{E}\left[\sum_{i\in[q]}\sum_{v\in V_i^*} w_v(\gamma t_i + C_v(\sigma_i^*))\right] &\le \mathbb{E}\left[\sum_{i\in[q]}\sum_{v\in V_i^*} w_v(\gamma t_i + C_v(\sigma^*))\right]\\ &= \sum_{v\in V^*} w_v C_v(\sigma^*) + \sum_{v\in V^*} \gamma w_v \mathbb{E}[t_{i(v)}], \end{aligned} \tag{3}$$

where $i(v)$ is defined in such a way that $v \in V_{i(v)}$ for each $v \in V$. To analyze the second summand on the right-hand side, note that $t_{i(v)}$ is a random variable of the form $t_{i(v)} = e^{-x}C_v(\sigma_\beta)$ where x is uniform on $[0, a]$. Hence,

$$\mathbb{E}[t_{i(v)}] = \frac{C_v(\sigma_\beta)}{a}\int_0^a e^{-x}\,\mathrm{d}x = \frac{C_v(\sigma_\beta)}{a}(1-e^{-a}) < \frac{C_v(\sigma_\beta)}{a}.$$

Plugging this into Inequality (3) yields

$$\begin{aligned}\mathbb{E}\left[\sum_{i\in[q]}\sum_{v\in V_i^*} w_v(\gamma t_i + C_v(\sigma_i^*))\right] &\le \sum_{v\in V^*} w_v C_v(\sigma^*) + \frac{\gamma}{a}\sum_{v\in V^*} w_v C_v(\sigma_\beta)\\ &= \sum_{v\in V^*} w_v C_v(\sigma^*) + \frac{\varepsilon}{\beta}\sum_{v\in V^*} w_v C_v(\sigma_\beta)\\ &\le (1+\varepsilon)\sum_{v\in V^*} w_v C_v(\sigma^*),\end{aligned}$$

using that σ_β is a β-approximation in the last step. □

The next lemma is concerned with Step 4 of the algorithm. For all $i \in [q]$, it bounds the cost of σ_i against the cost of σ_i^*, and it also bounds the total length of σ_i.

Lemma 10. *For each $i \in [q]$, the total length of σ_i is at most $\gamma t_{i+1} - \gamma t_i$. Furthermore, it holds that*

$$C'(\sigma_i) \leq (1+\varepsilon)^2 C'(\sigma_i^*). \tag{4}$$

Proof. Consider an $i \in [q]$. By construction, the total length of σ_i is at most

$$\left(1 + \frac{e^a}{\varepsilon\gamma}\right)\gamma t_i + t_{i+1} = \left(1 + \frac{e^a}{\varepsilon\gamma} + \frac{e^a}{\gamma}\right)\gamma t_i < (e^a - 1)\gamma t_i = \gamma t_{i+1} - \gamma t_i,$$

where we use $\varepsilon \leq 1$, $\gamma \geq 3$, and $a \geq 3$ for the inequality.

Regarding the search times $C_v(\sigma_i)$, we consider each vertex $v \in V_i$ with $w_v > 0$ individually. First note that, for any v visited by the first part of σ_i, i.e., $\sigma'_{1+\varepsilon,i}$, the search time $C_v(\sigma_i)$ remains unchanged with respect to $C_v(\sigma_{1+\varepsilon,i})$. For any other $v \in V_i$, observe that it holds that $C_v(\sigma_i) \leq (1 + {e^a}/{\varepsilon\gamma} + {e^a}/{\gamma})\gamma t_i$ while $C_v(\sigma_{1+\varepsilon,i}) \geq (1 + {e^a}/{\varepsilon\gamma})\gamma t_i$. Combining these two inequalities, we obtain

$$\frac{\gamma t_i + C_v(\sigma_i)}{\gamma t_i + C_v(\sigma_{1+\varepsilon,i})} \leq \frac{2 + \frac{e^a}{\varepsilon\gamma} + \frac{e^a}{\gamma}}{2 + \frac{e^a}{\varepsilon\gamma}} < 1 + \frac{2\varepsilon + \frac{e^a}{\gamma}}{2 + \frac{e^a}{\varepsilon\gamma}} = 1 + \varepsilon.$$

Summing over all vertices now yields

$$\sum_{v \in V_i^*} w_v(\gamma t_i + C_v(\sigma_i)) \leq (1+\varepsilon) \sum_{v \in V_i^*} w_v(\gamma t_i + C_v(\sigma_{1+\varepsilon,i})).$$

Using that $\sigma_{1+\varepsilon,i}$ is a $(1+\varepsilon)$-approximation for I_i completes the proof. □

With these lemmata at hand, Lemma 8 easily follows.

Proof of Lemma 8. Note that Lemma 10 implies that, in the concatenation of $\sigma_1, \ldots, \sigma_q$, σ_i starts after a total length of at most γt_i, for all $i \in [q]$. Therefore, again by Lemma 10, the cost of the concatenation, as a solution to I, has expected cost at most the left-hand side of Inequality (4) summed over all $i \in [q]$. Hence, applying this inequality, taking expectation, and then applying Lemma 9 completes the proof. □

The partition as stated in the decomposition algorithm is a random partition. We note that the algorithm can be derandomized using the same techniques as in [40], i.e., by enumerating all partitions.

