

Learning Minimum Linear Arrangement of Cliques and Lines*

Julien Dallot¹, Maciej Pacut¹, Marcin Bienkowski², Darya Melnyk¹, and Stefan Schmid¹

¹Technische Universität Berlin, Germany

²University of Wrocław, Poland

Abstract

In the well-known Minimum Linear Arrangement problem (MinLA), the goal is to arrange the nodes of an undirected graph into a permutation so that the total stretch of the edges is minimized. This paper studies an online (learning) variant of MinLA where the graph is not given at the beginning, but rather revealed piece-by-piece. The algorithm starts in a fixed initial permutation, and after a piece of the graph is revealed, the algorithm must update its current permutation to be a MinLA of the subgraph revealed so far. The objective is to minimize the total number of swaps of adjacent nodes as the algorithm updates the permutation.

The main result of this paper is an online randomized algorithm that solves this online variant for the restricted cases where the revealed graph is either a collection of cliques or a collection of lines. We show that the algorithm is $8 \ln n$ -competitive, where n is the number of nodes of the graph. We complement this result by constructing an asymptotically matching lower bound of $\Omega(\ln n)$.

1 Introduction

Minimum Linear Arrangement (MinLA) [29, 34, 23, 23, 27] is a classic combinatorial optimization problem that arises in the field of graph theory and network design. The objective is to arrange the nodes of an undirected graph along a straight line in such a way that the sum of the distances between adjacent nodes is minimized. The problem finds applications in various domains, including VLSI circuit design, transportation network planning, and communication network layout [19, 6, 4]. The problem is formally defined as follows. Let $G = (V, E)$ be a graph with n nodes and m edges. The goal of the MinLA problem is to find a permutation π of the nodes of G so that the total distance between the edges is minimized. Formally, it amounts to finding $\pi \in S_n$ such that the quantity

$$\sum_{(x,y) \in E} |\pi(x) - \pi(y)|$$

*This paper has been supported by Polish National Science Centre grant 2022/45/B/ST6/00559 and the Austrian Science Fund (FWF) project I 5025-N (DELTA). For the purpose of Open Access, the authors have applied CC-BY public copyright license to any Author Accepted Manuscript (AAM) version arising from this submission.

is minimized, where $\pi(x)$ denotes the position of node x in the permutation π .

In this paper, we study the online *learning* variant of the MinLA problem. In the online learning variant, the graph is not given right away but rather revealed piece-by-piece in a potentially adversarial order. We call G_0, G_1, \dots, G_k the sequence of the revealed graphs, where G_0 is the empty graph, each G_i is a super-graph of its predecessors, and $G_k = G$. The online algorithm starts with a given initial permutation π_0 and, upon receiving graph G_i , it *must* update its current permutation π_{i-1} , so that *the new permutation π_i is a MinLA of G_i* . The goal of the online MinLA problem is to find a sequence of permutations π_0, \dots, π_k , such that the total cost of the permutation updates is minimized. More formally, the algorithm must minimize the quantity

$$\sum_{i=0}^{k-1} d(\pi_i, \pi_{i+1}),$$

where d is the cost to update a permutation. We choose d to be Kendall's τ distance, that is, the minimum number of pairwise swaps of adjacent elements required to change one permutation into the other.

We use the notion of competitive ratio to assess the performance of our online algorithms [9]. For a given request sequence σ , let $\text{ONL}(\sigma)$ and $\text{OPT}(\sigma)$ be the cost incurred by an online algorithm ONL and an optimal offline algorithm OPT , respectively. In contrast to ONL , OPT knows the entire request sequence σ ahead of time. The goal is to design online algorithms for online learning MinLA that provide worst-case guarantees. In particular, ONL is said to be c -competitive if for any input sequence σ it holds that $\text{ONL}(\sigma) \leq c \cdot \text{OPT}(\sigma)$.

This work aligns with a broader line of work that searches for efficient online algorithms for classic offline optimization problems. As dealing with an unknown future is a ubiquitous scenario, there exists an extensive literature on online variants of well-known problems, including set cover [1, 10], graph matching [22, 17], graph coloring [35, 14], load balancing [28], maximum independent set [18], and bin packing [33, 5]. So far, the MinLA problem has received considerable attention in the offline setting [34, 23, 13, 27] either due to its many application cases [19, 6, 4] or its theoretical importance [2, 15], and developing a competitive online algorithm is a natural step in this line of research.

Our work is related the closest to work of Olver et al. [25], who introduced the *dynamic* MinLA problem and its variants (such as the itinerant list update problem). Their setting is less restricted than ours and it leaves more freedom on how to serve a request: when an edge is revealed in their variant, an algorithm simply pays the distance between its endpoints in its current permutation, and then may optionally rearrange the nodes to save cost on future requests, the goal being to minimize the total incurred cost. In particular, in their dynamic setting an algorithm does not have to maintain a MinLA of the graph induced by edges seen so far while our setting is learning the graph and remembers previous edges.

Their problem received the most attention in its offline setting: Olver et al. [25] presented a randomized $(\log^2 n)$ -approximation algorithm, later improved by Bienkowski and Even to a deterministic algorithm with a $(\log n \log \log n)$ -approximation [8]. While the online variant of the dynamic MinLA problem is regularly mentioned [25, 8], no advances were made on this front; the only results so far were a lower bound of $\Omega(\log n)$ for the competitive ratio of any randomized online algorithm [25], and a trivial upper bound of $O(n)$ held by the algorithm that never changes its permutation.

To the best of our knowledge, this paper hence is the first to tackle a variant of the problem in the online setting hopefully paving the way towards more general results for the online dynamic MinLA problem.

1.1 Contributions

We consider online learning MinLA problem in a restricted case where each graph G_i in the input is either a collection of disjoint cliques or a collection of disjoint lines (paths). This restriction captures fundamental questions that arise in a general setting: (1) where to move the two disjoint components that have just been connected by revealing of a new piece of the graph, and (2) which orientation should the connected component have in the permutation.

We develop two online algorithms for this setting: a deterministic one and a randomized one. We begin with the deterministic online algorithm, prove it is $2n$ -competitive and show that the analysis is tight. The main contribution of this paper, however, is the randomized online algorithm that is $8 \ln n$ -competitive against the oblivious adversary [9]. Finally, we show a randomized lower bound of $\Omega(\log n)$, proving that our algorithm is asymptotically optimal.

1.2 Motivation: Dynamic Virtual Network Embedding

With the increasing popularity of data-centric workloads, such as distributed machine learning and scale-out databases, datacenters are witnessing a significant surge in network traffic [32]. The performance of these distributed applications heavily relies on the efficiency of the underlying communication networks [24].

An intriguing approach to improve the efficiency of communication networks is to adjust the network in a demand-aware and online manner. This can be done by leveraging resource allocation flexibilities provided by virtualization. When communication requests arrive over time, algorithms can decide to move frequently communicating nodes topologically closer. Since relocation comes at a cost, the task is to strike a balance between the benefits and the costs of the adjustments. The underlying optimization problem involves dynamically embedding a virtual network on a physical network topology.

The basic virtual network embedding problem of finding a mapping from a request graph to the virtual network is well-studied [16, 30, 11, 12, 31], and many problem variants have been shown to be \mathcal{NP} -complete [16, 30, 11]. One of the most studied problems in graph embedding is the minimum linear arrangement problem. This problem is \mathcal{NP} -hard in general, therefore, significant efforts have been made to solve it for restricted guest graphs such as trees [34] or incomplete hypercubes [23].

