



An Improved Approximation Algorithm for Dynamic Minimum Linear Arrangement

Marcin Bienkowski  

University of Wrocław, Poland

Guy Even  

Tel Aviv University, Israel

Abstract

The dynamic offline linear arrangement problem deals with reordering n elements subject to a sequence of edge requests. The input consists of a sequence of m edges (i.e., unordered pairs of elements). The output is a sequence of permutations (i.e., bijective mapping of the elements to n equidistant points). In step t , the order of the elements is changed to the t -th permutation, and then the t -th request is served. The cost of the output consists of two parts per step: request cost and rearrangement cost. The former is the current distance between the endpoints of the request, while the latter is proportional to the number of adjacent element swaps required to move from one permutation to the consecutive permutation. The goal is to find a minimum cost solution.

We present a deterministic $O(\log n \log \log n)$ -approximation algorithm for this problem, improving over a randomized $O(\log^2 n)$ -approximation by Olver et al. [22]. Our algorithm is based on first solving spreading-metric LP relaxation on a time-expanded graph, applying a tree decomposition on the basis of the LP solution, and finally converting the tree decomposition to a sequence of permutations. The techniques we employ are general and have the potential to be useful for other dynamic graph optimization problems.

2012 ACM Subject Classification Theory of computation \rightarrow Approximation algorithms analysis

Keywords and phrases Minimum Linear Arrangement, dynamic Variant, Optimization Problems, Graph Problems, approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2024.15

Funding Supported by Polish National Science Centre grant 2022/45/B/ST6/00559.

1 Introduction

This paper is motivated by a growing interest in optimization problems in a *dynamic* setting [22, 26]. By dynamic we mean that constraints or penalties associated with requests are ephemeral (i.e., disappear after the request is served). Such problems are well established in the online setting (e.g., metrical task systems [21] or server problems [17]), but are also of interest in the offline case.

A special case of dynamic optimization is the dynamic version of the classic Minimum Linear Arrangement problem (MLA). In the Minimum Linear Arrangement (MLA) problem, the input consists of a graph $G = (V, E)$. The output is an ordering of elements of V (i.e., a bijection π from V to $\{1, \dots, n\}$). The cost of the solution is the total stretch of the edges, i.e., $\sum_{(u,v) \in E} |\pi(u) - \pi(v)|$, and the goal is to find an ordering with minimum cost. The MLA problem is NP-hard [11]. A large variety of ideas and approximation techniques were developed for MLA [14, 7, 28] culminating in an $O(\sqrt{\log n} \cdot \log \log n)$ -approximation [2, 8].

Recently, the MLA problem has been studied in a dynamic setting, where the input consists of a sequence of m edges, and an algorithm has to output a sequence of permutations [22]. For a given edge (u, v) (a request) appearing in the input sequence, an algorithm may first change its current permutation π of elements paying γ for each swap of adjacent elements, and then



© Marcin Bienkowski and Guy Even;

licensed under Creative Commons License CC-BY 4.0

41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024).

Editors: Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshantov;

Article No. 15; pp. 15:1–15:19



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



it has to pay the usual price of $|\pi(u) - \pi(v)|$. The parameter $\gamma > 0$ is used to quantify the ratio between the cost of swapping an adjacent pair and serving a request between adjacent vertices. While the problem is appealing from the theoretical point of view, its solution can be also used for track management in domain wall memory [13, 22]. Olver et al. [22] proved that the dynamic variant of MLA admits a randomized $O(\log^2 n)$ -approximation for $\gamma = 1$.

As our problem definition does not specify the initial permutation¹, setting $\gamma > n \cdot m$ penalizes rearrangement to the extent that every solution with even one swap is suboptimal. Hence, in this setting, the DMLA problem reduces to the static MLA problem, thus proving that DMLA is NP-hard.

1.1 Our Result and Techniques

We present a deterministic approximation algorithm for the dynamic offline MLA problem, achieving an improved approximation ratio of $O(\log n \log \log n)$. We emphasize that the constants of the approximation ratio do not depend on the parameter γ nor the input length m . Similarly to Olver et al. [22], we work on a time-expanded graph that contains copies of the element set (one for each time step), and whose edges encode requests and rearrangements.

The $O(\log^2 n)$ -approximation algorithm of [22] is based on a non-trivial divide-and-conquer argument, where the time-expanded graph is recursively and randomly partitioned using a balanced cut routine. The translation of the recursive partitioning to a sequence of permutations requires a “delicate shuffling” argument. One of the challenges in their analysis is locally bounding the cost of rearrangements induced by the recursive decomposition. In contrast to their approach, we propose an algorithm that computes the sequence of permutations in a unified and straightforward manner directly from the decomposition of the time-expanded graph.

Our approach builds on the following ideas. First, apply a spreading LP relaxation to assign lengths to edges of the time-expanded graph (cf. Section 3). Next, apply a tree decomposition algorithm [7] to the time-expanded graph (cf. Section 4). This binary decomposition tree represents a laminar partitioning of the time-expanded graph. For every time step, the permutation is simply extracted by applying an in-order traversal of the decomposition tree intersected with the corresponding time-slice. (See Section 5 for a description of the algorithm.)

A key component of our analysis is a local charging scheme that bounds the solution cost by the cost of the decomposition tree (see Section 6). We rely on [7] to bound the latter by a function of the cost of the optimal solution to the LP relaxation.

1.2 Related Work on Dynamic Graph Problems

Dynamic MLA Problem. Our paper is among the few that study the *approximability* of graph problems, where the input is a sequence of requests in the form of edges and the output is a sequence of “configurations”. The cost of a solution consists of “serving” the request and the cost of “moving” from one configuration to the next configuration. These problems have been usually considered in the context of *online* algorithms, where the solution must be created without knowledge of future edges, and its time complexity is of secondary importance.

¹ We discuss the implication of this assumption in Section 7.

Indeed, the dynamic MLA problem has been studied also in online flavor. As every request incurs the cost of at least 1 and at most n , the competitive ratio of an online algorithm that does not change its permutation is trivially $O(n)$. Surprisingly, it is unknown whether an online algorithm that beats this ratio exists. On the negative side, Olver et al. showed that many natural online policies have competitive ratios of $\Omega(n/\log n)$; the best known lower bound is merely $\Omega(\log n)$ [22]. We believe that the techniques of this paper will help improve our understanding of the problem, and provide insights that could eventually be used to design online algorithms for dynamic MLA with non-trivial competitive ratios.

Dynamic Balanced Graph Partitioning. Another (seemingly similar) task is the *balanced graph partitioning* problem [1, 9, 19], where the goal is to split n graph vertices into ℓ clusters, each containing n/ℓ vertices, so that the number of crossing edges is minimized. For $\ell = 2$, the problem becomes the *bisection* problem [18, 25]. This problem has been recently extended to a dynamic *online* setting, where edges arrive one by one, and the algorithm may change the clustering on the fly [3, 5, 10, 15, 16, 23, 24, 27]. Large competitive ratios of online solutions motivated the study of the problem in the dynamic *offline* setting, and recently Räcke et al. [26] constructed an $O(\log n)$ -approximation for this variant.

On a high level, the framework of Räcke et al. [26] bears some similarities to ours: they also create a time-expanded graph, solve an LP-relaxation of the problem on this graph, use the edge lengths returned by the LP to partition this graph, and finally transform the partition into the final solution (sequence of graph clusterings). Similarities end here as the details of their approach are quite different. In particular, their LP relaxation is based on a distinct type of spreading metric (using knapsack-like constraints), they use Bartal's randomized decomposition [4] to partition the graph, and the final step of transforming the partition to a sequence of clusterings is completely different.

That said, we hope that their and our paper will inspire further work on dynamic graph optimization problems, and eventually a coherent set of tools will be developed to tackle such problems.

2 Preliminaries

For an integer ℓ , we use $[\ell] \triangleq \{1, \dots, \ell\}$. The union of mutually disjoint sets X and Y is denoted by $X \uplus Y$. Throughout this paper, we work with a set of elements Q and we denote its cardinality by n . A *permutation of Q* (or simply *a permutation*) is a bijection from Q to $[n]$. We say that two elements a and b are *adjacent* in permutation π if $|\pi(a) - \pi(b)| = 1$. To reduce ambiguity, we refer to members of Q as *elements*, to vertices of decomposition trees as *nodes*, and use the *vertices* only for vertices of the time-expanded graph defined in Section 3.

Permutation Distances. An unordered pair $\{x, y\}$ of distinct elements of Q is *discordant* with respect to permutations π and π' if $(\pi(x) - \pi(y)) \cdot (\pi'(x) - \pi'(y)) < 0$. We use two notions of distance between partitions π and π' :

- Kendall's tau-distance $\text{tdist}(\pi, \pi')$ equal to the minimum number of swaps of adjacent elements required to reach permutation π' from π . This distance is also equal to the number of discordant pairs between π and π' .
- Spearman's footrule distance defined as $\text{fdist}(\pi, \pi') \triangleq \sum_{v \in Q} |\pi'(v) - \pi(v)|$.