We also observe that, from now on, we may also assume that all weights are in $\{0, 1\}$. This is due to the following lemma, proven by Sitters [40] for pathwise search, but the same proof works in our case.

Lemma 11 (See [40], Lemma 2.10)**.** *Consider any class $\mathcal{C}$ of metric spaces and any constant $\delta > 0$. If there exists a PTAS for δ-bounded ESP with weights in $\{0, 1\}$, then there exists a PTAS for δ-bounded ESP.*

In the proof, one first simply sets $w'_v = \lfloor w_v/M \rfloor$, where $M = \varepsilon/(n + n^2\delta)\max_v w_v$ and shows that this suffices, heavily exploiting that the instance is δ-bounded. After having rounded the instance, in order to obtain an equivalent instance with weights in $\{0, 1\}$, one can simply replace each vertex of weight $w > 1$ by w copies with weight 1. This reduction now works in polynomial time since the weights are polynomially bounded.

5.2 Reducing δ-Bounded ESP to κ-Segmented ESP

The following lemma can be proven analogously to a lemma of Sitters [40].

Lemma 12 (See [40], Lemma 2.14)**.** *Consider any class of metric spaces $\mathcal{C}$, any class of weights $\mathcal{W}$, and any constant $\delta > 0$. If, for each constant κ, there exists a PTAS for κ-segmented ESP on $\mathcal{C}$ with weights $\mathcal{W}$, then there exists a PTAS for δ-bounded ESP $\mathcal{C}$ with weights $\mathcal{W}$.*

In the proof of the lemma, a similar idea as for the proof of Lemma 10 is used to show that there is a cheap solution that completes before time $O_\varepsilon(1 + \delta)L$, where L is the given delay of the instance. Then, by considering appropriate time points starting at L and growing exponentially with base $(1 + \Theta(\varepsilon))$, one can show that for $\kappa \in O_\varepsilon(\log(1 + \delta))$, an α-approximate solution for the κ-segmented version of the instance can be transformed into the desired $((1+\varepsilon)\alpha)$-approximate solution for the original instance.

5.3 A PTAS for κ-Segmented ESP in the Euclidean Case

Sitters [40] observed that, in Euclidean space, the QPTAS for the traveling repairperson problem [10] (which is based on the well-known PTAS by Arora for TSP [9]) can be turned into a PTAS for the segmented version of the traveling repairperson problem. In this section, we observe that an adapted approach yields a PTAS for Euclidean segmented ESP with weights in $\{0, 1\}$. We focus on the two-dimensional case; an extension to the d-dimensional case for constant d is straightforward.

In the following, familiarity with Arora's PTAS [9] will be helpful but not required.

5.3.1 Setup

The core of our PTAS for segmented ESP is the dynamic-programming procedure. Before this procedure is called, however, there are several preprocessing steps. First, consider a smallest axis-aligned square that contains all weight-1 vertices from the input. We denote it by S_0 and its side length by L_0. Note that L_0 is a lower bound on the cost of an optimal solution. An optimal solution is, however, not necessarily entirely contained in the square as it may use a weight-0 vertex outside the square as a "Steiner" vertex. We therefore enlarge the square from its center by a factor of $3n^2 + 1$, yielding a new square S with side length $L = (3n^2 + 1)L_0$. The scaling factor is chosen such that all points whose distance from S_0 is at least $\sqrt{2}n^2L_0$ are included. Note that there exists an optimal solution that is entirely contained in S because a trivial upper bound on the cost of the optimal solution of $\sqrt{2}n^2L_0$ can be obtained by connecting all weight-1 vertices to r. We can therefore ignore all input points outside S.

Round the instance. We place a grid of granularity $\Theta(\varepsilon L/n^4)$ within S and move each input point to a closest grid point. Note that, this way, several input points may end up at the same location. In the same way as in the literature [10], any solution for the rounded instance can be turned into a solution for the original instance at a cost of $O(\varepsilon)$OPT in the objective-function value: The additional cost of $O(\varepsilon L/n^3)$ per vertex can be charged to the objective as it is $\Omega(L/n^2)$ by construction of S.

Build random quadtree. We first obtain an even larger square from S by enlarging it by an additional factor of 2 from its center and then shifting it to the left by a value a chosen uniformly at random from $\{-L/2, -L/2+1, \dots, L/2-1, L/2\}$ and to the top by a value b chosen uniformly at random from the same set, independently from a. Note that, in any event, the resulting square S' contains S.

We partition S' into four equal-sized squares, which are recursively partitioned in the same way until they only contain a single grid point at which there is a vertex (but possibly many vertices). From this partition, a so-called quadtree naturally emerges by identifying each of the squares (also called cells in the following) with a node and making a node a child of another node if its corresponding square is one of the four smaller squares within that node's square. We root the quadtree at the node corresponding to S'. Since the minimum distance between any two vertices not at the same grid point is $\Theta(\varepsilon L/n^4)$ by the rounding step, the quadtree has depth $O(\log L)$.

Derandomization. We remark that the randomization is only for a simpler analysis. Indeed, our algorithm can be derandomized in the same way as Arora's PTAS and its variants: Simply try all, polynomially many, values for the random variables a and b, and output the cheapest solution obtained this way.