The problem variant of *dynamically* embedding a virtual network on a physical network topology is, however, more challenging. So far, the dynamic problem has been considered for two fundamental physical network topologies: (1) a set of capacitated clusters connected in a clique and (2) the line topology. In the first setting, the problem is known as the *online graph partitioning* problem [3, 21, 20, 26], and in the latter setting, it is known as the *dynamic minimum linear arrangement* problem (a.k.a. itinerant list update problem [25]).

The online learning MinLA model is inspired by the learning variant of online graph partitioning, introduced at SIGMETRICS 2019 [21], with followup work at INFOCOM 2020 [26] and SODA 2021 [20]. The goal is to learn and embed the communication pattern (the guest

graph) over time in an online manner.

We develop an *optimally competitive* randomized online algorithms for online learning MinLA for two fundamental traffic patterns: collection of lines and collection of cliques. These topologies represent the most sparse and most dense traffic patterns and are an important stepping stone toward more complex network topologies.

1.3 Related Work

The model introduced in this paper is related to two online problems: *online graph partitioning*, introduced at DISC 2016 [3], and *dynamic minimum linear arrangement*, introduced at WAOA 2018 [25].

In the online graph partitioning problem [3], a network topology connecting ℓ clusters each containing k virtual nodes is considered. The requests concern pairs of nodes, and if the requested nodes are collocated within a single cluster, the request costs 0, and otherwise, it costs 1. For this problem, the best known competitive ratio is $(k \cdot \ell \cdot 2^{O(k)})$ -competitive [7], and no deterministic algorithm can be better than $\Omega(k \cdot \ell)$ -competitive [26]. Significant research efforts on the graph partitioning problem have focused on the learning variant [21, 26, 20].

In the dynamic minimum linear arrangement problem [25], the underlying topology is a line and the requests are pairs of virtual nodes. The cost for serving a request is proportional to the distance between the corresponding virtual nodes, but collocation is not strictly enforced. A $(\log n \cdot \log \log n)$ -approximation algorithm for this problem has been derived in the offline setting, where all the requests are known in advance [25, 8]. The paper of [25] shows a randomized lower-bound of $\Omega(\log n)$ for the dynamic MinLA problem in the online setting. So far, the only general upper bound is a trivial $O(n)$ bound, which can be achieved even by an algorithm that never migrates (however we point out that in the model studied in this paper, applying this algorithm is not viable). A simple online algorithm that moves the smaller component towards the larger has an upper bound of $O(n^2 \cdot \log n)$ on the total incurred cost [4], but under strict competitive analysis this algorithm is $\Omega(n)$ -competitive, and neither the algorithm nor the analysis translates to our setting.

1.4 Organization

This manuscript is organized as follows. In Section 2, we design and analyze a simple deterministic algorithm that attains a linear competitive ratio for the case of embedding a collection of lines or a collection of cliques. In Section 3, we design a randomized algorithm attaining a logarithmic competitive ratio for the case of embedding a collection of cliques. In Section 4, we adapt the previous algorithm to attain a logarithmic competitive ratio for the case of embedding a collection of lines. In Section 5, we prove the tightness of these algorithms. We conclude in Section 6.

2 A Deterministic Algorithm

In this section, we present a simple $O(n)$ -competitive deterministic algorithm DET for solving online MinLA for both cases where the subgraphs G_i either are collections of cliques or collection of lines. In Section 5 we show that the analysis of DET algorithm is tight for both graph

classes. This algorithm is an adaptation of an algorithm Perfect Partition Learner for the *online graph partitioning problem* [26].

The algorithm `DET` is defined as follows. Upon each request G_i , `DET` updates the permutation to an arbitrary MinLA of G_i that minimizes the distance to π_0 .

In the rest of this section, we argue that the competitive ratio of this algorithm is linear in terms of the number of nodes.

Theorem 1. *DET is $(2n - 2)$ -competitive for the case where the graph is either a collection of lines or a collection of cliques.*

Proof. Fix a request sequence G_0, G_1, \dots, G_k , an initial permutation π_0 and an optimal offline algorithm `OPT` with final permutation π_k^{OPT} . We denote $d(\pi_0, \pi_k^{\text{OPT}})$ by Δ^* .

First, we claim that for any permutation π_i reached by `DET` throughout its execution, we have $d(\pi_0, \pi_i) \leq \Delta^*$. Notice that, whether the subgraphs are collections of cliques or collections of lines, a permutation that is a MinLA of G_k is always also a MinLA of any G_i for $i \leq k$. As a result, `DET` considers π_k^{OPT} as a potential next permutation each time a new subgraph is revealed. Since `DET` picks the permutation that minimizes its distance from π_0 , we directly derive that $d(\pi_0, \pi_i) \leq \Delta^*$.

Second, we claim that the cost of serving each request by `DET` is at most $2 \cdot \Delta^*$. We argue as follows. Consider an update from a permutation π_{i-1} to π_i . In the previous paragraph, we argued that both these permutations are no further than Δ^* away from π_0 . Using the triangle inequality we have that $d(\pi_{i-1}, \pi_i) \leq d(\pi_{i-1}, \pi_0) + d(\pi_0, \pi_i) \leq 2 \cdot \Delta^*$.

Finally, we bound the competitive ratio. We have that `OPT` $\geq d(\pi_0, \pi_k^{\text{OPT}}) = \Delta^*$, as both `OPT` and `DET` start in the same permutation. The sequence of subgraphs G_1, \dots, G_k has length at most $n - 1$. In the previous paragraph, we argued that each request costs at most $2 \cdot \Delta^*$, hence the total cost of `DET` is at most $2(n - 1) \cdot \Delta^*$. Hence, $\text{DET}/\text{OPT} \leq 2(n - 1) \cdot \Delta^*/\Delta^* = 2(n - 1)$. \square

In Section 5, we show that our analysis of `DET` algorithm is tight.

3 An Optimal Randomized Algorithm for Cliques

3.1 Description of the Algorithm

In this section, we present a randomized algorithm `RAND` for the MinLA problem when the subgraphs G_0, \dots, G_k all are collections of cliques. We will prove that this algorithm holds an expected competitive ratio of $4 \ln n$ against the oblivious adversary [9].

Let X_i and Z_i be the nodes of the two components (cliques) that merge between G_i and G_{i+1} . In response to the reveal of G_{i+1} , `RAND` handles the necessary update by placing X_i and Z_i next to each other in π_{i+1} . `RAND` only considers two of the potentially many possible permutations where X_i and Z_i end up next to each other: starting from π_i , either the nodes of X_i move in direction of Z_i or conversely, the nodes of Z_i move in direction of X_i .

Our algorithm chooses between those two choices by flipping a biased coin: X_i moves with probability $|Z_i|/(|X_i| + |Z_i|)$ and Z_i moves with probability $|X_i|/(|X_i| + |Z_i|)$. Figure 1 summarizes the possible actions of our algorithm.

In the rest of this section, we will prove the following theorem.

