Improved Approximation Algorithms for the Expanding Search Problem

Svenja M. Griesbach
Institute for Mathematics, Technische Universität Berlin, Germany

Felix Hommelsheim
Faculty of Mathematics and Computer Science, Universität Bremen, Germany

Max Klimm
Institute for Mathematics, Technische Universität Berlin, Germany

Kevin Schewior
Department Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark

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 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 general case, we give a $(2e + \varepsilon)$-approximation for any $\varepsilon > 0$. For the case that all vertices have unit weight, we provide a 2e-approximation. Finally, we provide a PTAS for the case of a Euclidean graph. Previously, for all cases only an 8-approximation was known.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

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

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.54

Related Version *Full Version*: `https://arxiv.org/abs/2301.03638` [28]

Funding *Svenja M. Griesbach*: Supported by Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy, Berlin Mathematics Research Center (grant EXC-2046/1, Project 39068689) and HYPATIA.SCIENCE (Department of Mathematics and Computer Science, University of Cologne).
Max Klimm: Supported by Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy, Berlin Mathematics Research Center (grant EXC-2046/1, Project 390685689).
Kevin Schewior: Supported in part by the Independent Research Fund Denmark, Natural Sciences (grant DFF-0135-00018B).

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

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 with the time needed for rescue; see also the discussion in Averbakh and Pereira [13]. 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. 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 polynomial-time approximation algorithms with improved approximation guarantees for ESP. We first give an approximation algorithm for the general case with arbitrary vertex weights with an approximation guarantee of $2\mathrm{e} + \varepsilon$ for any $\varepsilon > 0$ where $2\mathrm{e} \approx 5.44$ (Theorem 1). For the unweighted case where all vertices have the same weight, we provide an approximation algorithm with an approximation guarantee of $2\mathrm{e}$ (Theorem 6). 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 [16, 26]; 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 where vertices have non-negative weights and, given $q \in \mathbb{N}$, the task is to find a length-minimal tree with vertex weight at least q. Johnson et al. [31] noted that any approximation algorithm for k-MST that relies on the Goemans–Williamson algorithm [27] for the prize-collecting Steiner-tree problem can be turned into an approximation algorithm for the quota version with the same approximation guarantee. We follow this line of reasoning and show that also the approximation algorithm of Garg [25] that relies on a modified version of the Goemans–Williamson algorithm can be turned into a 2-approximation algorithm for the quota version of the problem. Relying on this result, we 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 7). For this result, we use a decomposition approach by Sitters [40] for the pathwise search problem that relies on partitioning an instance into subinstances. A central challenge when adapting this approach to the expanding search setting is that, unlike pathwise search, expanding search is not memoryless as points contained in one subinstance may be used as Steiner points in another subinstance. We address this difficulty by keeping such points in the subinstance with zero weight so that the partitions become overlapping. To obtain a PTAS for the subproblem, we adapt techniques developed by Arora [9] for Euclidean TSP. As a byproduct, we obtain a $(1+\varepsilon)$-approximate reduction from general ESP to unweighted ESP, implying an alternative $(2\mathrm{e}+\varepsilon)$-aproximation which may be of independent interest.

For all variants considered in this paper, i.e., the unweighted case, the weighted case, and the Euclidean case, the best approximation algorithm was an 8-approximation due to Hermans et al. [29].

Finally, we show that there is no PTAS for ESP unless $\mathsf{P} = \mathsf{NP}$ (Theorem 14). The proof follows a similar idea as the hardness proof for the traveling repairperson problem suggested in [15]. However, in comparison to the pathwise search, in our setting the solution needs to be structured. Showing this property turns out to be rather elaborate. Previously, it was only known that the expanding search problem is NP-hard [13]. Due to space constraints, we defer some proofs and more detailed descriptions to the full version of this paper [28].