5.3.2 Portal-respecting solutions and the structural result

The set of solutions over which the dynamic-programming procedure optimizes are so-called portal-respecting solutions. Such solutions only cross cell boundaries at so-called portals, and they do so only a constant number of times at each portal. For each cell, we place $\Theta(\log n/\varepsilon)$ equidistant portals on each side of that cell, from corner to corner and including the corners. Additionally, each cell inherits all portals from all its ancestors in the quadtree. The following structural result states that we only lose a $1 + O(\varepsilon)$ factor when restricting to portal-respecting solutions.

Lemma 13. *With constant probability (over the random placement of the quadtree), there exists a* $(1 + O(\varepsilon))$*-approximate portal-respecting solution.*

The result can be proved precisely in the same way as in [10], by applying Arora's structural result [9] to each segment. In [10], the pathwise version of our problem is considered, but this difference does not affect the proof.

5.3.3 Further setup

Before we describe the dynamic program, we need two additional setup steps.

Additional rounding. Since we guess lengths of parts of the solution, we assume at the loss of another $1 + O(\varepsilon)$ factor that the distance between any two relevant points (i.e., input points or portals) is a polynomially bounded integer. This is possible because the objective-function value is the sum of polynomially many distances.

Guessing of segment lengths. It will be useful to know the completion times $t^{(1)}, \ldots, t^{(\kappa)}$ before running the dynamic-programming procedure. By our rounding procedure, we know that there are only $n^{O(1)}$ options for each of these $O(1)$ lengths, meaning that there are $n^{O(1)}$ combinations of different completion times for each of the segments, which we can all guess.

5.3.4 Dynamic programming

For each cell z of the quadtree we additionally "guess" the following pieces of information relevant for the other quadtree cells (reflected in the fact that there is a DP entry for each combination). Specifically, for each segment $i \in [\kappa]$, we guess

(i) the total length ℓ_i of segment i within the cell,

(ii) the number m_i of times that the segment crosses the boundary of the cell, and for each $j \in [m_i]$ of these crossing a *type* $\tau_{i,j}$ for the j-th such time, containing

- the portal $p_{i,j}$ at which the cell is intersected, and
- whether the segment enters or leaves the cell at $p_{i,j}$.

Note that, again, there are only polynomially many options for each of the parameters (in particular, m_i can be assumed to be at most $O(\log n/\varepsilon)$, and we only have constantly many options for the type of each crossing) and therefore only polynomially many DP entries.

Any DP entry $\mathrm{DP}[z, (\ell_i, (\tau_{i,j})_{j\in[m_i]})_{i\in[\kappa]}]$ is supposed to contain the cost of the cheapest portal-respecting solution restricted to the corresponding cell obeying the constraints imposed by the guessed parameters and visiting all vertices within the cell. Note that such a solution may not exist (e.g., the cell does not contain the root but some other vertices, and no segment ever enters the cell), in which case the cost is ∞. Otherwise, the cost of a solution restricted to a cell refers to the sum over all vertices in that cell of the completion time of the segment that they are visited in.

With this definition, the entry $\mathrm{DP}[z_0, (t^{(i)} - \sum_{i'<i} t^{(i')}, ())_{i\in[\kappa]}]$ is supposed to contain the cost of the optimal portal-respecting solution, where z_0 is the root of the quadtree and $()$ is the empty tuple. By standard techniques, the actual solution can be recovered from these entries.

The DP entries can be filled in a bottom-up manner. Indeed, an entry $\mathrm{DP}[z, (\ell_i, (\tau_{i,j})_{j\in[m_i]})_{i\in[\kappa]}]$ where z is a leaf of the quadtree can be easily computed: If the cell does not contain the root but at least one other vertex (recall that all vertices are then located at a common point), but there is no incoming crossing, save ∞ in the DP entry. Otherwise, guess which segment i_0 and which incoming crossing $j_0 \in [m_i]$ connects to the vertices. If the cell contains a vertex or the root (in which case we let $i_0 \coloneqq 0$), guess which of the outgoing crossings $j > j_0$ of the segment i_0 and which outgoing crossing from later segments connect to the vertex. Then guess a one-to-one correspondence between remaining incoming and outgoing crossings of each segment such that each incoming crossing is connected to an outgoing crossing with a later index, and connect the corresponding portals accordingly. If no such correspondence exists for some segment (e.g., because the number of remaining outgoing and incoming portals is different), discard this guess. Likewise, the guess is also discarded if, for some $i \in [\kappa]$, the resulting length of segment i within z is not ℓ_i. Among the remaining options, save the lowest cost in the DP entry. Again, if there is no such solution, the cost is ∞. Note that we are only considering polynomially many combinations of guesses.