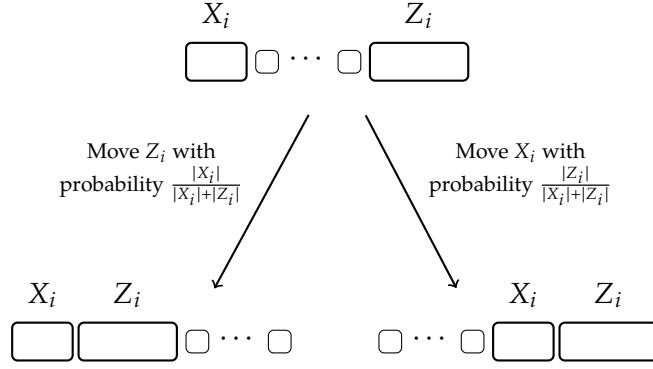


Figure 1: The two possible actions of RAND to move the merging components X_i and Z_i together when G_{i+1} is revealed

Theorem 2. *The algorithm RAND is $4 \ln n$ -competitive for online MinLA when all the revealed subgraphs are collections of cliques.*

To prove the theorem, we first introduce necessary notation.

3.2 Notation

Let C_i be the set of components of G_i , a component is the set of nodes of a connected component. We call $X_i, Z_i \in C_i$ the two components that merge into a bigger clique when G_{i+1} is revealed.

For $i \in \{0 \dots k\}$, let $T, U \in C_i$ be two components. For these, we define:

- $T \times U = \{(t, u) | t \in T, u \in U\}$ is the Cartesian product of T and U .
- $T \text{---} U$ is the event that T is on the left of U ; $P[T \text{---} U]$ is the associated probability that RAND is in a configuration with such order between the components.
- L_{π_i} is the set of all node pairs (x, y) such that x is on the left of y in π_i .
- $L_{T,U}$ is the set of all pairs that contain exactly one node from T and one node from U in any order, i.e., $L_{T,U} = T \times U \cup U \times T$.

Notice that the event $T \text{---} U$ is not indexed by i . This is because, by the properties of our algorithm, the relative position of two components T and U can change only if a node of T or U is requested; but if this happens then the requested component stops being a component (it becomes a part of a bigger one) and the event $T \text{---} U$ has no more meaning. Hence any two components will have only one relative position (if they happen to exist at the same moment) and the event $T \text{---} U$ is independent from i .

3.3 Upper Bound on the Expected Cost

We start the analysis with a lemma that gives the probability ruling the relative position of any two components throughout the execution of RAND.

Lemma 3. *Let $i \in \{0 \dots k\}$, let X and Y be two components of C_i . It holds that*

$$P[X \text{---} Y] = \frac{|X \times Y \cap L_{\pi_0}|}{|X| \cdot |Y|}.$$

Proof. We prove the claim by induction on i . If $i = 0$ then nothing was revealed so far, the components are the nodes themselves and the claim holds.

Let $i \in \{1 \dots k\}$, we assume that the claim holds for all components when G_i is revealed. We then prove that the claim also holds after G_{i+1} was revealed. Let Y be a component different from the merging components X_i and Z_i . We compute the probability that $(X_i \cup Z_i) \text{---} Y$, i.e., that the newly formed component $X_i \cup Z_i$ is on the left of Y after X_i merged with Z_i .

$$\begin{aligned} P[(X_i \cup Z_i) \text{---} Y] &= P[X_i \text{---} Y] \cdot P[Z_i \text{---} Y] + P[X_i \text{---} Y] \cdot P[Y \text{---} Z_i] \cdot \frac{|X_i|}{|X_i| + |Z_i|} \\ &\quad + P[Z_i \text{---} Y] \cdot P[Y \text{---} X_i] \cdot \frac{|Z_i|}{|X_i| + |Z_i|} \\ &= P[X_i \text{---} Y] \cdot \frac{|X_i|}{|X_i| + |Z_i|} + P[Z_i \text{---} Y] \cdot \frac{|Z_i|}{|X_i| + |Z_i|} \\ &= \frac{|(X_i \cup Z_i) \times Y \cap L_{\pi_0}|}{|X_i \cup Z_i| \cdot |Y|}. \end{aligned}$$

We applied the induction hypothesis to obtain the final result. By symmetry, we have the same results for computing $P[Y \text{---} (X_i \cup Z_i)]$. The induction holds and the claim follows. \square

Lemma 3 implies that, at any moment, the probabilistic relative position of any two components of RAND's permutation only depends on the initial permutation after G_i was revealed. In particular, the probability distribution of the permutation π_i is independent on the reveal order.

We now use this lemma to upper-bound the cost of an update. For any set of nodes U and T , we call $M_{U,T}^i$ the total number of swaps that occurs between a node of U and a node of T as RAND updates its permutation from π_i to π_{i+1} .

Lemma 4. *Let $0 \leq i \leq k - 1$, let $Y \in C_i$ different from the merging components X_i and Z_i . It holds that*

$$\begin{aligned} \frac{1}{2} \mathbb{E}[M_{Y, X_i \cup Z_i}^i] &\leq |(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{X_i, Y}| \cdot \frac{|Z_i|}{|X_i| + |Z_i|} \\ &\quad + |(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{Z_i, Y}| \cdot \frac{|X_i|}{|X_i| + |Z_i|}. \end{aligned}$$

Proof. The nodes of Y will swap with nodes of X_i or Z_i if and only if Y is between X_i and Z_i when G_{i+1} is revealed. Expressed in the terms of Lemma 3 this condition becomes: the nodes of Y will swap with those of X_i or Z_i only if X_i is on the left of Y and Z_i on the right, or conversely.

Multiplying the expected cost in case Y crosses one of the merging components by the probability that Y is between X_i and Z_i , we obtain that

$$\mathbb{E}[M_{Y, X_i \cup Z_i}^i] = 2|Y| \frac{|Z_i| \cdot |X_i|}{|X_i| + |Z_i|} \cdot (P[X_i \text{---} Y] \cdot P[Y \text{---} Z_i] + P[Z_i \text{---} Y] \cdot P[Y \text{---} X_i]).$$

In π_k^{OPT} , the nodes from $X_i \cup Z_i$ occupy contiguous places. Hence, Y is either on the right or on the left of $X_i \cup Z_i$ in π_k^{OPT} ; using the problem's symmetries we assume that Y is on the left of $X_i \cup Z_i$ in π_k^{OPT} without loss of generality. We now upper-bound the probabilities $P[Y \text{---} X_i]$ and $P[Y \text{---} Z_i]$ in the last equation by 1 and apply Lemma 3:

$$\begin{aligned} \frac{1}{2}\mathbb{E}[M_{Y, X_i \cup Z_i}^i] &\leq |Y| \cdot \frac{|Z_i| \cdot |X_i|}{|X_i| + |Z_i|} \cdot (P[X_i - Y] + P[Z_i - Y]) \\ &= |X_i \times Y \cap L_{\pi_0}| \cdot \frac{|Z_i|}{|X_i| + |Z_i|} + |Z_i \times Y \cap L_{\pi_0}| \cdot \frac{|X_i|}{|X_i| + |Z_i|}. \end{aligned}$$

All node pairs in $X_i \times Y \cap L_{\pi_0}$ have a node from X_i on the left and a node from Y on the right. Likewise, all pairs in $Z_i \times Y \cap L_{\pi_0}$ have a node from Z_i on the left part and a node from Y on the right part. Hence, neither of those two sets share a pair with $Y \times X_i$, $Y \times Z_i$ or π_k^{OPT} . As a result, it holds that

$$|X_i \times Y \cap L_{\pi_0}| = |(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{X_i, Y}|$$

and

$$|Z_i \times Y \cap L_{\pi_0}| = |(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{Z_i, Y}|,$$

which ends the proof. \square

We state an auxiliary lemma before stating the final theorem.