While $\text{tdist}(\pi, \pi')$ is convenient for establishing relations between decomposition trees and permutations, $\text{fdist}(\pi, \pi')$ is better suited for defining spreading metrics in LP relaxations. The Diaconis-Graham inequality [6, 20] states that these two distances can differ at most by a factor of 2, i.e.,

$$\text{tdist}(\pi, \pi') \leq \text{fdist}(\pi, \pi') \leq 2 \cdot \text{tdist}(\pi, \pi'). \quad (1)$$

Problem Definition. The Dynamic Minimum Linear Arrangement (DMLA) problem is specified by a tuple $\mathcal{I} = (n, m, \text{IN})$, where n is the number of elements (i.e., $|Q| = n$), m is the number of requests, and IN is the input sequence consisting of m requests, each being an unordered pair of distinct elements. That is, $\text{IN} \triangleq \{(a_t, b_t)\}_{t=1}^m$, where $(a_t, b_t) \in Q \times Q$ and $a_t \neq b_t$.

A feasible solution is a sequence of m permutations $\{\pi_t\}_{t=1}^m$, and its cost is

$$\text{COST}(\mathcal{I}, \{\pi_t\}_{t=1}^m) \triangleq \sum_{t=1}^m \{\gamma \cdot \text{fdist}(\pi_{t-1}, \pi_t) + |\pi_t(a_t) - \pi_t(b_t)|\}.$$

where we assume that $\pi_0 = \pi_1$, i.e., $\text{fdist}(\pi_{t-1}, \pi_t) = 0$. The goal is to find a feasible solution with minimum cost. We call $\text{fdist}(\pi_{t-1}, \pi_t)$ the *rearrangement cost* at time t and $|\pi_t(a_t) - \pi_t(b_t)|$ the *request cost* at time t . Finally, for an algorithm A , we denote its cost on \mathcal{I} by $A(\mathcal{I})$, and we use OPT to denote the optimal algorithm.

As the cost incurred in every time step is at least 1 and at most $n - 1$, we have the following trivial bounds on the value of OPT .

▷ **Claim 1.** $m \leq \text{OPT}(\mathcal{I}) \leq m \cdot (n - 1)$ for an instance $\mathcal{I} = (n, m, \text{IN})$.

Note on the Parameter γ . Consider the following greedy algorithm that at time t moves the element b_t so that it becomes adjacent to a_t . It pays at most $(n - 2) \cdot \gamma$ for rearrangement and then 1 for the request. For $\gamma < 1/n$, this amounts to at most 2. As OPT pays at least 1 for the request cost alone, the greedy algorithm is trivially an $O(1)$ -approximation for $\gamma < 1/n$. Hence, in the remaining part of the paper, we assume that $\gamma \geq 1/n$.

Consider an algorithm that does not employ rearranging. Namely, it finds an α -approximation for the (static) MLA instance that contains an edge for every request. This algorithm would also be an α -approximation for the DMLA instance if $\gamma > m \cdot (n - 1)$. Indeed, as $\text{OPT} \leq m \cdot (n - 1)$, employing even one swap would be suboptimal. Because the (static) MLA problem admits an $O(\log n \log \log n)$ -approximation, we may assume that $\gamma \leq m \cdot (n - 1)$.

Our algorithm crucially requires the assumption $\gamma \geq 1/n$ to guarantee the approximation ratio. The assumption $\gamma \leq m \cdot (n - 1)$ is used to ensure that the edge costs of our time-expanded graphs are polynomially bounded, and thus to ensure polynomial runtime.

3 Time-Expanded Graph and Linear Relaxation

In this section, we present a linear-programming relaxation for DMLA. The relaxation is defined over a time-expanded graph that represents the DMLA instance. We note that equivalent definitions of time-expanded graphs appeared in the papers of Olver et al. [22] and Räcke et al. [26].

Consider a DMLA instance $\mathcal{I} = (n, m, \text{IN})$. We represent \mathcal{I} by a weighted *time-expanded graph* $G = (V, E, c)$ as follows. Recall that Q is the set of elements. The set of vertices $V \triangleq Q \times \{0, \dots, m\}$ contains a copy of Q for every time $t \in \{0, \dots, m\}$. We refer to each such copy as a *time-slice*.

To simplify notation, we denote the vertex $(v, t) \in V$ by v^t , namely, v^t is the t -th copy of the *element* $v \in Q$. The t -th time-slice of graph G is $Q^t \triangleq Q \times \{t\}$. Furthermore, for a subset of elements $A \subseteq Q$, we use $A^t \triangleq \{v^t \mid v \in A\}$ to denote the corresponding set of vertices in the t -th time-slice.

There are two types of edges in E . First, we introduce a set of *request edges* E_{IN} containing an edge $\{a_t^t, b_t^t\}$ for every request (a_t, b_t) in the input IN . Second, we introduce $n \cdot m$ *migration edges* between copies of elements in consecutive time-slices. That is, the set of migration edges is $E_m \triangleq \{\{v^{t-1}, v^t\} \mid v \in Q, t \in [m]\}$. We identify each edge $e \in E$ with a set containing two of its endpoints.

Finally, for an edge $e \in E$, we define its *cost* by

$$c(e) \triangleq \begin{cases} 1 & \text{if } e \in E_{\text{IN}}, \\ \gamma & \text{if } e \in E_m. \end{cases}$$

We extend the function $c(e)$ to all subsets of edges $E' \subseteq E$, i.e., $c(E') = \sum_{e \in E'} c(e)$.

Naming Convention. Sometimes we want to refer to a vertex from V without specifying its time-slice. In such case, we use star instead of time superscript, i.e., we use names such as u^* or v^* , to emphasize that we refer to vertices (members of V) and not elements (members of Q).

Edge Lengths. A solution to the DMLA problem induces the assignment of *lengths* to edges of E in the following way:

- the length of a request edge $\{a_t^t, b_t^t\}$ is set to $|\pi_t(a_t) - \pi_t(b_t)|$;
- the length of a migration edge $\{v^{t-1}, v^t\}$ is set to $|\pi_{t-1}(v) - \pi_t(v)|$.

In particular, the total length of all (migration) edges between two consecutive time-slices Q^{t-1} and Q^t is equal to $\text{fdist}(\pi_{t-1}, \pi_t)$.

A valid solution to DMLA cannot induce an arbitrary assignment of edge lengths. As the permutation π_t assigns distinct positions to elements, the pairwise distances (shortest path distances induced by lengths) between their corresponding vertices can be lower-bounded appropriately. We make this observation more concrete when we create a linear relaxation of the problem. This hints at a possible way of tackling the problem: we first find an assignment of lengths to edges, and we use them to compute a sequence of permutations.

Spiders. To create a linear relaxation of the DMLA problem, we introduce a helper notion. For a vertex $v^* \in V$ and set $U \subseteq V \setminus \{v^*\}$, a multi-set of edges from E is called a (v^*, U) -*spider* if it is a union of $|U|$ paths from v^* to each vertex from U . (If an edge belongs to k such paths, the spider contains k copies of such edge.) We use $\mathcal{H}(v^*, U)$ to denote the set of all possible (v^*, U) -spiders.

LP Relaxation. The LP relaxation is obtained by introducing *spreading constraints* [7] to every time-slice. Consider a DMLA instance $\mathcal{I} = (n, m, \text{IN})$, and let $G = (V, E, c)$ denote the corresponding time-expanded graph. Let $S_j' \triangleq \sum_{i=1}^j i$ and $S_k \triangleq S_{\lfloor k/2 \rfloor}' + S_{\lceil k/2 \rceil}'$.

The LP relaxation introduces a variable z_e for every edge $e \in E$ and is formulated as follows.

$$\min \sum_{e \in E} c(e) \cdot z_e \tag{2a}$$

$$\text{s.t.} \quad \sum_{e \in H} z_e \geq S_{|A|} \quad \forall v \in Q, \forall A \subseteq Q \setminus v, \forall t \in \{0, \dots, m\}, \forall H \in \mathcal{H}(v^t, A^t), \tag{2b}$$

$$z_e \geq 0 \quad \forall e \in E. \tag{2c}$$

Before we argue that the formulation above is indeed a relaxation of the original DMLA problem, we first discuss the interpretation of spreading constraints (2b). For any two vertices $v^*, u^* \in V$, let $\text{dist}_z(v^*, u^*)$ denote their shortest-path distance in G induced by edge lengths $\{z_e\}_{e \in E}$. For a fixed time t , element $v \in Q$ and a set of elements $A \subseteq Q \setminus \{v\}$, constraints (2b) state that $\sum_{e \in H} z_e \geq S_{|A|}$ for every spider $H \in \mathcal{H}(v^t, A^t)$. That is, the total length of $|A|$ paths from v^t to all vertices of A^t has to be at least $S_{|A|}$. Constraints (2b) are thus equivalent to

$$\sum_{u \in A} \text{dist}_z(v^t, u^t) \geq S_{|A|}, \quad \forall v \in Q, \forall A \subseteq Q \setminus \{v\}, \forall t \in \{0, \dots, m\}. \quad (3)$$

► **Lemma 2.** *The minimization program above is a linear relaxation of the DMLA problem.*

Proof. Fix an instance \mathcal{I} of the DMLA problem specified by (n, m, IN) and the corresponding instance of the minimization LP above. Consider a solution to \mathcal{I} , that is, a sequence of permutations $\{\pi_t\}_{t=1}^m$. We have to show that there exists a solution to the LP whose cost is at most $\text{COST}(\mathcal{I}, \{\pi_t\}_{t=1}^m)$.