To compute a DP entry $\mathrm{DP}[z, (\ell_i, (\tau_{i,j})_{j\in[m_i]})_{i\in[\kappa]}]$ for a non-leaf z, we use previously computed entries for z's children z^{TL} (top left), z^{TR} (top right), z^{BL} (bottom left), and z^{BR} (bottom right). More specifically, we first guess, for each segment $i \in [\kappa]$, nonnegative integers ℓ_i^{TL}, ℓ_i^{TR}, ℓ_i^{BL}, and ℓ_i^{BR} such that $\ell_i = \ell_i^{\mathrm{TL}} + \ell_i^{\mathrm{TR}} + \ell_i^{\mathrm{BL}} + \ell_i^{\mathrm{BR}}$. Regarding the crossings, note that $(\tau_{i,j})_{i\in\kappa, j\in[m_i]}$ already dictate the crossings (and their types) of the sides of the children cells that coincide with the sides of z (outer boundaries) but not the crossings of the other sides of the children cells (inner boundaries). We therefore guess the crossings of the inner boundaries and their types, and we guess in which order all these crossings happen, in a way that is consistent with the given order on the crossings of the outer boundaries. These guesses induce the crossings $(\tau_{i,j}^{\mathrm{TL}})_{i\in\kappa, j\in[m_i^{\mathrm{TL}}]}$,$(\tau_{i,j}^{\mathrm{TR}})_{i\in\kappa, j\in[m_i^{\mathrm{TR}}]}$,$(\tau_{i,j}^{\mathrm{BL}})_{i\in\kappa, j\in[m_i^{\mathrm{BL}}]}$, and $(\tau_{i,j}^{\mathrm{BR}})_{i\in\kappa, j\in[m_i^{\mathrm{BR}}]}$ for the children cells. Note that a single guessed crossing of an inner boundary induces two crossings (one outgoing, one incoming) for the children cells. In the DP entry $\mathrm{DP}[z, (\ell_i, (\tau_{i,j})_{j\in[m_i]})_{i\in[\kappa]}]$, we save the minimum value of

$$\begin{aligned}\mathrm{DP}[z^{\mathrm{TL}}, (\ell_i^{\mathrm{TL}}, (\tau_{i,j}^{\mathrm{TL}})_{j\in[m_i^{\mathrm{TL}}]})_{i\in[\kappa]}] + \mathrm{DP}[z^{\mathrm{TR}}, (\ell_i^{\mathrm{TR}}, (\tau_{i,j}^{\mathrm{TR}})_{j\in[m_i^{\mathrm{TR}}]})_{i\in[\kappa]}] \\ + \mathrm{DP}[z^{\mathrm{BL}}, (\ell_i^{\mathrm{BL}}, (\tau_{i,j}^{\mathrm{BL}})_{j\in[m_i^{\mathrm{BL}}]})_{i\in[\kappa]}] + \mathrm{DP}[z^{\mathrm{BR}}, (\ell_i^{\mathrm{BR}}, (\tau_{i,j}^{\mathrm{BR}})_{j\in[m_i^{\mathrm{BR}}]})_{i\in[\kappa]}]\end{aligned}$$

obtained this way. Again, note that we are only considering polynomially many combinations of guesses.

6 Hardness of approximation

In this section we prove the following theorem.

Theorem 4. *There exists some constant $\varepsilon > 0$ such that there is no polynomial-time $(1+\varepsilon)$-approximation algorithm for the expanding search problem, unless* $\mathsf{P} = \mathsf{NP}$.

The hardness result for ESP follows by a reduction from a variant of the Steiner tree problem which is defined as follows. Given a graph $G = (V, E)$ with non-negative edge costs and a set $T \subseteq V$ of vertices, the so-called terminals, the Steiner tree problem on graphs asks for a minimum-cost tree that is a subgraph of G and that contains all vertices in T. The variant that we consider and use is the so-called SteinerTree(1,2), short ST(1,2), where G is a complete graph and all edge costs are either 1 or 2. Bern and Plassmann [15] showed the following theorem.

Theorem 5 (Theorem 4.2 in [15])**.** SteinerTree(1,2) *is* MaxSNP*-hard.*

It was shown in [11] that there exists no polynomial-time approximation scheme for any MaxSNP-hard problem, unless $\mathsf{P} = \mathsf{NP}$. Hence, there exists some constant $\rho > 0$ such that there is no polynomial-time $(1+\rho)$-approximation algorithm for ST(1,2), unless $\mathsf{P} = \mathsf{NP}$. We use this to show the hardness result for ESP.

The proof of Theorem 4 is organized as follows. Given a β-approximation algorithm Alg' for the expanding search problem for any $\beta > 1$ we construct a γ-approximation algorithm Alg for ST(1,2) with $\gamma < 1+\rho$. Due to the approximation hardness of ST(1,2) this contradicts the existence of a β-approximation algorithm Alg' for the expanding search problem for any $\beta > 1$.

Construct ESP instance. Let $I = (G, T, (c_e)_{e \in E})$ be an instance of ST(1,2) on the undirected complete graph $G = (V, E)$ with terminal set $T \subseteq V$ with $|T| \geq 2$ and edge costs $c_e \in \{1, 2\}$ for all $e \in E$. We construct an instance $I' = (G', (w_v)_{v \in V'}, (c'_e)_{e \in E'}, r)$ of ESP as follows. The graph $G' = (V', E')$ consists of k copies $G_1, \ldots, G_k$ of G where all vertices are connected to an additional vertex r, i.e., the root vertex of the ESP instance. The number k of copies will be determined later. All edges within some copy G_i are assigned the same cost as in the original instance G. Edges incident to r have cost $a = 2(|T| - 1)$. Finally, all vertices that correspond to terminals in the original instance (later also called terminal vertices) have weight $1/|T|$ while all other vertices are assigned weight 0. Note that, by this choice of weights, each copy of G has a total weight of 1. This finishes the construction of the ESP instance I'.