Lemma 5. *Let s_1, s_2, \dots, s_N be a series of strictly positive natural numbers, let $S = \sum_{i=1}^N s_i$ be their sum. It holds that*

$$\sum_{i=1}^N \frac{s_i}{\sum_{j=1}^i s_j} \leq H_S.$$

where $H_S = 1 + \frac{1}{2} + \dots + \frac{1}{S}$ is the harmonic sum.

Proof.

$$\sum_{i=1}^N \frac{s_i}{\sum_{j=1}^i s_j} = \sum_{i=1}^N \sum_{k=1}^{s_i} \frac{1}{\sum_{j=1}^i s_j} \leq \sum_{i=1}^N \sum_{k=1}^{s_i} \frac{1}{\sum_{j=1}^{i-1} s_j + k} = \sum_{i=1}^S \frac{1}{i}. \quad \square$$

Theorem 6. *Let M be the random variable equal to the cost of RAND. Then, $\mathbb{E}[M] \leq 4H_n \cdot |L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}|$, where H_n is the harmonic sum.*

Proof of Theorem 6. We use Lemma 4 to upper-bound M . It holds that

$$\begin{aligned} \frac{1}{2}\mathbb{E}[M] &= \sum_{i=0}^{k-1} \sum_{\substack{Y \in \\ \mathcal{C}_i \setminus \{X_i, Z_i\}}} \frac{1}{2}\mathbb{E}[M_{Y, X_i \cup Z_i}^i] \\ &\leq \sum_{i=0}^{k-1} \sum_{\substack{Y \in \\ \mathcal{C}_i \setminus \{X_i, Z_i\}}} \left(|(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{X_i, Y}| \cdot \frac{|Z_i|}{|X_i| + |Z_i|} + |(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{Z_i, Y}| \cdot \frac{|X_i|}{|X_i| + |Z_i|} \right) \\ &= \sum_{i=0}^{k-1} \left(|(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{X_i, V \setminus X_i}| \cdot \frac{|Z_i|}{|X_i| + |Z_i|} + |(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{Z_i, V \setminus Z_i}| \cdot \frac{|X_i|}{|X_i| + |Z_i|} \right). \end{aligned}$$

We now rewrite the above sum in terms of pairs rather than edges:

$$\frac{1}{2}\mathbb{E}[M] \leq \sum_{p \in L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}} \sum_{i=0}^{k-1} \left(\frac{|Z_i|}{|X_i| + |Z_i|} \cdot \mathbb{1}(p \in L_{X_i, V \setminus X_i}) + \frac{|X_i|}{|X_i| + |Z_i|} \cdot \mathbb{1}(p \in L_{Z_i, V \setminus Z_i}) \right). \quad (1)$$

Let p be a pair in $L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}$. p has a strictly positive coefficient for i in the above sum if and only if p has one of its nodes contained in X_i or Z_i . As a result, for each i so that p has a strictly positive coefficient in the sum, one of the components that contains a node of p merges into a bigger one in the next graph G_{i+1} .

Let y be a node in p , let $Y_1 \dots Y_N$ be the components that successively merge with y 's component — we call Y_0 the initial component containing only y . Summing all the coefficients corresponding to pair p and where y is in a merging component gives

$$\sum_{i=1}^N \frac{|Y_i|}{\sum_{j=0}^i |Y_j|},$$

which is lower than H_n using Lemma 5. There is such a sum for the two nodes of every pair $p \in L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}$ in inequality (1). Hence the claim holds. \square

Observation 7. Let OPT be an optimal offline algorithm and let π_k^{OPT} be its final permutation. Then, $c(\text{OPT}) \geq |L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}|$.

Proof. The claim directly follows from the fact that $d(\pi_0, \pi_k^{\text{OPT}}) = |L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}|$. \square

Combining Theorem 6 and Observation 7 and using the well-known relationships between the harmonic sum and the natural logarithm finally proves Theorem 2.

4 An Optimal Randomized Algorithm for Lines

4.1 Description of the Algorithm

In this section, we present a randomized algorithm RAND for the MinLA problem when the subgraphs G_0, G_1, \dots, G_k are collections of lines. We will prove that this algorithm holds an expected competitive ratio of $8 \ln n$ against the oblivious adversary [9].

Let x_i and z_i be the endpoints of the edge that is revealed between G_i and G_{i+1} . We call X_i and Z_i the components that contain x_i and z_i — a component refers to the nodes of a connected component. In response to G_{i+1} being revealed, our algorithm RAND handles the necessary update in two successive parts:

- first, RAND places X_i and Z_i next to each other;
- second, RAND places x_i and z_i next to each other.

We refer to the first part as the *moving* part of the update, and we refer to the second part as the *rearranging* part of the update. Next, we describe how our algorithm performs those two parts.

RAND performs the moving part exactly like in the clique case above: X_i moves with probability $|Z_i|/(|X_i| + |Z_i|)$ and Z_i moves with probability $|X_i|/(|X_i| + |Z_i|)$.

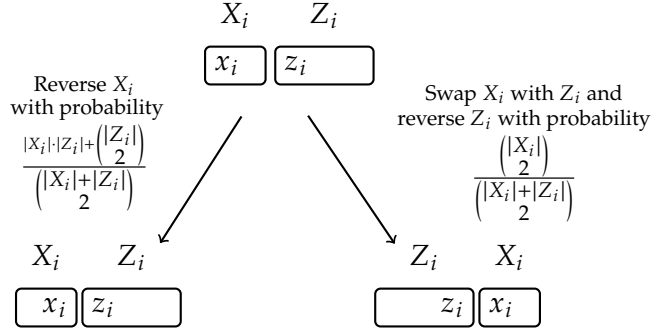


Figure 2: The two possible actions of RAND to rearrange the components X_i and Z_i together for a particular configuration of the merging components

In order to perform the rearranging part of the update, our algorithm has only two options, one for each of the two orientations of $X_i \cup Z_i$ in the permutation. Again, RAND makes its choices by flipping a biased coin weighted as follows: the probability to make one given choice equals the (normalized) cost of making the other choice. For a visual description, see Figure 2.

In the rest of this section, we will prove the following theorem.

Theorem 8. *The algorithm RAND is $8 \ln n$ competitive for the problem of online MinLA where the revealed subgraphs G_0, \dots, G_k are collections of lines.*

4.2 Additional Notation for the Proofs

Let x_i and z_i be the endpoints of the edge that is revealed between G_i and G_{i+1} for all $0 \leq i \leq k-1$. Let C_i be the set of components of G_i . We call $X_i, Z_i \in C_i$ the components that contain x_i and z_i respectively. If $T, U \in C_i$ are two components, we define:

- \vec{T} and \overleftarrow{T} are the events that T has one given orientation or the reversed one (an orientation of a component is the ordering of its nodes in the permutation). $P[\vec{T}]$ is the associated probability that RAND is in a configuration with the orientation \vec{T} .
- $L_{T,U}$ is the set of all pairs that contain exactly one node from T and one node from U in any order, i.e., $L_{T,U} = T \times U \cup U \times T$
- L_{π_i} is the set of all node pairs (x, y) such that x is on the left of y in π_i .
- $L_{\vec{T}}$ is the set of all pairs (t, t') in T such that t is on the left of t' when T has orientation \vec{T} .