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 gave an exact 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. [24]. Minieka [36] proposed an exact 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. [15] who gave a 144-approximation. After a series of improvements [7, 8, 10, 26, 33], the best factor so far is a 3.59-approximation for general metrics [16], 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 other settings with directed edges [20, 21, 37], with processing times and time windows [43], with profits at vertices [18], with multiple searchers [17, 19, 35], and online variants [34]. 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 [33]. 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 [13]. Alpern and Lidbetter [4] introduced a polynomial-time algorithm for the case when the graph is a tree and gave heuristics for general graphs. Averbakh et al. [12] 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]. Angelopoulos et al. [6] studied the expanding search ratio of a graph. This value is defined as the minimum over all expanding searches of the maximum ratio of the time to reach a vertex by the search algorithm and the time to reach the same vertex by a shortest path. Angelopoulos et al. showed that this ratio is NP-hard to compute and gave a search strategy that achieves a $(4 \ln 4)$-approximation of the optimum.

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 [22] 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, 23, 30, 32].

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 $w_v \in \mathbb{N} = \{0, 1, 2, \dots\}$, and we denote by $V^* \subseteq V$ the set of vertices with $w_v > 0$. Every edge $e \in E$ has a length $\ell_e \in \mathbb{N}$. We use $\mathbb{N}_{>0}$ when we refer to $\mathbb{N} \setminus \{0\}$. 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 distinct edges $\sigma = (e_1, \dots, e_m)$ for some $m \le n - 1$ 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 $k_v(\sigma) = \inf\{i \in \{1, \dots, m\} : v \in e_i\}$ 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. They may, however, be used as Steiner vertices in the constructed search trees and, hence, cannot be contracted as in the pathwise search problem. 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 weighted case

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

▶ **Theorem 1.** *For every $\varepsilon > 0$, there is a polynomial-time $(2\mathrm{e} + \varepsilon)$-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 $q \in [0, W]$, 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 2-approximation. The proof can be found in the full version [28].

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

To approximate ESP, fix $\varepsilon > 0$. For notational convenience, we show an approximation algorithm with guarantee $2(1 + \varepsilon)\mathrm{e}$. We solve the quota problem for quotas $W - W(1 + \varepsilon)^{-i}$ for all $i \in \{0, \dots, \omega\}$, where we let $\omega := \lceil \frac{\log W}{\log(1+\varepsilon)} \rceil$. 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 $T_0, T_1, \ldots, T_\omega$. 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 $V_H = \{0, \ldots, \omega\}$ 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 $P = (n_0, \ldots, n_l)$ with $n_0 = 0$ and $n_l = \omega$ for some $l \in \mathbb{N}$. We construct from this path an expanding search pattern with l phases. In phase $j \in \{1, \ldots, l\}$, 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 the prize-collecting Steiner tree problem with quota $q = W - W(1+\varepsilon)^{-i}$ with the 2-approximation algorithm of Lemma 2 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. Let $q \in [0, W]$ be arbitrary and let $j(q) \in \{1, \ldots, l\}$ be such that $W - W(1+\varepsilon)^{-n_{j(q)-1}} \leq q < W - W(1+\varepsilon)^{-n_{j(q)}}$. Then we define $\pi(q) := \sum_{i=0}^{j(q)} \ell(T_{n_i})$. By $L_q(\sigma_{\text{ALG}})$ we denote the latency of quota q in $\sigma_{\text{ALG}} = (e_1, ..., e_m)$, i.e., the sum of the edge cost of the shortest subsequence $(e_1, ..., e_k)$ of σ_{ALG}, such that the tree spanned by $(e_1, ..., e_k)$ has weight at least q. The following lemma gives an upper bound on the latency for each quota. Its proof uses the intuitive argument that the worst case for the latency is obtained when all trees are nested and exploration takes place only at the end of each tree. The formal proof can be found in the full version [28].

▶ **Lemma 3.** *The latency of quota $q \in [0, W]$ in σ_{ALG} can be bounded by $L_q(\sigma_{\text{ALG}}) \leq \pi(q)$.*

We can now give an upper bound on $L(\sigma_{\text{ALG}})$. The proof relies on the specific way how the length of the arcs in H are defined. Its proof can be found in the full version [28].

▶ **Lemma 4.** *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.*

The technically most challenging part of the analysis of the algorithm is to bound the cost of a shortest path in relation to the total latency of the optimal expanding search pattern. To this end, we use a probabilistic argument where a distribution over paths in H corresponding to the exploration of trees with exponentially increasing weight is considered.