We make the following observation. Any feasible Steiner tree for I consists of at least $|T| - 1$ edges, each having a cost of 1 or 2. Hence, we can lower-bound the cost of an optimal Steiner tree for I by $|T| - 1$. However, since G is a complete graph, simply choosing some spanning tree that only uses edges from $E[T]$ gives an upper bound for the optimal Steiner tree of $2(|T| - 1)$. This yields

$$\textsc{Opt}_{\text{ST}}(I) \leq a \leq 2\textsc{Opt}_{\text{ST}}(I) \ , \tag{5}$$

where $\textsc{Opt}_{\text{ST}}(I)$ denotes the cost of an optimal Steiner tree solution on I.

To show the main result we make some assumptions on the expanding search pattern σ_{ALG} obtained from the β-approximation algorithm $\textsc{Alg}'$ for the expanding search pattern on instance I'. For this manner, we call σ_{ALG} *structured* if each copy G_i is connected to the root by exactly one edge and all edges belonging to some copy G_i or connecting G_i to the root are consecutive within σ_{ALG}. The following lemma states that we can always obtain a structured expanding search pattern from the β-approximation σ_{ALG} that maintains the approximation ratio.

Lemma 14. *Given an expanding search pattern σ_{ALG}, a structured search pattern σ'_{ALG} can be computed in polynomial time such that $L(\sigma'_{\text{ALG}}) \leq L(\sigma_{\text{ALG}})$.*

If σ_{ALG} is structured, there is nothing left to show, so we may assume that σ_{ALG} is not structured. First, suppose that there is some copy G_i that is connected to the root by more than one edge. Let $e = (r, v)$ be the first edge that connects G_i to the root in σ_{ALG}. Then we can replace any other edge (r, w) with $w \in V[G_i]$ by the edge (v, w) of cost at most $2 \leq a$.

Hence, if σ_{ALG} is not structured, we may assume that this is due to the existence of some copy G_i such that not all edges belonging to G_i or connecting G_i to the root are consecutive within σ_{ALG}. We write σ_{ALG} as a concatenation of (consecutive) subsequences

$$\sigma_{\text{ALG}} = \sigma_1 + \sigma_2 + \sigma_3 + \cdots + \sigma_{2s} + \sigma_{2s+1},$$

for some $s > 1$ such that the subsequences with even index $\sigma_2, \sigma_4, \ldots, \sigma_{2s}$ are the inclusion-wise maximal subsequences of σ_{ALG} consisting only of edges belonging to G_i or connecting G_i to the root in the order as they appear in σ_{ALG}. The subsequences $\sigma_1, \sigma_3, \ldots, \sigma_{2s+1}$ with odd index are the inclusion-wise maximal subsequences of the remaining edges where we allow that σ_1 and σ_{2s+1} are empty, but all other subsequences with odd index are non-empty. For some subsequence $\hat{\sigma}$ of σ_{ALG} we denote by $\ell(\hat{\sigma}) = \sum_{e \in \hat{\sigma}} \ell_e$ the length of that subsequence and by $t(\hat{\sigma})$ the number of terminal vertices that $\hat{\sigma}$ connects to the rooted tree that has already been explored by previous subsequences. Then we can define the *ratio* of a subsequence as $r(\hat{\sigma}) \coloneqq \ell(\hat{\sigma})/t(\hat{\sigma})$.

Claim 1. *We can assume, without loss of generality, that $t(\sigma_j) \geq 1$ for all $j \in \{2, 3, \ldots, 2s\}$.*

Proof of the claim. Assume there exists some $j \in \{2, 3, \ldots, 2s\}$ such that $t(\sigma_j) = 0$. Then, we can swap the positions of σ_j and σ_{j+1} and continue with the newly obtained expanding search sequence with fewer subsequences. By this we only improve the total latency since no exploration of any terminal vertex is postponed. ◁

Hence, the ratio $r(\sigma_i)$ is well defined for all $i \in \{2, 3, \ldots, 2s\}$.

Claim 2. *We have $r(\sigma_2) \geq 2$.*

Proof of the claim. This follows from the very first edge in σ_2 having length $a = 2(|T|-1)$ and Claim 1 saying that there exists at least one more subsequence σ_{2l} for some $l > 1$ that connects at least one terminal vertex. Therefore, σ_2 can connect at most $|T| - 1$ terminal vertices for which at least $|T| - 1$ many edges are needed. Each of those edges has a length of $1, 2$, or a, where precisely one edge has length a. Hence, the minimum possible ratio for σ_1 is

$$\frac{a + (|T| - 2)}{|T| - 1} = 3 - \frac{1}{|T| - 1} \geq 2$$

using $|T| \geq 2$. ◁

Claim 3. *We can assume, without loss of generality, that $r(\sigma_{2s}) \leq 2$.*