4.3 Upper Bound on the Moving Costs

Calling M the random variable equal to the total moving costs of RAND, the Theorem 6 of the above clique case can also be applied and it holds that

$$\mathbb{E}[M] \leq 4 \ln n \cdot |L_{\pi_0} \setminus L_{\pi_k^{\text{Opt}}}|.$$

4.4 Upper Bound on the Rearranging Costs

In this subsection, we give an upper bound of the total cost incurred by the rearranging parts of the updates. We begin our proof by computing the probability of a component to be one or the other orientation (given in Lemma 10). Right below (Lemma 9) we present an auxiliary result used to derive Lemma 10.

Lemma 9. *Consider N real numbers a_1, a_2, \dots, a_N and N real numbers b_1, b_2, \dots, b_N each between 0 and 1, we call $\bar{b}_i = 1 - b_i$ for all $i \in [N]$. It holds that*

$$\sum_{t \in \{0,1\}^N} \left[\sum_{i=1}^N t_i \cdot a_i \right] \prod_{j=1}^N b_j^{t_j} \cdot \bar{b}_j^{\bar{t}_j} = \sum_{i=1}^N a_i \cdot b_i.$$

Proof.

$$\begin{aligned} \sum_{t \in \{0,1\}^N} \left[\sum_{i=1}^N t_i \cdot a_i \right] \prod_{j=1}^N b_j^{t_j} \cdot \bar{b}_j^{\bar{t}_j} &= \sum_{i=1}^N a_i \cdot \left[\sum_{t \in \{0,1\}^N} t_i \cdot \prod_{j=1}^N b_j^{t_j} \cdot \bar{b}_j^{\bar{t}_j} \right] \\ &= \sum_{i=1}^N a_i \cdot b_i \left[\sum_{\substack{t \in \{0,1\}^N \\ t_i=1}} \prod_{\substack{j=1 \\ j \neq i}}^N b_j^{t_j} \cdot \bar{b}_j^{\bar{t}_j} \right] \\ &= \sum_{i=1}^N a_i \cdot b_i \end{aligned}$$

The last equality was derived using the total probability formula. □

Lemma 10. *Let $1 \leq i \leq k$. For any $X \in \mathcal{C}_i$ containing strictly more than one node, it holds that*

$$P[\vec{X}] = \frac{|L_{\vec{X}} \cap L_{\pi_0}|}{\binom{|X|}{2}}.$$

Proof. We prove the claim by induction on i . The claim clearly holds for $i = 1$. Let $i \in \{1 \dots k\}$, we assume that the claim holds for i and prove it also holds for $i + 1$. We distinguish between cases depending on the size of the merging components X_i and Z_i .

- If both X_i and Z_i have size one, the claim clearly holds.
- If only one of X_i or Z_i have size one, say X_i . In the following, we assume that the

orientation $\overrightarrow{X_i \cup Z_i}$ refers to X_i being on the left of Z_i and Z_i having the orientation $\overrightarrow{Z_i}$.

$$\begin{aligned}
P[\overrightarrow{X_i \cup Z_i}] &= P[X_i \leftarrow Z_i] \cdot P[\overrightarrow{Z_i}] + \frac{|Z_i|}{\binom{|Z_i|+1}{2}} \cdot P[X_i \leftarrow Z_i] \cdot P[\overleftarrow{Z_i}] + \frac{\binom{|Z_i|}{2}}{\binom{|Z_i|+1}{2}} \cdot P[\overrightarrow{Z_i}] \cdot P[Z_i \leftarrow X_i] \\
&= \frac{|Z_i|}{\binom{|Z_i|+1}{2}} \cdot P[X_i \leftarrow Z_i] + \frac{\binom{|Z_i|}{2}}{\binom{|Z_i|+1}{2}} \cdot P[\overrightarrow{Z_i}] \\
&= \frac{|X_i \times Z_i \cap L_{\pi_0}| + |L_{\overrightarrow{Z_i}} \cap L_{\pi_0}|}{\binom{|Z_i|+1}{2}} \\
&= \frac{|L_{\overrightarrow{X_i \cup Z_i}} \cap L_{\pi_0}|}{\binom{|Z_i|+1}{2}}
\end{aligned}$$

which proves the claim. We used the induction hypothesis and Lemma 3 in the above calculations.

- Assume now that neither X_i nor Z_i have size one. To prove the results, we use Lemma 9 with the following values: $N = 3$, index a_1 refers to the cost to reverse X_i , a_2 is the cost to swap X_i and Z_i , a_3 is the cost to reverse Z_i , $b_1 = P[\overleftarrow{X_i}]$, $b_2 = P[Z_i \leftarrow X_i]$, $b_3 = P[\overleftarrow{Z_i}]$. With those parameters (and with the a_j s divided by $\binom{|X_i+Z_i|}{2}$), the left term of Lemma 9 equals $P[\overrightarrow{X_i \cup Z_i}]$, where $\overrightarrow{X_i \cup Z_i}$ stands for the orientation where $\overrightarrow{X_i}$, $X_i \leftarrow Z_i$ and $\overrightarrow{Z_i}$. Using Lemma 9, it therefore holds:

$$P[\overrightarrow{X_i \cup Z_i}] = \frac{1}{\binom{|X_i|+|Z_i|}{2}} \cdot \left(\binom{|X_i|}{2} \cdot P[\overrightarrow{X_i}] + |X_i||Z_i| \cdot P[X_i \leftarrow Z_i] + \binom{|Z_i|}{2} \cdot P[\overrightarrow{Z_i}] \right).$$

Using the induction hypothesis and Lemma 3, we obtain

$$\begin{aligned}
P[\overrightarrow{X_i \cup Z_i}] &= \frac{1}{\binom{|X_i|+|Z_i|}{2}} \cdot \left(|L_{\overrightarrow{X_i}} \cap L_{\pi_0}| + |X_i \times Z_i \cap L_{\pi_0}| + |L_{\overrightarrow{Z_i}} \cap L_{\pi_0}| \right) \\
&= \frac{|L_{\overrightarrow{X_i \cup Z_i}} \cap L_{\pi_0}|}{\binom{|X_i|+|Z_i|}{2}},
\end{aligned}$$

which ends this case.

We investigated all possible three cases hence the claim also holds after G_{i+1} was revealed. By induction, the claim holds for any $i + 1$. \square

Next, we state an auxiliary lemma that is later used to prove Lemma 12.

Lemma 11. *Let be N real numbers a_1, a_2, \dots, a_N . Let be N real numbers b_1, b_2, \dots, b_N each between 0 and 1, we call $\bar{b}_i = 1 - b_i$ for all $i \in \{1 \dots N\}$. It holds that*

$$\sum_{t \in \{0,1\}^N} \left[\sum_{i=1}^N \bar{t}_i \cdot a_i \right] \cdot \left[\sum_{i=1}^N t_i \cdot a_i \right] \prod_{k=1}^N b_k^{t_k} \cdot \bar{b}_k^{\bar{t}_k} \leq \sum_{i=1}^N b_i a_i (A - a_i),$$

where $A = \sum_{i=1}^N a_i$.