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

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 the rooted prize-collecting Steiner tree problem 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 $2(1+\varepsilon)\mathrm{e}L^*$, 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 $\tilde{m} \in \mathbb{N}$ to be the smallest number such that $\lambda^*(W) \le b\gamma^{\tilde{m}}$. For $j \in \{0,\dots,\tilde{m}\}$, let $\tilde{n}_j := \max\bigl\{k \in \{0,\dots,\omega\} : \ell(T_k) \le 2b\gamma^j\bigr\}$. These values are well-defined as $\ell(T_0) = 0$. Note that the sequence $\tilde{n}_0, \tilde{n}_1, \dots, \tilde{n}_{\tilde{m}}$ is non-decreasing. We denote by $n_0, n_1, \dots, n_m$ the longest increasing subsequence of $\tilde{n}_0, \tilde{n}_1, \dots, \tilde{n}_{\tilde{m}}$. 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 $W - W(1+\varepsilon)^{-i} \le q < W - W(1+\varepsilon)^{-(i+1)}$. 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 $q \in [0,W]$ 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: $d \le b$. Since $\lambda^*(q) = \lambda^*(W-\bar{q}) = d\gamma^j$, there is a tree of cost $d\gamma^j$ that contains the root and explores a total weight of at least $W-\bar{q}$. When computing the 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 $\ell(T_i) \le 2\lambda^*\bigl(W - W(1+\varepsilon)^{-i}\bigr) \le 2\lambda^*(W-\bar{q}) = 2d\gamma^j$. The first inequality is obtained by using the 2-approximation and the second inequality from λ^* being non-decreasing. Since $\ell(T_i) \le 2d\gamma^j \le 2b\gamma^j$, we have that $n_j \ge i$. Using that $\pi(q)$ is non-decreasing, that $W - W(1+\varepsilon)^{-i} \ge W - (1+\varepsilon)\bar{q}$, and that $\gamma > 1$, we obtain

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

Second case: $d > b$. Analogously to the first case we obtain $\ell(T_i) \le 2d\gamma^j$. However, with $1 \le b$ and $d < \gamma$ we have $d < b\gamma$ which yields $\ell(T_i) \le 2d\gamma^j < 2b\gamma^{j+1}$. Hence, we have that $n_{j+1} \ge i$. Again, using that $\pi(q)$ is non-decreasing, that $W - W(1+\varepsilon)^{-i} \ge W - (1+\varepsilon)\bar{q}$, and that $\gamma > 1$, we obtain

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

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

$$\begin{aligned}\mathbb{E}_U\left[\pi\bigl(W - (1+\varepsilon)\bar{q}\bigr)\right] &\le \int_{\log_\gamma d}^{1} 2b\gamma^j\frac{\gamma}{\gamma-1}\,\mathrm{d}U + \int_{0}^{\log_\gamma d} 2b\gamma^{j+1}\frac{\gamma}{\gamma-1}\,\mathrm{d}U \\ &= 2\gamma^j\frac{\gamma}{\gamma-1}\left[\int_{\log_\gamma d}^{1}\gamma^U\,\mathrm{d}U + \gamma\int_{0}^{\log_\gamma d}\gamma^U\,\mathrm{d}U\right] = 2\gamma^j\frac{\gamma}{\gamma-1}\left[\frac{\gamma-d}{\ln\gamma} + \gamma\frac{d-1}{\ln\gamma}\right] \\ &= 2\gamma^j d\frac{\gamma}{\ln\gamma} = 2\frac{\gamma}{\ln\gamma}\lambda^*(W-\bar{q}).\end{aligned}$$

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 of b and d.

First case: $d \leq b$. 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 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 $\ell(T_\omega) \leq 2\lambda^*(W) = 2d\gamma^j \leq 2b\gamma^j$, which yields $n_j \geq \omega$, and thus, $j = m$. By using $\gamma > 1$, we obtain

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

Second case: $d > b$. Analogously to the first case, we obtain $\ell(T_\omega) \leq 2d\gamma^j$. However, with $1 \leq b$ and $d < \gamma$ we have $d < b\gamma$ which yields $\ell(T_\omega) \leq 2d\gamma^j < 2b\gamma^{j+1}$. Hence, we have that $n_{j+1} \geq \omega$, i.e., $j \geq m-1$. In any case, by using that $\gamma > 1$, we obtain

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

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

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

The expected cost of the $(0,\omega)$-path $P = (n_0, n_1, \dots, n_m)$ is then given by