Proof of the claim. Assume that this were not the case, so $r(\sigma_{2s}) > 2$. Then, we can do the following changes to decrease that ratio without worsening the objective value of the initial expanding search pattern σ_{ALG}. Let $\sigma_{2s} = (e_1, \ldots, e_m)$. If e_m does not connect a terminal vertex we can simply remove e_m. With $\ell_{e_m} > 0$ this decreases the ratio $r(\sigma_{2s})$ and only decreases the latencies of any terminal vertices connected by edges in σ_{2s+1}. Hence, we may assume that e_m connects a terminal vertex. Now choose the shortest subsequence $\bar{\sigma} = (e_p, \ldots, e_{m-1}, e_m)$ with $p \in \{1, \ldots, m\}$ such that $r(\bar{\sigma}) > 2$. This is well-defined since $r(\sigma_{2s}) > 2$. Let $\{e_1^*, \ldots, e_q^* = e_m\}$ be the set of edges that connect a terminal vertex in the order as they appear in $\bar{\sigma}$. We now claim that for any subsequence $\bar{\sigma}_j = (e_p, e_{p+1}, \ldots, e_j^*)$ with $j \in \{1, \ldots, q\}$ it holds that $r(\bar{\sigma}_j) > 2$. Assume for contradiction that there exists some $j \in \{1, \ldots, q\}$ such that $r(\bar{\sigma}_j) \leq 2$. Then let $\bar{\sigma}_{-j}$ be such that $\bar{\sigma}$ is a concatenation of $\bar{\sigma}_j$ and $\bar{\sigma}_{-j}$. However, since $\bar{\sigma}$ is the shortest contiguous subsequence of σ_{2s} that contains e_m such that $r(\bar{\sigma}) > 2$ holds, it follows that $r(\bar{\sigma}_{-j}) \leq 2$, otherwise $\bar{\sigma}$ would not be not minimal. In total this yields

$$2 < r(\bar{\sigma}) = \frac{\ell(\bar{\sigma})}{t(\bar{\sigma})} = \frac{\ell(\bar{\sigma}_j) + \ell(\bar{\sigma}_{-j})}{t(\bar{\sigma}_j) + t(\bar{\sigma}_{-j})} \leq \frac{2t(\bar{\sigma}_j) + 2t(\bar{\sigma}_{-j})}{t(\bar{\sigma}_j) + t(\bar{\sigma}_{-j})} = 2,$$

a contradiction. Hence, for any subsequence $\bar{\sigma}_j = (e_p, \ldots, e_j^*)$ with $j \in \{1, \ldots, q\}$ it holds that $r(\bar{\sigma}_j) > 2$. Let Q denote the set of terminal vertices which are connected by $\bar{\sigma}$. It follows, that then we can remove all edges in $\bar{\sigma}$ and instead connect each vertex in Q with a single edge of length 2 in the same order as they had been connected in $\bar{\sigma}$. This strictly decreases the latency of all vertices in Q and in particular the latency of that terminal vertex, originally connected by

e_m. Hence, also all terminal vertices connected by some edges in σ_{2s+1} will have a strictly smaller latency. Additionally, this strictly decreases the ratio $r(\sigma_{2s})$, so this procedure can be repeated until $r(\sigma_{2s}) \leq 2$ as there are only finitely many values that this ration can take. $\triangleleft$

With those three claims at hand we are now ready to prove Lemma 14.

Proof of Lemma 14. Consider the ratios $r(\sigma_{2s-1})$ and $r(\sigma_{2s})$ and distinguish two cases. First, assume $r(\sigma_{2s-1}) \geq r(\sigma_{2s})$. We then claim that swapping those two subsequences decreases the total latency. To this end, note that $r(\sigma_{2s-1}) \geq r(\sigma_{2s})$ yields

$$\frac{\ell(\sigma_{2s-1})}{t(\sigma_{2s-1})} \geq \frac{\ell(\sigma_{2s})}{t(\sigma_{2s})} \qquad \Leftrightarrow \qquad \ell(\sigma_{2s-1})t(\sigma_{2s}) \geq \ell(\sigma_{2s})t(\sigma_{2s-1}). \tag{6}$$

Swapping the positions of σ_{2s-1} and σ_{2s} causes the latency of $t(\sigma_{2s})$ many terminal vertices to decrease by $\ell(\sigma_{2s-1})$ while the latency of $t(\sigma_{2s-1})$ many terminal vertices increase by $\ell(\sigma_{2s})$. Note that the latencies of all terminal vertices connected by other subsequences remain unaffected. Hence, by Equation (6) the total latency of the expanding search pattern cannot increase. For the second case we assume $r(\sigma_{2s-1}) < r(\sigma_{2s})$ which yields $r(\sigma_{2s-1}) < 2$. We then compare $r(\sigma_{2s-2})$ to $r(\sigma_{2s-1})$ and continue recursively with adjacent subsequences until we find the first pair σ_j and σ_{j-1} with $j \in \{2, 3, \ldots, 2s-2\}$ such that $r(\sigma_j) \geq r(\sigma_{j-1})$. This pair does exist since $r(\sigma_{2s-1}) < 2$ and $r(\sigma_2) \geq 2$. If this pair is found, those two subsequences can swap positions and with a computation analogous to the one in the first case the total latency does not increase. After swapping at most s pairs of subsequences in the same way, the desired property is established for G_i.