Proof.

$$\begin{aligned}
\sum_{t \in \{0,1\}^N} \left[\sum_{i=1}^N t_i \cdot a_i \right] \cdot \left[\sum_{j=1}^N \bar{t}_j \cdot a_j \right] \prod_{k=1}^N b_k^{t_k} \cdot \bar{b}_k^{\bar{t}_k} &= \sum_{i=1}^N a_i \sum_{\substack{t \in \{0,1\}^N \\ t_i=1}} \left[\sum_{j=1}^N \bar{t}_j \cdot a_j \right] \prod_{k=1}^N b_k^{t_k} \cdot \bar{b}_k^{\bar{t}_k} \\
&= \sum_{i=1}^N a_i \cdot b_i \sum_{\substack{t \in \{0,1\}^N \\ t_i=1}} \left[\sum_{\substack{j=1 \\ k \neq i}}^N \bar{t}_j \cdot a_j \right] \prod_{\substack{k=1 \\ k \neq i}}^N b_k^{t_k} \cdot \bar{b}_k^{\bar{t}_k} \\
&= \sum_{i=1}^N a_i \cdot b_i \sum_{\substack{t \in \{0,1\}^N \\ t_i=1}} \left(\left[\sum_{\substack{j=1 \\ j \neq i}}^N \bar{t}_j \cdot a_j \right] \prod_{\substack{k=1 \\ k \neq i}}^N b_k^{t_k} \cdot \bar{b}_k^{\bar{t}_k} \right) \\
&= \sum_{i=1}^N a_i \cdot b_i \left[\sum_{\substack{j=1 \\ j \neq i}}^N a_j \cdot b_j \right] \\
&\leq \sum_{i=1}^N a_i \cdot b_i (A - a_i).
\end{aligned}$$

We used Lemma 9 from line 4 to line 5, and used that $b_i \leq 1$ for the last inequality. \square

Lemma 12. *Let $i \in \{0 \dots k-1\}$. Let R_{X_i, Z_i} be the cost of the rearranging part of the update that follows the reveal of G_{i+1} . The following holds:*

$$\begin{aligned}
\frac{1}{2} \mathbb{E} [R_{X_i, Z_i}] &\leq |(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{X_i, X_i}| \cdot \frac{|X_i||Z_i| + \binom{|Z_i|}{2}}{\binom{|X_i|+|Z_i|}{2}} + |(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{X_i, Z_i}| \cdot \frac{\binom{|X_i|}{2} + \binom{|Z_i|}{2}}{\binom{|X_i|+|Z_i|}{2}} \\
&\quad + |(L_{\pi_0} \setminus L_{\pi_k^{\text{OPT}}}) \cap L_{Z_i, Z_i}| \cdot \frac{|X_i||Z_i| + \binom{|X_i|}{2}}{\binom{|X_i|+|Z_i|}{2}}.
\end{aligned}$$

Proof. In its final permutation π_k^{OPT} , the optimal offline algorithm OPT must have chosen one configuration for each component. In particular, OPT chose one configuration for the orientation of X_i , one for the relative position of X_i and Z_i , and one for the orientation of Z_i . Without loss of generality, we assume that OPT configured X_i and Z_i as follows: \overleftarrow{X}_i , $Z_i \text{---} X_i$ and \overleftarrow{Z}_i . In that case, we use Lemma 11 with the following values: $N = 3$, $a_1 = \binom{|X_i|}{2}$, $a_2 = |X_i| \cdot |Z_i|$, $a_3 = \binom{|Z_i|}{2}$, $b_1 = P[\overrightarrow{X}_i]$, $b_2 = P[X_i \text{---} Z_i]$ and $b_3 = P[\overrightarrow{Z}_i]$. The obtained expression corresponds to the expected value of R_{X_i, Z_i} if we divide it by $\binom{|X_i|+|Z_i|}{2} / 2$. We use the inequality of Lemma 11 and then Lemma 10 to replace the probabilities by their expressions. Finally, we perform the same reasoning as in Lemma 4 to link our upper bound with the final permutation of OPT and obtain the claim. \square

Next, we state an auxiliary lemma that is useful in the proof the next theorem.

Lemma 13. Let s_1, s_2, \dots, s_N be a series of strictly positive natural numbers, let $S = \sum_{i=1}^N s_i$ be their sum. It holds:

$$\sum_{i=1}^N \frac{s_i^2}{\binom{\sum_{j=1}^i s_j}{2}} \leq 2H_S \quad \text{and} \quad \sum_{i=2}^N \frac{s_{i-1} \cdot s_i}{\binom{\sum_{j=2}^i s_j}{2}} \leq 2H_S.$$

where $H_S = 1 + \frac{1}{2} + \dots + \frac{1}{S}$ is the harmonic sum until the S -th term.

Proof. Proving the first inequality is direct since all the terms of its left member are smaller than those of lemma 5. We now prove the second inequality.

$$\begin{aligned} \sum_{i=2}^N \frac{s_{i-1} \cdot s_i}{\binom{\sum_{j=2}^i s_j}{2}} &\leq 2 \sum_{i=2}^N \frac{s_{i-1}}{\sum_{j=2}^{i-1} s_j} \cdot \frac{s_i}{\sum_{j=2}^i s_j} \\ &\leq 2 \sqrt{\sum_{i=2}^N \left(\frac{s_{i-1}}{\sum_{j=2}^{i-1} s_j} \right)^2} \cdot \sqrt{\sum_{i=2}^N \left(\frac{s_i}{\sum_{j=2}^i s_j} \right)^2} \\ &\leq 2H_S \end{aligned}$$

We used the Cauchy-Schwarz inequality in the second line. \square

Theorem 14. Calling M and R respectively the moving and rearranging costs of RAND , it holds that

$$\mathbb{E}[M + R] \leq 8H_n \cdot |L_{\pi_0} \setminus L_{\pi_k^{\text{Opt}}}|$$

where H_n is the harmonic sum.

Proof. We first derive an upper bound of the expected value of R .

$$\begin{aligned} \frac{1}{2} \mathbb{E}[R] &= \sum_{i=0}^{k-1} \frac{1}{2} \mathbb{E}[R_{X_i, Z_i}] \\ &\leq \sum_{i=0}^{k-1} |(L_{\pi_0} \setminus L_{\pi_k^{\text{Opt}}}) \cap L_{X_i, X_i}| \cdot \frac{|X_i||Z_i| + \binom{|Z_i|}{2}}{\binom{|X_i|+|Z_i|}{2}} + |(L_{\pi_0} \setminus L_{\pi_k^{\text{Opt}}}) \cap L_{X_i, Z_i}| \cdot \frac{\binom{|X_i|}{2} + \binom{|Z_i|}{2}}{\binom{|X_i|+|Z_i|}{2}} \\ &\quad + |(L_{\pi_0} \setminus L_{\pi_k^{\text{Opt}}}) \cap L_{Z_i, Z_i}| \cdot \frac{|X_i||Z_i| + \binom{|X_i|}{2}}{\binom{|X_i|+|Z_i|}{2}} \\ &= \sum_{p \in L_{\pi_0} \setminus L_{\pi_k^{\text{Opt}}}} \sum_{i=0}^{k-1} \frac{|X_i||Z_i| + \binom{|Z_i|}{2}}{\binom{|X_i|+|Z_i|}{2}} \cdot \mathbb{1}(p \in L_{X_i, X_i}) + \frac{\binom{|X_i|}{2} + \binom{|Z_i|}{2}}{\binom{|X_i|+|Z_i|}{2}} \cdot \mathbb{1}(p \in L_{X_i, Z_i}) \\ &\quad + \frac{|X_i||Z_i| + \binom{|X_i|}{2}}{\binom{|X_i|+|Z_i|}{2}} \cdot \mathbb{1}(p \in L_{Z_i, Z_i}) \end{aligned} \tag{2}$$

Combining the upper bound (2) above with the upper bound (1) in the proof of Theorem 6, we now have upper bounds for the expected costs of both moving and rearranging parts of the updates, and hence for the total expected cost of RAND .