For the purpose of defining variables $\{z_e\}_{e \in E}$, we set $\pi_0 \triangleq \pi_1$. For an edge $e = \{u^{t_1}, v^{t_2}\} \in E$, we set the variable $z_e = |\pi_{t_1}(u) - \pi_{t_2}(v)|$. The cost of the LP solution is then

$$\begin{aligned} \sum_{e \in E} c(e) \cdot z_e &= \sum_{e \in E_m} \gamma \cdot z_e + \sum_{e \in E_{\text{IN}}} z_e \\ &= \sum_{t=1}^m \sum_{v \in Q} \gamma \cdot |\pi_t(v) - \pi_{t-1}(v)| + \sum_{t=1}^m |\pi_t(a_t) - \pi_t(b_t)| \\ &= \text{COST}(\mathcal{I}, \{\pi_t\}_{t=1}^m). \end{aligned}$$

It remains to show that the edge length variables satisfy the constraints; the only non-trivial one is (2b).

Fix such a constraint, given by element $v \in Q$, a subset $A \subseteq Q \setminus \{v\}$, time t , and a spider $H \in \mathcal{H}(v^t, A^t)$. We decompose H into $|A|$ paths: for an element $u \in A$, let H_u be the path from v^t to u^t in spider H . By a simple induction on path length, we have

$$\sum_{e \in H_u} z_e \geq |\pi_t(u) - \pi_t(v)| \quad (4)$$

for every element $u \in A$. By summing (4) over all elements from A , we obtain that

$$\sum_{e \in H} z_e = \sum_{u \in A} \sum_{e \in H_u} z_e \geq \sum_{u \in A} |\pi_t(u) - \pi_t(v)|.$$

We split the elements of A into two disjoint parts $A^- \triangleq \{u \in A \mid \pi_t(u) < \pi_t(v)\}$ and $A^+ \triangleq \{u \in A \mid \pi_t(u) > \pi_t(v)\}$. As π_t is bijective, $\sum_{u \in A^+} |\pi_t(u) - \pi_t(v)| \geq S'_{|A^+|}$ and $\sum_{u \in A^-} |\pi_t(u) - \pi_t(v)| \geq S'_{|A^-|} = S'_{|A| - |A^+|}$. Thus,

$$\sum_{e \in H} z_e \geq S'_{|A^+|} + S'_{|A| - |A^+|} \geq S'_{\lfloor |A|/2 \rfloor} + S'_{\lceil |A|/2 \rceil} = S_{|A|}.$$

The last inequality follows as the expression $S'_\ell + S'_{|A| - \ell}$ is minimized for $\ell = \lfloor |A|/2 \rfloor$. This shows that constraints (2b) hold and concludes the proof. ◀

4 Graph Decomposition

In this section, we present a poly-time deterministic graph decomposition procedure that is based on [7]. The input for the graph decomposition consists of an undirected graph with non-negative edge costs $G = (V, E, c)$ and non-negative edge lengths $\{z_e\}_{e \in E}$. A diameter parameter $d > 0$ specifies the “granularity” of the decomposition.

Distances and Diameters. For any two vertices $u, v \in V$, let $\text{dist}_z(u, v)$ denote the shortest-path distance in G induced by the edge lengths z . Furthermore, for a non-empty set $U \subseteq V$, let $\text{diam}_z(U) = \max_{u, v \in U} \text{dist}_z(u, v)$ be the *diameter* of U . If U contains two vertices that are not connected in G , then its diameter is infinite.

Decomposition Tree. Fix a real number $d > 0$. A d -decomposition tree of the graph $G = (V, E, c)$ is a triple $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}}, \alpha)$ where $(V_{\mathcal{T}}, E_{\mathcal{T}})$ is a rooted binary tree where each internal node has two children, one marked *left*, and the other one *right*. The function $\alpha : V_{\mathcal{T}} \rightarrow 2^V \setminus \{\emptyset\}$ maps tree nodes to non-empty subsets of graph vertices V and satisfies the following conditions:

- $\alpha(w_r) = V$, where w_r is the root of \mathcal{T} ;
- $\alpha(w) = \alpha(w_L) \uplus \alpha(w_R)$ for an internal node $w \in V_{\mathcal{T}}$ and its two children w_L and w_R ;
- $\text{diam}_z(\alpha(w)) \geq d$ for every internal node $w \in V_{\mathcal{T}}$ and $\text{diam}_z(\alpha(w)) < d$ for every leaf $w \in V_{\mathcal{T}}$.

Note that the decomposition tree \mathcal{T} represents a laminar decomposition of the graph vertices V .

Cuts. For any subsets of graph nodes $U, U' \subseteq V$, let $E[U] \subseteq E$ be the subset of edges with both endpoints in U , and let $E[U, U'] \subseteq E$ be the subset of edges with one endpoint in U and the other one in U' . For an internal tree node $w \in V_{\mathcal{T}}$ with children w_L and w_R , we define $\text{cut}(w) \subseteq E$ as the set of edges between $\alpha(w_L)$ and $\alpha(w_R)$, i.e.,

$$\begin{aligned} \text{cut}(w) &\triangleq E[\alpha(w_L), \alpha(w_R)] \\ &= E[\alpha(w)] \setminus (E[\alpha(w_L)] \uplus E[\alpha(w_R)]). \end{aligned}$$

For a leaf $w \in V_{\mathcal{T}}$, we define $\text{cut}(w)$ to be the empty set.

► **Definition 3.** The cost of a decomposition tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}}, \alpha)$ of a graph $G = (V, E, c)$ with non-negative edge lengths $\{z_e\}_{e \in E}$ is defined as

$$\text{COST}_{G,z}(\mathcal{T}) \triangleq \sum_{w \in V_{\mathcal{T}}} c(\text{cut}(w)) \cdot \text{diam}_z(\alpha(w)).$$

In the definition above, we assume that $0 \cdot \infty = 0$.

Cheap Decomposition Tree. The following theorem is implicitly proven (in a slightly different form) in [7]. For completeness, we present its proof in the appendix.

► **Theorem 4.** Fix a real $d > 0$ and a graph $G = (V, E, c)$ with non-negative edge lengths $\{z_e\}_{e \in E}$. It is possible to construct, in polynomial time, a d -decomposition tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}}, \alpha)$ of G , such that

$$\text{COST}_{G,z}(\mathcal{T}) \leq \xi \cdot O(\log(f_{c,d} \cdot \xi) \cdot \log \log(f_{c,d} \cdot \xi)),$$

where $\xi = \sum_{e \in E} c(e) \cdot z_e$, $f_{c,d} = \max\{1, 1/(d \cdot c_{\min})\}$, and $c_{\min} = \min_{e \in E} c(e)$.

We emphasize that Theorem 4 does not depend on particular properties of graph G or the edge lengths $\{z_e\}_{e \in E}$. In particular, G does not have to be a time-expanded graph of a DMLA instance, and the theorem does not assume that lengths $\{z_e\}_{e \in E}$ satisfy the spreading constraints (2b). In [7], the stopping condition for the decomposition is when one reaches an independent set with respect to an auxiliary graph defined over the same vertex set. Conceptually, our stopping condition can be viewed as reaching an independent set with respect to an auxiliary *hypergraph* defined over the same vertex set. Indeed, the auxiliary hypergraph in a d -decomposition contains a hyperedge for every subset of vertices whose diameter is at least d .

5 Approximation Algorithm for DMLA

Algorithm definition. Consider an instance $\mathcal{I} = (n, m, \text{IN})$ of the DMLA problem. Our algorithm ALG first constructs a time-expanded graph $G = (V, E, c)$ for \mathcal{I} . Next, ALG solves the LP relaxation defined by (2a)–(2c), obtaining an optimal solution $\{z_e\}_{e \in E}$ to this LP. Then, it computes a $(1/4)$ -decomposition tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}}, \alpha)$ of G using the routine guaranteed by Theorem 4.

Finally, ALG decodes \mathcal{T} into a sequence of permutations $\{\pi_t\}_{t=1}^m$. To this end, let w_1, w_2, \dots, w_ℓ denote the sequence of leaves of \mathcal{T} ordered by an in-order traversal of \mathcal{T} (that scans the left subtree before the right subtree). Note that the corresponding sequence of sets $\{\alpha(w_i)\}_{i=1}^\ell$ is a partition of V . For a fixed time t , we define a sequence of sets $\{B_i^t\}_{i=1}^\ell$, where $B_i^t \triangleq \alpha(w_i) \cap Q^t$. By Lemma 9 (cf. Section 6), every set B_i^t contains either one element of Q^t or is empty. Thus, the sequence $\{B_i^t\}_{i=1}^\ell$ induces a permutation of Q^t (and hence of Q). Let π_t denote this permutation.

Handling Arbitrary Input Lengths. In the next section, we prove the following bound.

► **Theorem 5.** *On an instance \mathcal{I} , ALG returns a feasible solution of cost $O(\log(n \cdot \text{OPT}(\mathcal{I})) \cdot \log \log(n \cdot \text{OPT}(\mathcal{I}))) \cdot \text{OPT}(\mathcal{I})$, where $\text{OPT}(\mathcal{I})$ denotes the cost of the optimal solution on \mathcal{I} .*

We are interested in obtaining a smaller asymptotic approximation ratio even if $\text{OPT}(\mathcal{I})$ is super-polynomial in n .