This entire process can be repeated for each other copy of G until the obtained expanding search sequence is structured. Computing the ratios and performing the swaps of the subsequences takes time polynomial in the length of the sequence and, hence, this procedure is polynomial. $\square$

With Lemma 14 at hand, we will assume from now on that σ_{ALG} is structured and that it visits the k copies of G in the order $G_1, \ldots, G_k$. Next, we define the algorithm ALG that uses the β-approximation algorithm for the expanding search problem to obtain a solution for the original ST(1,2) instance I.

Construct algorithm for ST(1,2). The algorithm ALG is defined as follows. It takes an ST(1,2) instance I as input and constructs the corresponding ESP instance I'. Afterwards it runs the β-approximation algorithm ALG$'$ on the instance I' and obtains the expanding search pattern σ_{ALG}. Given σ_{ALG}, ALG computes a corresponding structured expanding search pattern σ'_{ALG}; this can be done in polynomial time as shown in Lemma 14. For each copy G_i, ALG$'$ computes as a byproduct a Steiner tree solution T_i for the original instance I. Out of these k solutions, ALG chooses the one with minimum cost $T^* = \arg\min_{i \in \{1,\ldots,k\}} \left\{\sum_{e \in E[T_i]} c_e\right\}$ as its output. Hence, $\text{ALG}(I) = c(T^*)$, where $c(T^*) \coloneqq \sum_{e \in E[T^*]} c_e$.

In the remainder of this section we will denote the cost of the algorithms $\textsc{Alg}$ and $\textsc{Alg}'$ on instances I and I' by $\textsc{Alg}(I)$ and $\textsc{Alg}'(I')$. Further, we will denote the cost of the optimal algorithms for ST(1,2) and the expanding search problem on instances I and I' by $\textsc{Opt}_{\text{ST}}(I)$ and $\textsc{Opt}_{\text{ESP}}(I')$, respectively.

First, we show how to obtain the upper bound on $\textsc{Alg}(I)$. For this manner, let $T_1, \ldots, T_k$ be the Steiner tree solutions that $\textsc{Alg}'$ explores on the k copies $G_1, \ldots, G_k$ of instance I'. The upper bound is now obtained by assuming that the expanding search pattern $\sigma_{\textsc{Alg}}$ collects the total weight of 1 of each copy G_i when visiting the very first vertex of that copy. This yields

$$\begin{aligned}\textsc{Alg}'(I') &\geq \sum_{i=1}^{k} \left(a + \sum_{j=1}^{i-1} c(T_j) \right) \\ &\geq \sum_{i=1}^{k} \left(a + \sum_{j=1}^{i-1} c(T^*) \right) \\ &= \sum_{i=1}^{k} \frac{k(k+1)}{2} a + \frac{(k-1)k}{2} c(T^*)\end{aligned}$$

which is equivalent to

$$\textsc{Alg}(I) = c(T^*) \leq \frac{2}{(k-1)k} \left(\textsc{Alg}'(I') - \frac{k(k+1)}{2} a \right) . \tag{7}$$

Next we give an upper bound on $\textsc{Opt}_{\text{ESP}}(I')$. For this manner, assume we have an optimal Steiner tree solution for the instance I. Using this optimal solution, we obtain an ESP solution by first visiting an arbitrary vertex v in G_1, then exploring all edges of the optimal Steiner tree solution of I, and afterwards continuing in the same manner for the remaining copies $G_2, \ldots, G_k$. By assuming that the total weight of 1 of each copy is only collected when visiting the very last terminal vertex of each copy we obtain the following upper bound on $\textsc{Opt}_{\text{ESP}}(I')$

$$\begin{aligned}\textsc{Opt}_{\text{ESP}}(I') &\leq \sum_{i=1}^{k} i(a + \textsc{Opt}_{\text{ST}}(I)) \\ &= \frac{k(k+1)}{2}(a + \textsc{Opt}_{\text{ST}}(I)) \ .\end{aligned} \tag{8}$$

Now, combining Equations (7) and (8) with $\textsc{Alg}'(I') \leq \beta \textsc{Opt}_{\text{ESP}}(I')$ yields

$$
\begin{aligned}
\textsc{Alg}(I) &\leq \frac{2}{(k-1)k}\left(\textsc{Alg}'(I') - \frac{k(k+1)}{2}a\right) \\
&\leq \frac{2}{(k-1)k}\left(\beta\textsc{Opt}_{\mathrm{ESP}}(I') - \frac{k(k+1)}{2}a\right) \\
&\leq \frac{2}{(k-1)k}\left(\beta\left(\frac{k(k+1)}{2}\big(a + \textsc{Opt}_{\mathrm{ST}}(I)\big)\right) - \frac{k(k+1)}{2}a\right) \\
&= \frac{k+1}{k-1}\left(a(\beta-1) + \beta\textsc{Opt}_{\mathrm{ST}}(I)\right).
\end{aligned}
$$

With $\beta - 1 > 0$ and Equation (5) we finally obtain

$$
\textsc{Alg}(I) \leq \frac{k+1}{k-1}(3\beta-2)\textsc{Opt}_{\mathrm{ST}}(I). \tag{9}
$$

Thus, using the β-approximation algorithm for the expanding search problem yields a γ-approximation algorithm for ST(1,2) with $\gamma = \frac{k+1}{k-1}(3\beta - 2)$. However, by choosing β and k such that $\beta < 1 + \rho/3$ and $k > (\rho + 3\beta - 1)/(\rho - 3\beta + 3)$ we have $\gamma < (1 + \rho)$, a contradiction to the approximation hardness of ST(1,2). This proves that there exists some constant $\varepsilon > 0$ such that there is no polynomial-time $(1 + \varepsilon)$-approximation algorithm for ESP, unless $\mathsf{P} = \mathsf{NP}$, which finishes the proof of Theorem 4.