Let $p \in L_{\pi_0} \setminus L_{\pi_k}^{\text{Opr}}$ and let $y \in p$. In the upper bounds (1) and (2), we sum and upper-bound all the positive coefficients associated with node y and pair p . In case the nodes of p never belong to the same component then the upper bound of Theorem 6 then the upper bound (1) will be sufficient for that pair. We now assume that the nodes of p eventually belong to the same component, let i_p be the request index when this happens.

Using the upper bound (1) and the same reasoning as in Theorem 6, we find that the sum of the positive coefficients associated to y and p is no greater than $4H_{i_p}$ from revealed subgraphs G_1 to G_{i_p} . We now upper-bound the cost associated to y and p after i_p using (2). Let $Y_{i_p} \dots Y_N$ be the components that successively merge with y 's component after i_p : Y_{i_p} is the component that contains the other node in p . We can associate the rearranging cost paid during the i_p 's update with the coefficient $\binom{|Y_{i_p}|}{2} + \binom{|Y_{i_p-1}|}{2} / \binom{|Y_{i_p}| + |Y_{i_p-1}|}{2}$ in the sum. All the rearranging costs paid after the i_p 's update are associated with the coefficients $(|Y_{i+1}| \cdot |Y_i| + \binom{|Y_{i+1}|}{2}) / \binom{|Y_{i+1}| + |Y_i|}{2}$. Hence, by applying Lemma 13 for those coefficients, we obtain that the expected cost incurred between request indices i_p and k is no greater than $4(H_k - H_{i_p})$ for y in the pair p . Applying this reasoning for the two nodes in each pair $p \in L_{\pi_0} \setminus L_{\pi_k}^{\text{Opr}}$, we finally have that

$$\mathbb{E}[M + R] \leq \sum_{p \in L_{\pi_0} \setminus L_{\pi_k}^{\text{Opr}}} 4H_{i_p} + 8(H_k - H_{i_p}) \leq 8H_n \cdot |L_{\pi_0} \setminus L_{\pi_k}^{\text{Opr}}|,$$

which ends the proof. \square

Finally, using the above Theorem 14 with the Observation 7 directly proves Theorem 8.

5 Lower Bounds

In this section, we show tightness of our analysis. First, we show a lower bound for any randomized online algorithm, concluding that our randomized algorithm is asymptotically optimal among all online randomized algorithms. Second, we show a weaker lower bound for the competitiveness of the deterministic algorithms belonging to the family of algorithms considered in Section 2, concluding that our analysis of this family is tight; however the question whether this family is optimally competitive remains open.

5.1 A Lower Bound of $\Omega(\log n)$ for any Randomized Algorithm

The lower bound is an adaptation of the lower bound given by Olver et al. for the *itinerant list update problem* [25]. This theorem gives an asymptotically tight lower bound to complement Theorems 2 and 8.

Theorem 15. *If a randomized online algorithm for the online minimum linear arrangement is c -competitive, then $c \geq \frac{1}{16} \cdot \log n$.*

Proof. We apply Yao's principle to derive the lower bound. We construct a distribution of request sequences in the following way. Let n be a power of 2 so that $n = 2^q$, and $q = \log n$. First, we choose a random permutation P of the n nodes. Then, we construct a balanced binary tree whose leaves are the permutation P and the internal nodes are added to form the tree. Then, we request pairs of nodes as follows. We traverse the tree level by level bottom-up

starting from the penultimate level. For each internal node z of the current level of binary tree, we issue a request between the two leaves, u and v , chosen by following a line in the tree from the internal node until we encounter a leaf. Concretely, to obtain the leaf u , we first descend from z towards its left child once, and then we continue to descend to the right child until a leaf is reached. To obtain the leaf v , we first descend from z to its right child once, and then we continue to descend to the left child until a leaf is reached.

Any optimal offline algorithm pays at most n^2 for the constructed sequence of requests. Consider an offline algorithm OFF that orders the leaves according to P at the beginning and then does not move. OFF produces a feasible solution since all requests are between neighbors in P . Any optimal offline algorithm pays no more than OFF .

Fix any deterministic online algorithm ALG . We show that ALG pays at least $\Omega(n^2 \cdot \log n)$ to serve the requests. To this end, we will show that on the request issued at each level of the binary tree, ALG pays at least $\frac{1}{8}n^2$. Before the requests of the level i arrives, ALG has 2^i components, each of size 2^{q-i} . We claim that the expected cost of ALG on the j -th request is $(2^{i-1} + j) \cdot 2^{2(q-i)}$. First, assume that ALG moves components only after their nodes are requested. At the j -th request, we have exactly $2^i - 2j$ components not requested in this round. As requests are issued accordingly to a random permutation, for the j -th request the number of components not yet requested at this level between the requested nodes is at least $(2^i - 2j)/2$. The requested components must be collocated, hence for each of such components in between, swapping its position with the requested component costs $(2^{q-i})^2$. In total, the cost for the j -th request is at least

$$\sum_{j=1}^{2^{i-1}} (2^{i-1} - j) \cdot (2^{q-i})^2 = 2^{2q-3} = \frac{1}{8}n^2.$$

Second, if ALG moves some components before they are requested, each such swap could reduce the migration cost for the requests at the current level by at most 2, hence the cost of ALG for the requests at the current level is at least $\frac{1}{16}n^2$.

Summing over the $q = \log n$ levels results in the total cost at least $\frac{1}{16}n^2 \cdot \log n$, whereas any optimal offline algorithm pays at most n^2 , which concludes the proof. \square

5.2 A Lower Bound of $\Omega(n)$ for a Family of Deterministic Algorithms

Now, we give a lower bound of $\Omega(n)$ for competitiveness of the family of algorithms introduced in Section 2. Combined with Theorem 1, it implies that our analysis of these algorithms is tight. However, the question whether the competitiveness of *any* deterministic online algorithm is linear remains unresolved.

Theorem 16. *Any deterministic online algorithm that always moves to any feasible permutation with the lowest distance to π_0 is no better than $\Omega(n)$ -competitive.*

Proof. Fix any deterministic algorithm ALG that always moves to any feasible permutation with the lowest distance to π_0 . Consider any line topology with an odd number of nodes, and let x be a node in the middle of the line.

We construct the sequence of requests as follows. Let y_1 and y_2 be the nodes directly on the left and right of x . First, we request y_1 and y_2 , and refer to Y_2 as the component containing y_1 and y_2 . To serve this request, ALG must place Y_2 either on the left or on the right of x . Let y_3

be the neighbor of x such that x is between y_3 and Y_2 — this node exists since putting x at one end is not a closest permutation to π_0 . Then, we request y_3 and a node from Y_2 . We end up with component $Y_3 = \{y_1, y_2, y_3\}$. We continue to issue requests growing the component Y_i by issuing a request to the neighbor of x that neighbors x but is not contained in the component Y_i . We continue this process until we have one component of size $n - 1$ and x alone.