► **Theorem 6.** *There is a poly-time deterministic approximation algorithm for DMLA that achieves an approximation ratio of $O(\log n \cdot \log \log n)$.*

Proof. By Claim 1, $\text{OPT}(\mathcal{I})$ is super-polynomial in n only if m is. In this case, we modify ALG as follows. Split the input sequence into L phases $\mathcal{I}_1, \dots, \mathcal{I}_L$, each consisting of $m' = n^2$ requests, with the last phase possibly being shorter. Apply ALG separately to each phase, and return the concatenation of the permutation sequences output by ALG for each phase.

As OPT needs to pay 1 for each request, $\text{OPT}(\mathcal{I}) \geq (L - 1) \cdot m'$. On the other hand, by Claim 1, $\text{OPT}(\mathcal{I}_\ell) \leq n \cdot m' = n^3$ for every phase \mathcal{I}_ℓ . The rearrangement cost incurred by the transition from a permutation ending a phase to the permutation beginning the next phase at most n^2 . The cost of the whole solution is then

$$\begin{aligned} \text{ALG}(\mathcal{I}) &= (L - 1) \cdot n^2 + \sum_{\ell=1}^L \text{ALG}(\mathcal{I}_\ell) \\ &\leq \text{OPT}(\mathcal{I}) + \sum_{\ell=1}^L O(\log(n \cdot \text{OPT}(\mathcal{I}_\ell)) \cdot \log \log(n \cdot \text{OPT}(\mathcal{I}_\ell))) \cdot \text{OPT}(\mathcal{I}_\ell) \\ &= \text{OPT}(\mathcal{I}) + O(\log n \cdot \log \log n) \cdot \sum_{\ell=1}^L \text{OPT}(\mathcal{I}_\ell) \\ &= O(\log n \cdot \log \log n) \cdot \text{OPT}(\mathcal{I}), \end{aligned}$$

where in the inequality above we used Theorem 5 for upper-bounding $\text{ALG}(\mathcal{I}_\ell)$ for each ℓ . ◀

Polynomial Runtime. ALG can be implemented in time polynomial in n and m . This follows trivially except for its two building blocks: solving the LP and computing the decomposition tree. The latter runs in polynomial time by Theorem 4.

For the former (solving the LP), we note that although the linear programming formulation contains exponentially many constraints, it can be solved in polynomial time by the Ellipsoid method [12] with a separation oracle that returns a violating constraint if one exists. To this end, following [7], given edge lengths $\{z_e\}_{e \in E}$, the oracle first computes all-pairs shortest paths, i.e., the value of function dist_z . To find a violated constraint among constraints (3), it suffices to check, for every time t and every element $v \in Q$, whether there exists a set $A \subseteq Q \setminus \{v\}$, such that $\sum_{u \in A} \text{dist}_z(v^t, u^t) \geq S_{|A|}$. This becomes simple once we fix the cardinality k of A : if inequality is violated, it is violated by the set A , such that A^k contains k elements of $Q^t \setminus \{v^t\}$ that are closest to v^t (with respect to dist_z).

6 Approximation Ratio

In this section, we prove the approximation ratio stated in Theorem 5. Fix a DMLA instance $\mathcal{I} = (n, m, \text{IN})$. Let $G = (V, E, c)$ be the time-expanded graph, $\{z_e\}_{e \in E}$ be the edge lengths returned by the LP relaxation, and $\mathcal{T} = (V_T, E_T, \alpha)$ be the $(1/4)$ -decomposition tree of G computed according to Theorem 4.

6.1 Relating OPT to the Decomposition Tree

► **Lemma 7.** *It holds that $\text{COST}_{G,z}(\mathcal{T}) \leq \text{OPT}(\mathcal{I}) \cdot O(\log(n \cdot \text{OPT}(\mathcal{I})) \cdot \log \log(n \cdot \text{OPT}(\mathcal{I})))$.*

Proof. Let $c_{\min} = \min_{e \in E} \{c(e)\}$. Note that $c_{\min} = \min\{1, \gamma\}$ by the definition of graph G . As we assumed (cf. Section 2) that $\gamma \geq 1/n$, it holds that $c_{\min} \geq 1/n$. Let $\xi = \sum_{e \in E} c(e) \cdot z_e$. By Theorem 4, the computed $(1/4)$ -decomposition tree satisfies

$$\text{COST}_{G,z}(\mathcal{T}) \leq \xi \cdot O(\log(4n \cdot \xi) \cdot \log \log(4n \cdot \xi)).$$

As ξ is the optimal solution to the fractional relaxation of the problem (cf. Lemma 2), $\xi \leq \text{OPT}(\mathcal{I})$, which concludes the proof. ◀

6.2 Graph Cost and its Relation to the Decomposition Tree

In the previous section, we related $\text{COST}_{G,z}(\mathcal{T})$ to $\text{OPT}(\mathcal{I})$, and thus it remains to relate $\text{ALG}(\mathcal{I})$ to $\text{COST}_{G,z}(\mathcal{T})$. As we show later, $\text{ALG}(\mathcal{I})$ can be naturally expressed as a sum of costs of particular edges in G taken with some multiplicity. On the other hand, $\text{COST}_{G,z}(\mathcal{T})$ is defined as $\sum_{w \in V_T} c(\text{cut}(w)) \cdot \text{diam}_z(\alpha(w))$, i.e., in terms of edge lengths. To facilitate a combinatorial comparison between $\text{ALG}(\mathcal{I})$ and $\text{COST}_{G,z}(\mathcal{T})$, we first provide an alternative lower bound on $\text{COST}_{G,z}(\mathcal{T})$, called *graph cost*, which is easier to relate to $\text{ALG}(\mathcal{I})$. To this end, we start with a few helper notions.

Least Common Ancestor. We define the function $\text{lca} : 2^V \setminus \{\emptyset\} \rightarrow V_T$ as follows. Fix a non-empty set of vertices $U \subseteq V$. The set of tree nodes w such that $\alpha(w) \supseteq U$ forms a path in \mathcal{T} starting at the root. Let $\text{lca}(U)$ be the last node on this path (furthest from the root).

We drop the set notation and use $\text{lca}(u_1, u_2, \dots, u_\ell)$ instead of $\text{lca}(\{u_1, u_2, \dots, u_\ell\})$. Note that for every internal tree node w and every edge $\{u, v\} \in \text{cut}(w)$, it holds that $\text{lca}(u, v) = w$.

15:10 An Improved Approximation Algorithm for Dynamic Minimum Linear Arrangement

Width. For a non-empty subset of vertices $U \subseteq V$, we define

$$\text{width}(U) \triangleq \max_{0 \leq t \leq m} |U \cap Q^t| - 1.$$

Graph Cost. The *cost* of a graph $G = (V, E, c)$ with respect to its decomposition tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}}, \alpha)$ is defined as

$$\text{COST}_{\mathcal{T}}^*(G) \triangleq \sum_{e \in E} c(e) \cdot \text{width}(\text{lca}(e)).$$

Relating Graph Cost to Decomposition Tree Cost. We show that $\text{COST}_{\mathcal{T}}^*(G) \leq 4 \cdot \text{COST}_{G,z}(\mathcal{T})$. To this end, we use spreading properties of $\{z_e\}_{e \in E}$ to relate the width of a set to its diameter.

► **Lemma 8.** *For a non-empty set $U \subseteq V$, it holds that $\text{width}(U) \leq 4 \cdot \text{diam}_z(U)$.*

Proof. Let $t = \arg \max_{s \in \{0, \dots, m\}} |U \cap Q^s|$, i.e., $\text{width}(U) = |U \cap Q^t| - 1$. For succinctness, let $Y = U \cap Q^t$. Below, we prove that

$$\text{diam}_z(Y) \geq (|Y| - 1)/4. \quad (5)$$

This will imply the lemma as $\text{width}(U) = |Y| - 1 \leq 4 \cdot \text{diam}_z(Y) \leq 4 \cdot \text{diam}_z(U)$.

If $|Y| = 1$, then $\text{diam}_z(Y) = 0$, and (5) follows trivially. Thus, in the following, we assume that $|Y| > 1$. Let v^t be an arbitrary vertex from Y . As $\{z_e\}_{e \in E}$ satisfies LP constraints and thus also (3), we obtain

$$\sum_{u^t \in Y \setminus \{v^t\}} \text{dist}_z(v^t, u^t) \geq S_{|Y|-1}. \quad (6)$$

A simple calculation shows that $S_k \geq k^2/4 + k/2$ for every $k \geq 0$ (the relation is tight for even k). Thus, $S_k/k = k/4 + 1/2 > k/4$ for every $k \geq 1$. Applying the averaging argument to (6), we obtain that there exists $u^t \in Y \setminus \{v^t\}$, such that

$$\text{dist}_z(v^t, u^t) \geq \frac{S_{|Y|-1}}{|Y|-1} > \frac{|Y|-1}{4}.$$

As $\text{diam}_z(Y) \geq \text{dist}_z(v^t, u^t)$, (5) follows. ◀

► **Lemma 9.** *Fix a leaf $w \in \mathcal{T}$. Then, $\text{width}(\alpha(w)) = 0$, and consequently, $|\alpha(w) \cap Q^t| \leq 1$ for every time t .*

Proof. As \mathcal{T} is a $(1/4)$ -decomposition tree, $\text{diam}_z(\alpha(w)) < 1/4$. Thus, $\text{width}(\alpha(w)) < 1$ by Lemma 8. The first part of the lemma follows as $\text{width}(\alpha(w))$ is an integer. The second part follows as $|\alpha(w) \cap Q^t| \leq \text{width}(\alpha(w)) + 1$ by the definition of *width*. ◀

► **Lemma 10.** *It holds that $\text{COST}_{\mathcal{T}}^*(G) \leq 4 \cdot \text{COST}_{G,z}(\mathcal{T})$.*

Proof. Fix an edge $e \in E$. We first claim that

$$\text{width}(\alpha(\text{lca}(e))) = \sum_{w \in V_{\mathcal{T}} : e \in \text{cut}(w)} \text{width}(\alpha(w)). \quad (7)$$

Indeed, if $\text{lca}(e)$ is a leaf of \mathcal{T} , then by Lemma 9, the left-hand side is 0 and the sum on the right-hand side is empty. If, however, $\text{lca}(e)$ is an internal node of \mathcal{T} , then $e \in \text{cut}(\text{lca}(e))$, and thus the sum on the right-hand side contains only one element $w = \text{lca}(e)$, and the claim follows.