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

As $\pi(q)$ is piece-wise constant, we exchange the order of expectation and integral such that

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

where we further used the substitution rule for integrals and the fact that π is non-decreasing. Using (1), we further obtain

$$\begin{aligned}
\mathbb{E}\big[c(P)\big] &\leq \frac{2\gamma(1+\varepsilon)}{\ln\gamma}\left[\int_{W(1+\varepsilon)^{-\omega}}^{W(1+\varepsilon)^{-1}} \lambda^*(W-\bar{q})\,\mathrm{d}\bar{q} + W(1+\varepsilon)^{-\omega}\mathbb{E}\big[\lambda^*(W)\big]\right] \\
&= \frac{2\gamma(1+\varepsilon)}{\ln\gamma}\int_0^{W(1+\varepsilon)^{-1}} \lambda^*(W-\bar{q})\,\mathrm{d}\bar{q},
\end{aligned}$$

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

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

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

Theorem 1 now follows from combining Lemma 4 and Lemma 5.

4 The unweighted case

Compared to the weighted case we do the following adjustments in order to obtain a 2e-approximation. First, instead of using the quota problem of k-MST, for all $k \in \{1,\dots,n\}$ we solve the original k-MST problem with the 2-approximation algorithm of Garg [25] and obtain n trees $T_1,\dots,T_n$. The auxiliary weighted directed graph $H = (V_H, A_H)$ is then defined by $V_H = \{1,2,\dots,n\}$, $A_H = \{(i,j) \in V_H^2 : i < j\}$, and $\ell_{i,j} := (n-i)c(T_j)$. Finally, we compute a shortest $(1,n)$-path $P = (n_0, n_1, \dots, n_l)$ with $n_0 = 1$ and $n_l = n$ in H. For each phase $j \in \{1,\dots,l\}$, we explore all unexplored vertices of $V[T_{n_j}]$ in any feasible order using the edge-set of $E[T_{n_j}]$. The better approximation factor is due to the fact that we save the factor of $(1+\varepsilon)$ since we can compute the k-MSTs for all relevant values of k, whereas before we needed a rounding technique. We then obtain the following result, which we prove in the full version [28].

▶ **Theorem 6.** *There is a polynomial-time* 2e*-approximation algorithm for the unweighted expanding search problem.*

5 The Euclidean case

In this section, we show the following theorem.

▶ **Theorem 7.** *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 $D \geq 0$, and it is guaranteed that there exists a solution visiting all nonzero-weight vertices and completing by time δD, i.e. this solution has length δD (recall definition in Section 2). The objective is minimizing $L'(\sigma) = \sum_{v\in V^*} w_v L'_v(\sigma)$ where $L'_v(\sigma) = D + L_v(\sigma)$. Second, in κ-segmented ESP, the output needs to come with $\kappa+1$ additional numbers $0 = t^{(0)} \leq t^{(1)} \leq \dots \leq t^{(\kappa)}$. For $v \in V$, its rounded search time is then $\bar{L}_v(\sigma) = \inf\{t^{(i)} : 0 \leq i \leq \kappa,\ L_v(\sigma) \leq t^{(i)}\}$, and the objective is minimizing $\bar{L}(\sigma) = \sum_{v\in V^*} w_v \bar{L}_v(\sigma)$.

We assume $0 < \varepsilon \leq 1$ 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 and constants $\alpha > 1, \varepsilon > 0$. There exists a constant δ such that, if there exists a polynomial-time α-approximation algorithm for δ-bounded ESP on $\mathcal{C}$, then there exists an $(\alpha + \varepsilon)$-approximation algorithm 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 polynomial-time α-approximation algorithm for δ-bounded ESP on $\mathcal{C}$, denoted $\textsc{Approx}_{\delta\text{-bd}}$ 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 $(\alpha + O(\varepsilon))$-approximation algorithm.

For some constant β, we need, in addition to $\textsc{Approx}_{\delta\text{-bd}}$, 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 $\alpha + \varepsilon$ in polynomial time. Therefore, we can pick, e.g., the algorithm from Section 3. For notational purposes, we assume $\alpha \neq \beta$.