Acknowledgements. We thank Spyros Angelopoulos for fruitful discussions and pointers to earlier literature.

References

[1] F. N. Afrati, S. S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou. The complexity of the travelling repairman problem. *RAIRO – Theoretical Informatics and Applications*, 20(1):79–87, 1986.

[2] S. Alpern, R. Fokkink, L. Gasieniec, R. Lindelauf, and V. S. Subrahmanian. *Search Theory*. Springer, New York, 2013.

[3] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*, volume 55 of *International Series in Operations Research and Management Science*. Kluwer, 2003.

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

[5] S. Alpern and T. Lidbetter. Approximate solutions for expanding search games on general networks. *Annals of Opererations Research*, 275(2):259–279, 2019.

[6] S. Angelopoulos, C. Dürr, and T. Lidbetter. The expanding search ratio of a graph. *Discrete Applied Mathematics*, 260:51–65, 2019.

[7] A. Archer and A. 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.

[8] A. Archer, A. Levin, and D. P. Williamson. A faster, better approximation algorithm for the minimum latency problem. *SIAM Journal on Computing*, 37(5):1472–1498, 2008.

[9] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

[10] S. Arora and G. Karakostas. Approximation schemes for minimum latency problems. *SIAM Journal on Computing*, 32(5):1317–1337, 2003.

[11] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[12] S. Arya and H. Ramesh. A 2.5-factor approximation algorithm for the k-MST problem. *Information Processing Letters*, 65:117–118, 1998.

[13] I. Averbakh. Emergency path restoration problems. *Discrete Optimization*, 9(1):58–64, 2012.

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

[15] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989.

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

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

[18] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, volume 3122 of *Lecture Notes in Computer Science*, pages 72–83, 2004.

[19] T. Dewilde, D. Cattrysse, S. Coene, F. C. R. Spieksma, and P. Vansteenwegen. Heuristics for the traveling repairman problem with profits. *Computers & Operations Research*, 40:1700–1707, 2013.

[20] J. Fakcharoenphol, C. Harrelson, and S. Rao. The k-traveling repairmen problem. *ACM Transactions on Algorithms*, 3(4):40–es, 2007.

[21] M. Fischetti, G. Laporte, and S. Martello. The delivery man problem and cumulative matroids. *Operations Research*, 41:1010–1176, 1993.

[22] Z. Friggstad, M. R. Salavatipour, and Z. Svitkina. Asymmetric traveling salesman path and directed latency problems. *SIAM Journal on Computing*, 42:1596–1619, 2013.

[23] S. Gal. Search games with mobile and immobile hider. *SIAM Journal on Control and Optimization*, 17:99–122, 1979.

[24] S. Gal. *Search Games*. Academic Press, New York, 1980.

[25] A. García, P. Jodrá, and J. Tejel. A note on the travelling repairman problem. *Networks*, 40:27–31, 2002.

[26] N. 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.

[27] M. X. Goemans and J. M. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82:111–124, 1998.

[28] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[29] B. Hermans, R. Leus, and J. Matuschke. Exact and approximation algorithms for the expanding search problem. *INFORMS Journal on Computing*, 34(1):281–296, 2022.

[30] R. Isaacs. *Differential Games*. John Wiley and Sons, New York, 1965.

[31] D. S. Johnson, M. Minkoff, and S. Philipps. The prize collecting Steiner tree problem: Theory and practice. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 760–769, 2000.

[32] E. Koutsoupias, C. H. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 280–289, 1996.

[33] S. Li and S. Huang. Multiple searchers searching for a randomly distributed immobile target on a unit network. *Networks*, 71(1):60–80, 2018.

[34] E. Minieka. The delivery man problem on a tree network. *Annals of Operations Research*, 18:261–266, 1989.

[35] V. Nagarajan and R. Ravi. The directed minimum latency problem. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 193–206. Springer, 2008.

[36] S. O.Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie. News from the online traveling repairman. *Theoretical Computer Science*, 295:279–294, 2003.

[37] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees short or small. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 546–555, 1994.

[38] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

[39] R. 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.

[40] R. Sitters. Polynomial time approximation schemes for the traveling repairman and other minimum latency problems. *SIAM Journal on Computing*, 50(5):1580–1602, 2021.

[41] Y. Tan, F. Qiu, A. K. Das, D. S. Kirschen, P. Arabshahi, and J. Wang. Scheduling post-disaster repairs in electricity distribution networks. *IEEE Trans. Power Syst.*, 34(4):2611–2621, 2019.

[42] K. E. Trummel and J. R. Weisinger. Technical note – The complexity of the optimal searcher path problem. *Opererations Research*, 34(2):324–327, 1986.

[43] J. N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22:262–283, 1992.