Now, by the definition of $\text{COST}_{\mathcal{T}}^*(G)$,

$$\begin{aligned}
\text{COST}_{\mathcal{T}}^*(G) &= \sum_{e \in E} c(e) \cdot \sum_{w \in V_{\mathcal{T}} : e \in \text{cut}(w)} \text{width}(\alpha(w)) && \text{(by (7))} \\
&= \sum_{w \in V_{\mathcal{T}}} \sum_{e \in \text{cut}(w)} c(e) \cdot \text{width}(\alpha(w)) \\
&= \sum_{w \in V_{\mathcal{T}}} c(\text{cut}(w)) \cdot \text{width}(\alpha(w)) \\
&\leq 4 \cdot \sum_{w \in V_{\mathcal{T}}} c(\text{cut}(w)) \cdot \text{diam}_z(\alpha(w)) && \text{(by Lemma 8)} \\
&= 4 \cdot \text{COST}_{G,z}(\mathcal{T}). && \blacktriangleleft
\end{aligned}$$

6.3 Relating ALG to the Graph Cost

Now we may relate $\text{ALG}(\mathcal{I})$ to the graph cost $\text{COST}_{\mathcal{T}}^*(G) = \sum_{e \in E} c(e) \cdot \text{width}(\text{lca}(e))$. Our charging scheme preserves time locality: we relate the request and the rearrangement costs in step t to the graph cost pertaining to edges of G corresponding to time t .

► **Lemma 11.** *For every time t , it holds that $|\pi_t(a_t) - \pi_t(b_t)| \leq \text{width}(\alpha(\text{lca}(a_t^t, b_t^t)))$.*

Proof. It is convenient to look at the permutation π_t output by ALG as an ordered sequence of vertices from Q^t (cf. Section 5). Recall that this sequence is obtained by an in-order traversal of \mathcal{T} restricted to vertices from Q^t . In particular, all vertices from $\alpha(\text{lca}(a_t^t, b_t^t)) \cap Q^t$ form a contiguous part of permutation π_t and this part contains both a_t^t and b_t^t . Therefore, $|\pi_t(a_t) - \pi_t(b_t)| \leq |\alpha(\text{lca}(a_t^t, b_t^t)) \cap Q^t| - 1 \leq \text{width}(\alpha(\text{lca}(a_t^t, b_t^t)))$. ◀

► **Lemma 12.** *Fix a time t and let $\{u, v\}$ be a discordant pair of elements from Q with respect to permutations π_{t-1} and π_t . Then, either $v^t \in \alpha(\text{lca}(u^{t-1}, u^t))$ or $u^t \in \alpha(\text{lca}(v^{t-1}, v^t))$ (or both).*

Proof. Let $U = \{u^{t-1}, u^t, v^{t-1}, v^t\}$ and let $w = \text{lca}(U)$. As $\alpha(w)$ contains both u^t and v^t , Lemma 9 implies that w is an internal node of \mathcal{T} . Let w_L and w_R be the children of w in \mathcal{T} . By the definition of lca , $\alpha(w_L) \cap U \neq \emptyset$ and $\alpha(w_R) \cap U \neq \emptyset$.

It is not possible that $\{u^{t-1}, u^t\} \subseteq \alpha(w_L)$ and $\{v^{t-1}, v^t\} \subseteq \alpha(w_R)$: in such case, $\{u, v\}$ would not be a discordant pair, i.e., u would be before v in both permutations π_{t-1} and π_t . For the same reason, it is not possible that $\{u^{t-1}, u^t\} \subseteq \alpha(w_R)$ and $\{v^{t-1}, v^t\} \subseteq \alpha(w_L)$.

This means that either $\{u^{t-1}, u^t\} \in \text{cut}(w)$ or $\{v^{t-1}, v^t\} \in \text{cut}(w)$ (or both). In the former case $\text{lca}(u^{t-1}, u^t) = w$ while in the latter $\text{lca}(v^{t-1}, v^t) = w$, which concludes the proof. ◀

► **Lemma 13.** *For every time t , it holds that $\text{tdist}(\pi_{t-1}, \pi_t) \leq \sum_{v \in Q} \text{width}(\alpha(\text{lca}(v^{t-1}, v^t)))$.*

Proof. We create an injective mapping M from all $\text{tdist}(\pi_{t-1}, \pi_t)$ discordant pairs (with respect to permutations π_{t-1} and π_t) to pairs in $V_{\mathcal{T}} \times V$. A discordant pair $\{u, v\}$ is mapped:

- to the pair $(\text{lca}(u^{t-1}, u^t), v^t)$ if $v^t \in \alpha(\text{lca}(u^{t-1}, u^t))$, or
- to the pair $(\text{lca}(v^{t-1}, v^t), u^t)$ if $u^t \in \alpha(\text{lca}(v^{t-1}, v^t))$.

By Lemma 12, at least one of the conditions above must hold. If both hold, then we map the discordant pair arbitrarily to one of the pairs above.

Because the domain of M contains all discordant pairs, its cardinality equals $\text{tdist}(\pi_{t-1}, \pi_t)$. Moreover, the range of M is contained in the set of the following pairs

$$\bigcup_{v \in Q} \{(\text{lca}(v^{t-1}, v^t), y^t) \mid y^t \in \alpha(\text{lca}(v^{t-1}, v^t)) \cap (Q^t \setminus \{v^t\})\}.$$

15:12 An Improved Approximation Algorithm for Dynamic Minimum Linear Arrangement

Thus, the cardinality of the range of M is at most $\sum_{v \in Q} (|\alpha(\text{lca}(v^{t-1}, v^t)) \cap Q^t| - 1)$. As M is injective, its range is not smaller than its domain, i.e.,

$$\begin{aligned} \text{tdist}(\pi_{t-1}, \pi_t) &\leq \sum_{v \in Q} |\alpha(\text{lca}(v^{t-1}, v^t)) \cap Q^t| - 1 \\ &\leq \sum_{v \in Q} \text{width}(\alpha(\text{lca}(v^{t-1}, v^t))). \end{aligned} \quad \blacktriangleleft$$

► **Lemma 14.** *It holds that $\text{ALG}(\mathcal{I}) \leq 2 \cdot \text{COST}_{\mathcal{T}}^*(G)$.*

Proof. Let $\{\pi_t\}_{t=1}^m$ be the output of ALG on instance \mathcal{I} . The total rearrangement cost of ALG is

$$\begin{aligned} \sum_{t=1}^m \gamma \cdot \text{fdist}(\pi_{t-1}, \pi_t) &= 2 \cdot \sum_{t=1}^m \gamma \cdot \text{tdist}(\pi_{t-1}, \pi_t) && \text{(by (1))} \\ &\leq 2 \cdot \sum_{t=1}^m \sum_{v \in Q} \gamma \cdot \text{width}(\alpha(\text{lca}(v^{t-1}, v^t))) && \text{(by Lemma 13)} \\ &= 2 \cdot \sum_{e \in E_m} c(e) \cdot \text{width}(\alpha(\text{lca}(e))). && \text{(by the definition of } E_m) \end{aligned}$$

Moreover, its total request cost is

$$\begin{aligned} \sum_{t=1}^m |\pi_t(a_t) - \pi_t(b_t)| &\leq \sum_{t=1}^m 1 \cdot \text{width}(\alpha(\text{lca}(a_t^t, b_t^t))) && \text{(by Lemma 11)} \\ &= \sum_{e \in E_{\text{IN}}} c(e) \cdot \text{width}(\alpha(\text{lca}(e))). && \text{(by the definition of } E_{\text{IN}}) \end{aligned}$$

Summing up, we obtain that $\text{ALG}(\mathcal{I}) \leq 2 \cdot \sum_{e \in E} c(e) \cdot \text{width}(\alpha(\text{lca}(e))) = 2 \cdot \text{COST}_{\mathcal{T}}^*(G)$. ◀

6.4 Calculating Approximation Ratio

We are now ready to prove our main result, restated for convenience below.