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 $\textsc{Approx}_{\delta\text{-bd}}$ 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 $t_i := \mathrm{e}^{(i-2)a+b}$ for $i \in [q+1]$, where q is as small as possible such that $L_v(\sigma_\beta) < t_{q+1}$ for all $v \in V$.
   - For $i \in [q]$, let $V_i := \{v \in V : t_i \leq L_v(\sigma_\beta) < t_{i+1}\}$ 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 $({}^{\mathrm{e}^a}/_{\gamma})$-bounded ESP. Indeed, the prefix $\sigma_{\beta,i+1}$ of σ_β visiting U_{i+1} has total length at most $({}^{\mathrm{e}^a}/_{\gamma})\gamma t_i = t_{i+1}$.
3) **Approximate subproblems:** For $i \in [q]$, apply $\textsc{Approx}_{\delta\text{-bd}}$ to I_i to obtain an α-approximation $\sigma_{\alpha,i}$.
4) **Modify:** For each $i \in [q]$, define σ_i to be $\sigma'_{\alpha,i} + \sigma_{\beta,i+1}$ where:
   - $\sigma'_{\alpha,i}$ is the longest prefix of $\sigma_{\alpha,i}$ of length at most $(1 + {}^{\mathrm{e}^a}/_{\varepsilon\gamma})\gamma t_i$.
   - $\sigma_{\beta,i+1}$ 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 $({}^{\mathrm{e}^a}/_{\gamma})$-bounded ESP is only at the loss of a $1 + \varepsilon$ factor in the achievable (total) objective-function value. Formally, we denote by σ^* an optimal solution for I and, for all $i \in [q]$, by σ_i^* an optimal solution for I_i. The proofs of the following two lemmata can be found in the full version [28].

▶ **Lemma 9.** *It holds that $\mathbb{E}\left[\sum_{i\in[q]} L'(\sigma_i^*)\right] \leq (1 + \varepsilon)L(\sigma^*)$.*

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 $L'(\sigma_i) \leq \alpha(1+\varepsilon)L'(\sigma_i^*)$.*

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 the inequality in Lemma 10 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 as shown in the full version [28].

▶ **Lemma 11** (See [40], Lemma 2.10). *Consider any class $\mathcal{C}$ of metric spaces and any constants $\alpha > 0, \delta, \varepsilon > 0$. If there exists a polynomial-time α-approximation algorithm for δ-bounded ESP with weights in $\{0, 1\}$, then there exists a polynomial-time $(\alpha+\varepsilon)$-approximation algorithm for δ-bounded ESP.*

Note that Lemma 10 and 11, together with Theorem 6, yields an alternative polynomial-time $(2\mathrm{e} + \varepsilon)$-approximation algorithm for the general ESP (Theorem 1).

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 constants $\alpha > 1, \delta, \varepsilon > 0$. If, for each constant κ, there exists a polynomial-time α-approximation algorithm for κ-segmented ESP on $\mathcal{C}$ with weights $\mathcal{W}$, then there exists a polynomial-time $(\alpha + \varepsilon)$-approximation algorithm 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 $O_\varepsilon(1 + \delta)D$, where D is the given delay of the instance. Then, by considering appropriate time points starting at D 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. The following description is self-contained up to parts deferred to the full version [28], but familiarity with Arora's PTAS [9] may still be helpful.

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 d_0. Note that d_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 $d=(3n^2+1)d_0$. The scaling factor is chosen such that all points whose distance from S_0 is at least $\sqrt{2}n^2d_0$ 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 $\sqrt{2}n^2d_0$ 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 $\Theta(\varepsilon d/n^4)$ 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 $O(\varepsilon)\mathrm{OPT}$ in the objective-function value: The additional cost of $O(\varepsilon d/n^3)$ per vertex can be charged to the objective as it is $\Omega(d/n^2)$ 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 $\{-d/2,-d/2+1,\dots,d/2-1,d/2\}$ 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 $\Theta(\varepsilon d/n^4)$ by the rounding step, the quadtree has depth $O(\log d)$.

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 $t^{(1)}, \dots, t^{(\kappa)}$ 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 computed in a fairly standard manner. A more detailed description can be found in the full version [28].

6 Hardness of approximation

This section is dedicated to the following theorem.

▶ **Theorem 14.** *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 [14] showed the following theorem.