The cost of an optimal offline solution is at most n , since the nodes in Y_{n-1} have the same internal order as in π_0 , so an offline algorithm can serve all the requests by moving x to either the leftmost or the rightmost position on the line and then no further node movements are necessary.

We claim that ALG pays $\Omega(n^2)$ for this sequence of requests. Since ALG moves to a permutation closest to π_0 , x alternates between the left and the right side of the growing component Y_i . Below we prove that this behavior must occur. Let L be the set of nodes that were on the left side of x in the initial configuration, R for the nodes on the right. Let Y_i be the growing component after i requests. We first show that: In Y_i , there are strictly more nodes from L than R , and hence x is on the right of Y_i . Likewise, In Y_i , there are strictly more nodes from R than L , and hence x is on the left of Y_i . This is true since ALG moves to a permutation with the smallest distance to π_0 . Hence, any node from L will stay on the left of x as long as it is not requested, and the equivalent statement holds for R . Since the ordering within Y_i is forced, the only ordering that ALG could change is the order between x and S_i , hence the result. Now, notice that: If x is on the left of Y_i then the next request will add a node from L into Y_{i+1} . If x is on the right of Y_i then the next request will add a node from R into Y_{i+1} . The nodes in Y_{i+1} belong to either L or R , and at all times we either have a majority of nodes from L or from R . The majority of nodes within Y will change every second request. Each time the majority changes, x and Y change their relative positions, which incurs a final cost of $\Omega(n^2)$ for ALG.

The cost of the optimal solution is at most $O(n)$, and the cost of ALG is $\Omega(n^2)$, hence the competitive ratio of ALG is $\Omega(n)$. \square

As a result of the theorems in this section, the analysis of our deterministic and the randomized algorithms is tight. Furthermore, our randomized algorithm attains asymptotically optimal competitive ratio among all randomized online algorithms.

6 Conclusions

This paper considered a fundamental online variant of the minimum linear arrangement problem. The problem is motivated by the goal of understanding how to efficiently adjust virtual network embeddings. Our main contribution is a tight randomized online algorithms for this problem in a restricted case where the to-be-embedded graph is either a collection of lines or a collection of cliques. An interesting open research question is whether online algorithms can attain the logarithmic competitive ratio also for general graphs.

References

- [1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009.

- [2] C. Ambühl, M. Mastrolilli, and O. Svensson. Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. *SIAM Journal on Computing*, 40(2):567–596, 2011.
- [3] C. Avin, M. Bienkowski, A. Loukas, M. Pacut, and S. Schmid. Dynamic balanced graph partitioning. *SIAM J. Discret. Math.*, 34(3):1791–1812, 2020.
- [4] C. Avin, I. van Duijn, M. Pacut, and S. Schmid. Self-adjusting grid networks. *Information and Computation*, 292, 2023.
- [5] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. A new and improved algorithm for online bin packing. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 5:1–5:14, 2018.
- [6] J. Bhasker and S. Sahni. Optimal linear arrangement of circuit components. *Journal of VLSI and computer systems*, 2, 03 2002.
- [7] M. Bienkowski, M. Böhm, M. Koutecký, T. Rothvoß, J. Sgall, and P. Veselý. Improved analysis of online balanced clustering. *Approximation and Online Algorithms - International Workshop (WAOA)*, 2021.
- [8] M. Bienkowski and G. Even. An improved approximation algorithm for dynamic minimum linear arrangement. In *41st International Symposium on Theoretical Aspects of Computer Science, (STACS)*, volume 289 of *LIPICs*, pages 15:1–15:19, 2024.
- [9] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. 1998.
- [10] N. Buchbinder and J. Naor. The design of competitive online algorithms via a primal-dual approach. *Found. Trends Theor. Comput. Sci.*, 3(2-3):93–263, 2009.
- [11] M. Chowdhury, M. R. Rahman, and R. Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. Netw.*, 20(1):206–219, 2012.
- [12] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, 2002.
- [13] M. Eikel, C. Scheideler, and A. Setzer. Minimum linear arrangement of series-parallel graphs. In *Approximation and Online Algorithms - 12th International Workshop (WAOA)*, volume 8952 of *Lecture Notes in Computer Science*, pages 168–180. Springer, 2014.
- [14] L. Epstein and M. Levy. Online interval coloring and variants. In *Automata, Languages and Programming*, pages 602–613, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [15] S. Even. Np-completeness of several arrangement problems. *Technical Report; Department of computer Science*, 43, 1975.
- [16] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *IEEE Commun. Surv. Tutorials*, 15(4):1888–1906, 2013.

- [17] B. Gamlath, M. Kapralov, A. Maggiori, O. Svensson, and D. Wajc. Online matching with general arrivals. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, 2019.
- [18] M. M. Halldórsson, K. Iwama, S. Miyazaki, and S. Taketomi. Online independent sets. *Theoretical Computer Science*, 289(2):953–962, 2002. Computing and Combinatorics.
- [19] L. H. Harper. Optimal assignments of numbers to vertices. *Journal of The Society for Industrial and Applied Mathematics*, 12:131–135, 1964.
- [20] M. Henzinger, S. Neumann, H. Räcke, and S. Schmid. Tight bounds for online graph partitioning. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021.
- [21] M. Henzinger, S. Neumann, and S. Schmid. Efficient distributed workload (re-)embedding. In *2019 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*. ACM, 2019.
- [22] A. Mehta. Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science (FOCS)*, 8(4):265–368, 2013.
- [23] M. Miller, R. S. Rajan, N. Parthiban, and I. Rajasingh. Minimum Linear Arrangement of Incomplete Hypercubes. *The Computer Journal*, 58(2):331–337, 2013.
- [24] J. C. Mogul and L. Popa. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Computer Communication Review*, 42(5):44–48, 2012.
- [25] N. Olver, K. Pruhs, K. Schewior, R. Sitters, and L. Stougie. The itinerant list update problem. In *Approximation and Online Algorithms*, 2018.
- [26] M. Pacut, M. Parham, and S. Schmid. Optimal online balanced graph partitioning. In *40th IEEE Conference on Computer Communications, INFOCOM 2021, Vancouver, BC, Canada, May 10-13, 2021*, pages 1–9, 2020.
- [27] J. Petit. Experiments on the minimum linear arrangement problem. *ACM J. Exp. Algorithmics*, 8, 2004.
- [28] S. J. Phillips and J. R. Westbrook. Online load balancing and network flow. In *ACM Symposium on Theory of Computing (STOC)*, pages 402–411, 1993.
- [29] S. Rao and A. W. Richa. New approximation techniques for some linear ordering problems. *SIAM J. Comput.*, 34(2):388–404, 2004.
- [30] M. Rost and S. Schmid. Charting the complexity landscape of virtual network embeddings. In *IFIP Networking Conference*, pages 55–63, 2018.
- [31] M. Rost and S. Schmid. Virtual network embedding approximations: Leveraging randomized rounding. *IEEE/ACM Trans. Netw.*, 27(5):2071–2084, 2019.
- [32] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the social network’s (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 123–137, 2015.

- [33] S. S. Seiden. On the online bin packing problem. *J. ACM*, 49(5):640–671, 2002.
- [34] Y. Shiloach. A minimum linear arrangement algorithm for undirected trees. *SIAM Journal on Computing*, 8(1):15–32, 1979.
- [35] S. Vishwanathan. Randomized online graph coloring. *Journal of Algorithms*, 13(4):657–669, 1992.