► **Theorem 5.** *On an instance \mathcal{I} , ALG returns a feasible solution of cost $O(\log(n \cdot \text{OPT}(\mathcal{I})) \cdot \log \log(n \cdot \text{OPT}(\mathcal{I}))) \cdot \text{OPT}(\mathcal{I})$, where $\text{OPT}(\mathcal{I})$ denotes the cost of the optimal solution on \mathcal{I} .*

Proof. Fix an instance \mathcal{I} , corresponding graph G , output $\{z_e\}_{e \in E}$ of the LP, and the decomposition tree \mathcal{T} of G . Then,

$$\begin{aligned} \text{ALG}(\mathcal{I}) &\leq 4 \cdot \text{COST}_{\mathcal{T}}^*(G) && \text{(by Lemma 14)} \\ &\leq 8 \cdot \text{COST}_{G, z}^*(G) && \text{(by Lemma 10)} \\ &\leq 8 \cdot \text{OPT}(\mathcal{I}) \cdot O(\log(n \cdot \text{OPT}(\mathcal{I})) \cdot \log \log(n \cdot \text{OPT}(\mathcal{I}))). && \text{(by Lemma 7)} \quad \blacktriangleleft \end{aligned}$$

As stated in Theorem 6, the algorithm can be easily transformed into an $O(\log n \cdot \log \log n)$ -approximation. We also note that the constants hidden in O -notation do not depend on γ .

7 Final Remarks

In this paper, we present an $O(\log n \cdot \log \log n)$ -approximation algorithm for the dynamic minimum linear arrangement (DMLA) problem, improving over $O(\log^2 n)$ bound by Olver et al. [22].

We note that in the variant considered by Olver et al. [22], the initial permutation π_0 is fixed and is the part of the input instance. A small modification of their approach yields the same approximation ratio for the case where the input does not specify the initial permutation (as in our solution).

Conversion in the other direction is trivial, albeit it comes at a certain cost. Let π_1, \dots, π_m denote the output of our algorithm without an initial permutation. Simply prepend the given initial permutation π_0 as the initial one. The extra cost of the rearrangement from permutation π_0 to π_1 is at most $O(n^2)$. This additive cost does not influence the asymptotic approximation ratio if $m = \Omega(n^2)$. We leave handling shorter instances more efficiently as further work.

References

- 1 Konstantin Andreev and Harald Räcke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006. doi:10.1007/s00224-006-1350-7.
- 2 Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM*, 56(2):5, 2009. doi:10.1145/1502793.1502794.
- 3 Chen Avin, Marcin Bienkowski, Andreas Loukas, Maciej Pacut, and Stefan Schmid. Dynamic balanced graph partitioning. *SIAM Journal on Discrete Mathematics*, 34(3):1791–1812, 2020. doi:10.1137/17M1158513.
- 4 Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proc. 37th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 184–193, 1996. doi:10.1109/SFCS.1996.548477.
- 5 Marcin Bienkowski, Martin Böhm, Martin Koutecký, Thomas Rothvoß, Jirí Sgall, and Pavel Veselý. Improved analysis of online balanced clustering. In *Proc. 19th Workshop on Approximation and Online Algorithms (WAOA)*, pages 224–233, 2021. doi:10.1007/978-3-030-92702-8_14.
- 6 Persi Diaconis and R. L. Graham. Spearman’s footrule as a measure of disarray. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(2):262–268, 1977. doi:10.1111/j.2517-6161.1977.tb01624.x.
- 7 Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *Journal of the ACM*, 47(4):585–616, 2000. doi:10.1145/347476.347478.
- 8 Uriel Feige and James R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Information Processing Letters*, 101(1):26–29, 2007. doi:10.1016/j.ipl.2006.07.009.
- 9 Andreas Emil Feldmann and Luca Foschini. Balanced partitions of trees and applications. *Algorithmica*, 71(2):354–376, 2015. doi:10.1007/s00453-013-9802-3.
- 10 Tobias Forner, Harald Räcke, and Stefan Schmid. Online balanced repartitioning of dynamic communication patterns in polynomial time. In *2nd Symposium on Algorithmic Principles of Computer Systems (APOCS)*, pages 40–54, 2021. doi:10.1137/1.9781611976489.4.
- 11 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 12 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.

- 13 Shouzhen Gu, Edwin Hsing-Mean Sha, Qingfeng Zhuge, Yiran Chen, and Jingtong Hu. Area and performance co-optimization for domain wall memory in application-specific embedded systems. In *Proc. 52nd Annual Design Automation Conference (DAC)*, pages 20:1–20:6, 2015. doi:10.1145/2744769.2744800.
- 14 Mark D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems (extended abstract). In *Proc. 30th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 604–609, 1989. doi:10.1109/SFCS.1989.63542.
- 15 Monika Henzinger, Stefan Neumann, Harald Räcke, and Stefan Schmid. Tight bounds for online graph partitioning. In *Proc. 32nd ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 2799–2818, 2021. doi:10.1137/1.9781611976465.166.
- 16 Monika Henzinger, Stefan Neumann, and Stefan Schmid. Efficient distributed workload (re-)embedding. In *Proc. SIGMETRICS/Performance Joint Int. Conf. on Measurement and Modeling of Computer Systems*, pages 43–44, 2019. doi:10.1145/3309697.3331503.
- 17 Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009. doi:10.1016/j.cosrev.2009.04.002.
- 18 Robert Krauthgamer. Minimum bisection. In *Encyclopedia of Algorithms*, pages 1294–1297. Springer, 2016. doi:10.1007/978-1-4939-2864-4_231.
- 19 Robert Krauthgamer, Joseph Naor, and Roy Schwartz. Partitioning graphs into balanced components. In *Proc. 20th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 942–949, 2009. doi:10.1137/1.9781611973068.102.
- 20 Ravi Kumar and Sergei Vassilvitskii. Generalized distances between rankings. In *Proc. 19th Int. World Wide Web Conference (WWW)*, pages 571–580, 2010. doi:10.1145/1772690.1772749.
- 21 Manor Mendel. Metrical task systems. In *Encyclopedia of Algorithms*, pages 1279–1282. Springer, 2016. doi:10.1007/978-1-4939-2864-4_229.
- 22 Neil Olver, Kirk Pruhs, Kevin Schewior, René Sitters, and Leen Stougie. The itinerant list update problem. In *Proc. 16th Workshop on Approximation and Online Algorithms (WAOA)*, pages 310–326, 2018. doi:10.1007/978-3-030-04693-4_19.
- 23 Maciej Pacut, Mahmoud Parham, and Stefan Schmid. Brief announcement: Deterministic lower bound for dynamic balanced graph partitioning. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 461–463, 2020. doi:10.1145/3382734.3405696.
- 24 Maciej Pacut, Mahmoud Parham, and Stefan Schmid. Optimal online balanced graph partitioning. In *Proc. 40th IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1–9, 2021. doi:10.1109/INFOCOM42981.2021.9488824.
- 25 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, pages 255–264, 2008. doi:10.1145/1374376.1374415.
- 26 Harald Räcke, Stefan Schmid, and Ruslan Zabrodin. Approximate dynamic balanced graph partitioning. In *Proc. 34th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 401–409. ACM, 2022. doi:10.1145/3490148.3538563.
- 27 Rajmohan Rajaraman and Omer Wasim. Improved bounds for online balanced graph repartitioning. In *Proc. 30th European Symp. on Algorithms (ESA)*, pages 83:1–83:15, 2022. doi:10.4230/LIPIcs.ESA.2022.83.
- 28 Satish Rao and Andréa W. Richa. New approximation techniques for some linear ordering problems. *SIAM Journal on Computing*, 34(2):388–404, 2004. doi:10.1137/S0097539702413197.

A Constructing Decomposition Tree

In this section, we show how to construct a d -decomposition tree satisfying the guarantees of Theorem 4. In the description below, we fix a real number $d > 0$, a graph $G = (V, E, c)$ with non-negative edge costs c , and a set of edge lengths $\{z_e\}_{e \in E}$. Recall that for all subsets of vertices $U, U' \subseteq V$, we defined $E[U] \triangleq E \cap (U \times U)$ and $E[U, U'] \triangleq E \cap (U \times U' \cup U' \times U)$. Furthermore, let $G[U]$ be the graph G restricted to set U . Note that $G[U]$ might be disconnected even if G is connected. Let

$$\text{vol } U \triangleq \sum_{e \in E[U]} c(e) \cdot z_e$$

be the *volume* of U .

In Appendix A.1, we argue that for an arbitrary subset $U \subseteq V$ satisfying $\text{diam}_z(U) \geq d$, we may efficiently partition U into two parts, and relate the cost of the cut between these two parts, the diameter of U , and the volume of U (cf. Lemma 17).

The d -decomposition tree \mathcal{T} of G is simply a binary tree, constructed by the iterative application of set partitioning procedure: we start with the whole vertex set V and terminate when the diameter of the considered set of vertices is less than d . In Appendix A.2 and Appendix A.3, we show that the resulting tree \mathcal{T} satisfies the properties of Theorem 4.

We present our construction and cost bound assuming that $\min_{e \in E} c(e) \geq 1/d$. Later, we show how to get rid of this assumption by a simple scaling.

A.1 Partitioning a Vertex Set

We start with two technical lemmas.