▶ **Theorem 15** (Theorem 4.2 in [14])**.** *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 main idea of the proof of Theorem 14 is 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 $\gamma < 1 + \rho$. With the approximation hardness of ST(1,2), this contradicts the existence of a β-approximation algorithm Alg$'$ for the expanding search problem for any $\beta > 1$. The construction of the ESP instance in the reduction from ST(1,2) is similar to the one used for the travelling repairperson problem. Therein, we construct several copies of the ST(1,2) instance, which are then connected to a root vertex with an edge of high cost. However, a significant challenge is that we need to prove that the obtained expanding search sequence fulfills a property which we call *structured*. Intuitively, this property means that no copy of the original ST(1,2) instance is visited more than once and that all edges belonging to one copy are contiguous within the expanding search pattern. This property is trivial for the travelling repairperson problem since here using an expensive edge from the root to one of the copies more than once increases the total cost significantly. In the expanding search problem, however, these costs are not paid multiple times. The exact construction of the ESP instance and the proof of Theorem 14 can be found in the full version [28].

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.

2 Steve Alpern, Robbert Fokkink, Leszek Gasieniec, Roy Lindelauf, and V. S. Subrahmanian. *Search Theory*. Springer, New York, 2013.

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

4 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.

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

6 Spyros Angelopoulos, Christoph Dürr, and Thomas Lidbetter. The expanding search ratio of a graph. *Discrete Applied Mathematics*, 260:51–65, 2019.

7 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.

8 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.

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

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

11 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

12 Igor Averbakh. Emergency path restoration problems. *Discrete Optimization*, 9(1):58–64, 2012.

13 Igor Averbakh and Jordi Pereira. The flowtime network construction problem. *IIE Transactions*, 44(8):681–694, 2012.

14 Marshall Bern and Paul Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989.

15 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.

16 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.

17 Chandra Chekuri and Amit Kumar. Maximum coverage problem with group budget constraints and applications. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 72–83, 2004.

18 Thijs Dewilde, Dirk Cattrysse, Sofie Coene, Frits C. R. Spieksma, and Pieter Vansteenwegen. Heuristics for the traveling repairman problem with profits. *Computers & Operations Research*, 40:1700–1707, 2013.

19 Jittat Fakcharoenphol, Chris Harrelson, and Satish Rao. The k-traveling repairmen problem. *ACM Transactions on Algorithms*, 3(4):40:1–40:16, 2007.

20 Matteo Fischetti, Gilbert Laporte, and Silvano Martello. The delivery man problem and cumulative matroids. *Operations Research*, 41:1010–1176, 1993.

21 Zachary Friggstad, Mohammad R. Salavatipour, and Zoya Svitkina. Asymmetric traveling salesman path and directed latency problems. *SIAM Journal on Computing*, 42:1596–1619, 2013.

22 Shmuel Gal. Search games with mobile and immobile hider. *SIAM Journal on Control and Optimization*, 17:99–122, 1979.

23 Shmuel Gal. *Search Games.* Academic Press, New York, 1980.

24 Alfredo García, Pedro Jodrá, and Javier Tejel. A note on the travelling repairman problem. *Networks*, 40:27–31, 2002.

25 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.

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

27 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

28 Svenja M Griesbach, Felix Hommelsheim, Max Klimm, and Kevin Schewior. Improved approximation algorithms for the expanding search problem. *arXiv preprint arXiv:2301.03638*, 2023.

29 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.

30 Rufus Isaacs. *Differential Games.* John Wiley and Sons, New York, 1965.

31 David S. Johnson, Maria Minkoff, and Steven 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 David Kirkpatrick. Hyperbolic dovetailing. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, pages 516–527, 2009.

33 Elias Koutsoupias, Christos H. Papadimitriou, and Mihalis Yannakakis. Searching a fixed graph. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 280–289, 1996.

34 Sven O. Krumke, Willem E. de Paepe, Diana Poensgen, and Leen Stougie. News from the online traveling repairman. *Theoretical Computer Science*, 295:279–294, 2003.

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

36 Edward Minieka. The delivery man problem on a tree network. *Annals of Operations Research*, 18:261–266, 1989.

37 Viswanath 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.

38 Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

39 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.

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

41 Yushi Tan, Feng Qiu, Arindam K. Das, Daniel S. Kirschen, Payman Arabshahi, and Jianhui 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 John N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22:262–283, 1992.