► **Lemma 15** (Lemma 5 of [7]). *Let $f : [r_0, r_1] \rightarrow \mathbb{R}$ be a nonnegative monotone increasing function that is differentiable almost everywhere. If the derivative f' is continuous almost everywhere, then there exists an $r \in (r_0, r_1)$ such that $f'(r)$ is defined and satisfies*

$$f'(r) \leq \frac{f(r)}{r_1 - r_0} \cdot \ln\left(\frac{e \cdot f(r_1)}{f(r)}\right) \cdot \ln\ln\left(\frac{e \cdot f(r_1)}{f(r_0)}\right).$$

► **Lemma 16.** *Fix $0 < x \leq y$. Then,*

$$\ln\left(\frac{e \cdot (x + y)}{2 \cdot x}\right) \leq \frac{1}{\ln 2} \cdot \ln\left(\frac{x + y}{x}\right).$$

Proof. Since $y \geq x$, we have $\ln((x + y)/x) \geq \ln 2$. Therefore,

$$\begin{aligned} \ln\left(\frac{e \cdot (x + y)}{2 \cdot x}\right) &= \ln\left(\frac{x + y}{x}\right) + \left(\frac{1}{\ln 2} - 1\right) \cdot \ln 2 \\ &\leq \ln\left(\frac{x + y}{x}\right) + \left(\frac{1}{\ln 2} - 1\right) \cdot \ln\left(\frac{x + y}{x}\right) \\ &= \frac{1}{\ln 2} \cdot \ln\left(\frac{x + y}{x}\right). \end{aligned} \quad \blacktriangleleft$$

► **Lemma 17.** *Fix a real $d > 0$. Fix a graph $G = (V, E, c)$ with edge costs c satisfying $c(e) \geq 1/d$ for every $e \in E$. Let $\{z_e\}_{e \in E}$ be the (non-negative) lengths of edges. For a subset of vertices $U \subseteq V$ satisfying $\text{diam}_z(U) \geq d$, it is possible to partition U , in polynomial time, into two disjoint non-empty sets U_L and U_R , satisfying the following conditions.*

15:16 An Improved Approximation Algorithm for Dynamic Minimum Linear Arrangement

- If $G[U]$ is disconnected, then $c(E[U_L, U_R]) = 0$ and $\text{vol } U_L + \text{vol } U_R = \text{vol } U$.
- If $G[U]$ is connected, then

$$c(E[U_L, U_R]) \cdot \text{diam}_z(U) \leq g \cdot \beta \cdot \ln\left(\frac{\text{vol } U}{\beta}\right) \cdot \ln \ln(\text{vol } V),$$

where g is a universal constant, and β satisfies the following constraints:

$$\begin{aligned} \beta &\geq \max\{\text{vol } U_L, 1/4\}, \\ \text{vol } U - \beta &\geq \max\{\text{vol } U_R, 1/4\}. \end{aligned}$$

Proof. If $G[U]$ is disconnected, then we simply take U_L to be an arbitrary connected component of U and set $U_R = U \setminus U_L$. The lemma follows immediately as $c(E[U_L, U_R]) = 0$ and $\text{vol } U_L + \text{vol } U_R = \text{vol } U$. Thus in the following, we assume that $G[U]$ is connected.

We first show *the existence* of U_L and U_R satisfying the constraints above, and later we argue how to find them using a polynomial-time procedure.

For any two vertices $a, b \in U$, we define $\text{dist}_z^U(a, b)$ as the shortest-path distance between a and b that uses edges only from $G[U]$; we call it U -distance. Clearly, $\text{dist}_z^U(a, b) \geq \text{dist}_z(a, b)$. Let $u, u' \in U$ be the vertices satisfying $\text{dist}_z(u, u') = \text{diam}_z(U)$, and let

$$\Delta \triangleq \text{dist}_z^U(u, u')$$

be their U -distance. Then, $\Delta \geq \text{dist}_z(u, u') = \text{diam}_z(U) \geq d$. From this point on, we focus on the graph $G[U]$ only and use U -distances exclusively.

For a vertex $a \in U$ and a real $r \geq 0$, let

$$B_a(r) \triangleq \{v \in U \mid \text{dist}_z^U(a, v) < r\}$$

be the set of vertices in U whose U -distance to a is *strictly smaller* than r , i.e., contained in a ball centered at a of radius r . For an edge $e \in E[U]$, we define its (a, r) -adjusted length as

$$z_e^a(r) \triangleq \begin{cases} z_e & \text{if } e \in B_a(r) \times B_a(r), \\ r - \text{dist}_z^U(a, v_1) & \text{if } e = (v_1, v_2) \in B_a(r) \times (U \setminus B_a(r)), \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, the (a, r) -adjusted length of an edge e is the length of e “contained” (entirely or partially) in the ball of radius r centered at a .

Next, we introduce the function $\text{vol}_a^* : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ as

$$\begin{aligned} \text{vol}_a^*(r) &\triangleq \sum_{e \in E[U]} c(e) \cdot z_e^a(r) \\ &= \text{vol}(B_a(r)) + \sum_{e=(v_1, v_2) \in E \cap (B_a(r) \times (U \setminus B_a(r)))} c(e) \cdot (r - \text{dist}_z^U(a, v_1)). \end{aligned}$$

As $\Delta = \text{dist}_z^U(u, u')$, the balls of radii $\Delta/2$ centered at u and u' are disjoint. In particular, for each edge $e \in E[U]$, it holds that $z_e^u(\Delta/2) + z_e^{u'}(\Delta/2) \leq z_e$. This implies that $\text{vol}_u^*(\Delta/2) + \text{vol}_{u'}^*(\Delta/2) \leq \text{vol } U$. Without loss of generality, we may assume that $\text{vol}_u^*(\Delta/2) \leq \text{vol } U/2$; otherwise we may swap the roles of u and u' .

As $G[u]$ is connected, $\text{vol}_u^*(r)$ is monotonically increasing in the interval $[0, \Delta]$. Moreover, except finitely many points it holds that

$$\frac{\partial \text{vol}_u^*(r)}{\partial r} = c(B_u(r) \times (U \setminus B_u(r))).$$

That is, $\text{vol}_u^*(r)$ satisfies the conditions of Lemma 15 as the function of r in the interval $[0, \Delta]$, and thus also in the interval $[\Delta/4, \Delta/2]$. By Lemma 15, there exists $r^* \in (\Delta/4, \Delta/2)$, such that

$$c(E[B_u(r^*), U \setminus B_u(r^*)]) \leq \frac{\text{vol}_u^*(r^*)}{\Delta/4} \cdot \ln\left(\frac{e \cdot \text{vol}_u^*(\Delta/2)}{\text{vol}_u^*(r^*)}\right) \cdot \ln \ln\left(\frac{e \cdot \text{vol}_u^*(\Delta/2)}{\text{vol}_u^*(\Delta/4)}\right). \quad (8)$$

Recall that $G[U]$ is connected and contains a path between u and u' of length $\Delta \geq d$. As $c(e) \geq 1/d$ for every edge e , we have $\text{vol } U \geq \Delta \cdot (1/d) \geq 1$. Similarly, $\text{vol}_u^*(\Delta/4) \geq (\Delta/4) \cdot (1/d) \geq 1/4$. We set $U_L = B_u(r^*)$ and $U_R = U \setminus U_L$. Furthermore, we set $\beta = \text{vol}_u^*(r^*)$. By plugging these values into (8), and recalling that $\text{vol}_u^*(\Delta/2) \leq \text{vol } U/2$, we obtain

$$\begin{aligned} c(E[U_L, U_R]) \cdot \Delta &\leq 4 \cdot \beta \cdot \ln\left(\frac{e \cdot \text{vol } U}{2 \cdot \beta}\right) \cdot \ln \ln(2e \cdot \text{vol } U) \\ &\leq \frac{4}{\ln 2} \cdot \beta \cdot \ln\left(\frac{\text{vol } U}{\beta}\right) \cdot \ln \ln(2e \cdot \text{vol } U). \quad (\text{by Lemma 16 and } U \subseteq V) \end{aligned}$$

It remains to argue that β satisfies the constraints of the lemma.

- By the definition of vol_u^* , we have $\beta = \text{vol}_u^*(r^*) \geq \text{vol}(B_u(r^*)) = \text{vol } U_L$.
- By the monotonicity of vol_u^* , we have $\beta = \text{vol}_u^*(r^*) \geq \text{vol}_u^*(\Delta/4) \geq 1/4$.
- By the monotonicity of vol_u^* , we have $\beta = \text{vol}_u^*(r^*) \leq \text{vol}_u^*(\Delta/2) \leq \text{vol } U/2$, and thus $\text{vol } U - \beta \geq \text{vol } U/2 \geq 1/2 > 1/4$.
- Observe that $z_e^u(r^*) = 0$ for every edge $e \in E[U_R]$. (This follows as both endpoints of e are from U_R , and thus their distance to u is at least r^* .) In effect, $\text{vol}_u^*(r^*) + \text{vol } U_R \leq \text{vol } U$, or equivalently $\text{vol } U - \beta \geq \text{vol } U_R$.

The argument above shows the *existence* of r^* , such that setting $\beta = \text{vol}_u^*(r)$ satisfies the constraints of the lemma. To make the argument constructive and efficient, we note that the left-hand side of (8) is constant except for at most $|U|$ values of r^* (more concretely, these are values from the set $D_u = \{\text{dist}_z(u, v) \mid v \in U\}$) and the right-hand side is monotonically increasing in the interval $[\Delta/4, \Delta/2]$. Thus, it is sufficient to look for r^* only in the set $D_u \cup \{\Delta/2\}$. ◀

A.2 Using Set Partitioning for Tree Decomposition

As stated earlier, the d -decomposition tree $\mathcal{T} = (V_T, E_T, \alpha)$ of graph $G = (V, E, c)$ with edge lengths $\{z_e\}_{e \in E}$ is obtained by the iterative application of Lemma 17, starting at V and terminating with leaves corresponding to sets whose diameter is less than d .

Throughout this section, we use the function $F : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ defined as

$$F(x) \triangleq \max\{x \cdot \ln(4x), 0\}.$$

That is, $F(x) = 0$ for $x \leq 1/4$ and $F(x) = x \cdot \ln(4x)$ for $x \geq 1/4$. Note that F is monotonically non-decreasing. Moreover, it satisfies the following superadditivity property.

► **Lemma 18.** *Fix $x, y \geq 0$. Then, $F(x) + F(y) \leq F(x + y)$. Moreover, if $x \geq 1/4$ and $y \geq 1/4$, then $F(x) + F(y) + x \cdot \ln((x + y)/x) \leq F(x + y)$.*

Proof. If $x < 1/4$, then $F(x) = 0$, and hence the lemma follows by monotonicity of F . The case when $y < 1/4$ is analogous. Hence, we may now assume that $x \geq 1/4$ and $y \geq 1/4$. Using the definition of F ,

$$\begin{aligned} F(x + y) - F(x) - F(y) &= x \cdot (\ln(4x + 4y) - \ln(4x)) + y \cdot (\ln(4x + 4y) - \ln(4y)) \\ &\geq x \cdot \ln\left(\frac{x + y}{x}\right) + y \cdot 0. \end{aligned} \quad \blacktriangleleft$$

15:18 An Improved Approximation Algorithm for Dynamic Minimum Linear Arrangement

► **Lemma 19.** For a node $w \in V_T$, let $V_T(w)$ be the set of nodes of a subtree of \mathcal{T} rooted at w . Let

$$A(w) \triangleq \frac{1}{g \cdot \ln \ln(\text{vol } V)} \cdot \sum_{w' \in V_T(w)} c(\text{cut}(w')) \cdot \text{diam}_z(\alpha(w')),$$

where g is the universal constant of Lemma 17. Then, $A(w) \leq F(\text{vol } \alpha(w))$.

Proof. For brevity, for a node $w \in V_T$, we use $\text{vol } w$ instead of $\text{vol } \alpha(w)$.

We show the lemma by induction. For the induction base (w is a leaf), we observe that $\text{cut}(w) = \emptyset$, and thus $c(\text{cut}(w)) = 0$. Hence, $A(w) = 0 \leq F(\text{vol } w)$.

For the induction step, we consider an internal node $w \in V_T$ with children w_L and w_R . We assume that the lemma inequality follows for both w_L and w_R , and we prove it for w . By the definition of A it follows that

$$A(w) = A(w_L) + A(w_R) + \frac{1}{g \cdot \ln \ln(\text{vol } V)} \cdot c(\text{cut}(w)) \cdot \text{diam}_z(\alpha(w)). \quad (9)$$

We consider two cases depending on whether $G[\alpha(w)]$ is connected or not.

■ If $G[\alpha(w)]$ is disconnected, Lemma 17 implies $c(\text{cut}(w)) = 0$ and $\text{vol}(w) = \text{vol}(w_L) + \text{vol}(w_R)$. Thus,

$$\begin{aligned} A(w) &= A(w_L) + A(w_R) && \text{(applying } c(\text{cut}(w)) = 0 \text{ to (9))} \\ &\leq F(\text{vol } w_L) + F(\text{vol } w_R) && \text{(by the inductive assumption)} \\ &\leq F(\text{vol } w_L + \text{vol } w_R) = F(\text{vol } w). && \text{(by Lemma 18)} \end{aligned}$$

■ If $G[\alpha(w)]$ is connected, Lemma 17 guarantees that

$$c(\text{cut}(w)) \cdot \text{diam}_z(\alpha(w)) \leq g \cdot \beta_w \cdot \ln\left(\frac{\text{vol } w}{\beta_w}\right) \cdot \ln \ln(\text{vol } V), \quad (10)$$

where β_w satisfies $\max\{\text{vol } w_L, 1/4\} \leq \beta_w$ and $\max\{\text{vol } w_R, 1/4\} \leq \text{vol } w - \beta_w$. Thus,

$$\begin{aligned} A(w) &\leq A(w_L) + A(w_R) + \beta_w \cdot \ln(\text{vol } w / \beta_w) && \text{(applying (10) to (9))} \\ &\leq F(\text{vol } w_L) + F(\text{vol } w_R) + \beta_w \cdot \ln(\text{vol } w / \beta_w) && \text{(by the inductive assumption)} \\ &\leq F(\beta_w) + F(\text{vol } w - \beta_w) + \beta_w \cdot \ln(\text{vol } w / \beta_w) && \text{(by the monotonicity of } F) \\ &\leq F(\beta_w + \text{vol } w - \beta_w) = F(\text{vol } w). && \text{(by Lemma 18)} \end{aligned}$$

As in both cases $A(w) \leq F(\text{vol } w)$, this concludes the inductive proof. ◀

A.3 Bounding Cost of Decomposition Tree

Before we prove Theorem 4, we prove its variant, where we assume that $\min_{e \in E} c(e) \geq 1/d$. Theorem 4 follows then by scaling as a simple corollary.

► **Lemma 20.** Fix a real $d > 0$, a graph $G = (V, E, c)$ such that $c(e) \geq 1/d$ for every $e \in E$, and non-negative edge lengths $\{z_e\}_{e \in E}$. It is possible to construct, in polynomial time, a d -decomposition tree $\mathcal{T} = (V_T, E_T, \alpha)$ of G , such that

$$\text{COST}_{G,z}(\mathcal{T}) \leq \xi \cdot O(\log \xi \cdot \log \log \xi),$$

where $\xi = \sum_{e \in E} c(e) \cdot z_e$.

Proof. We construct a d -decomposition tree \mathcal{T} as described in the previous section. Let w_r be its root. Using Definition 3,

$$\begin{aligned} \text{COST}_{G,z}(\mathcal{T}) &= \sum_{w \in V_{\mathcal{T}}(w_r)} c(\text{cut}(w)) \cdot \text{diam}_z(\alpha(w)) \\ &\leq O(1) \cdot F(\text{vol}(\alpha(w_r))) \cdot \ln \ln(\text{vol } V) && \text{(by Lemma 19)} \\ &= \text{vol } V \cdot O(\ln(\text{vol } V) \cdot \ln \ln(\text{vol } V)). && \text{(by the definition of } F) \end{aligned}$$

The lemma follows by observing that $\text{vol } V = \xi$. ◀

► **Theorem 4.** Fix a real $d > 0$ and a graph $G = (V, E, c)$ with non-negative edge lengths $\{z_e\}_{e \in E}$. It is possible to construct, in polynomial time, a d -decomposition tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}}, \alpha)$ of G , such that

$$\text{COST}_{G,z}(\mathcal{T}) \leq \xi \cdot O(\log(f_{c,d} \cdot \xi) \cdot \log \log(f_{c,d} \cdot \xi)),$$

where $\xi = \sum_{e \in E} c(e) \cdot z_e$, $f_{c,d} = \max\{1, 1/(d \cdot c_{\min})\}$, and $c_{\min} = \min_{e \in E} c(e)$.

Proof. Let $c_{\text{thr}} = 1/d$. Note that $f_{c,d} = \max\{1, c_{\text{thr}}/c_{\min}\}$. If $c_{\min} \geq c_{\text{thr}}$, then G satisfies the requirements of Lemma 20 and $f_{c,d} = 1$. The theorem follows immediately by invoking this lemma.

In the following, we thus assume that $c_{\min} < c_{\text{thr}}$. In this case, $f_{c,d} = c_{\text{thr}}/c_{\min}$. We take the graph $G' = (V, E, c')$, where $c'(e) = (c_{\text{thr}}/c_{\min}) \cdot c(e)$ for every edge $e \in E$. We construct a d -decomposition tree \mathcal{T} of G' using Lemma 20. We now argue that \mathcal{T} satisfies the theorem statement with respect to the original graph G . Let $\xi' = \sum_{e \in E} c'(e) \cdot z_e$. Then,

$$\begin{aligned} \text{COST}_{G,z}(\mathcal{T}) &= (c_{\min}/c_{\text{thr}}) \cdot \text{COST}_{G',z}(\mathcal{T}) && \text{(by Definition 3)} \\ &\leq (c_{\min}/c_{\text{thr}}) \cdot \xi' \cdot O(\log \xi' \cdot \log \log \xi') && \text{(by Lemma 20)} \\ &= \xi \cdot O(\log((c_{\text{thr}}/c_{\min}) \cdot \xi) \cdot \log \log((c_{\text{thr}}/c_{\min}) \cdot \xi)) \\ &= \xi \cdot O(\log(f_{c,d} \cdot \xi) \cdot \log \log(f_{c,d} \cdot \xi)). && \text{◀} \end{aligned}$$