Approximating the Minimum Logarithmic Arrangement Problem

Julián Mestre
Meta Platforms Inc.
University of Sydney

Sergey Pupyrev
Meta Platforms Inc.

Abstract

We study a graph reordering problem motivated by compressing massive graphs such as social networks and inverted indexes. Given a graph, $G = (V, E)$, the *Minimum Logarithmic Arrangement* problem is to find a permutation, π, of the vertices that minimizes

$$\sum_{(u,v) \in E} 1 + \lfloor \lg |\pi(u) - \pi(v)| \rfloor .$$

This objective has been shown to be a good measure of how many bits are needed to encode the graph if the adjacency list of each vertex is encoded using relative positions of two consecutive neighbors under the π order in the list rather than using absolute indices or node identifiers, which requires at least $\lg n$ bits per edge.

We show the first non-trivial approximation factor for this problem by giving a polynomial time $\mathcal{O}(\log k)$-approximation algorithm for graphs with treewidth k.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases approximation algorithms, graph compression

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

We study theoretical aspects of a graph reordering problem that has applications to compressing social networks and inverted indexes. The formal model of the problem has been suggested by Chierichetti et al. [6], who proposed a simple heuristic for reordering web-scale graphs. Later Dhulipala et al. [11] extended the model and described a practical approach for graph reordering based on recursive bisection. The algorithm of [11] is widely used in practice producing the most "compression-friendly" vertex orders for a large variety of real-world datasets and is considered the state-of-the-art in the field [31].

A *linear layout* (an order or an arrangement) of a graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges is a bijection $\pi : V \to \{1, \ldots, n\}$. Most graph encoding schemes are based on performing a *delta-encoding* of the adjacency lists using a linear layout. The basic idea is to sort each adjacency list according to the layout π, store the index of the first neighbor in the list, followed by the the gaps between two consecutive neighbors using a variable length encoding. As such, it is desirable that the neighborhood of each vertex is laid out close together, since that translates into smaller gaps and higher compression rates. This motivates two problems that we define next.

The *minimum linear arrangement* (MLA) problem is to find a layout π so that

$$\mathrm{LA}_\pi(G) := \sum_{(u,v) \in E} |\pi(u) - \pi(v)|$$

is minimized. This is a classical NP-hard problem [23], even when restricted to certain graph classes. The problem is APX-hard under Unique Games Conjecture [10] but admits

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

an $\mathcal{O}(\sqrt{\log n}\log\log n)$ approximation [5, 19]. Notice that the objective measures the total length of the gaps across all adjacency lists.

A closely related problem is *minimum logarithmic arrangement* (MLogA) in which the goal is to minimize

$$\mathrm{LGA}_\pi(G) := \sum_{(u,v)\in E} (1 + \lfloor \lg |\pi(u) - \pi(v)| \rfloor),$$

where $\lg x$ denotes the logarithm base 2 of x. Note that for an integer x, $1 + \lfloor \lg x \rfloor$ is the number of bits needed to represent x. We write $\mathrm{LGA}(G) = \min_\pi \mathrm{LGA}_\pi(G)$, where the minimum is taken over all permutations of vertices V. Seen from this perspective, $\mathrm{LGA}(G)$ is a measure of the compressed size of G. It is worth noting that in practice, the size of an encoded integer and the total size of a graph depends on the utilized encoding scheme; we refer to [2] for a survey of modern graph compression techniques.[1]

1.1 Our Contributions

In this paper we study MLogA from a theoretical perspective. First, in Section 2, we investigate basic properties of the problem: analyze the performance of two natural heuristics, provide explicit optimal and near-optimal layouts for several graph classes, and describe a lower bound for the LGA cost of a graph.

Section 3 describes our main result, an $\mathcal{O}(\log k)$-approximation for graphs with treewidth k. It is worth noting that the optimal ordering has cost at least m and that *every* ordering has cost $\mathcal{O}(m \log n)$. Therefore outputting an arbitrary order is, technically speaking, a logarithmic approximation for MLogA. The challenge is to design an approximation algorithm with approximation factor $o(\log n)$. Our result is the first such approximation for the natural and broad class of graphs with low treewidth. The algorithm works by recursively splitting the input graph using small balanced separators. It is worth noting that our algorithm can be implemented to run in polynomial time regardless of the value of k. While the algorithm is fairly straightforward, its analysis is highly non-trivial.

Regarding the applicability of our approach, we point to the recent work of Maniu *et al.* [32] who experimentally estimated the treewidth of graphs arising from a variety of domains. They found that real-world instances usually have treewidth values that are very small compared the number of vertices in the graph. Therefore, we can reasonably expect our $\mathcal{O}(\log k)$ approximation to yield much better results in real-world instances over the trivial $\mathcal{O}(\log n)$ approximation.

We conclude the paper in Section 4 with some interesting open problems.

1.2 Related Work

Not many results on MLogA are known. Chierichetti et al. [6] show that the problem is `NP`-hard on multi-graphs and present lower bounds on expander-like graphs. More specifically, they show that if a graph G has constant conductance then the cost of MLogA is $\Omega(m \log n)$, and that if G has constant node or edge expansion then the cost of MLogA is $\Omega(n \log n)$.

[1] In addition to the MLogA problem, Chierichetti et al. [6] and Dhulipala et al. [11] define and study practical variants of the problem, which they call *minimum logarithmic gap arrangement* and *bipartite minimum logarithmic arrangement*. In this paper we focus on MLogA and mention the relationship between different variants in Appendix A.

Most closely related to our problem is the minimum linear arrangement problem, which has been studied under various names [12], such as optimal linear ordering, minimum-1-sum, or the edge sum problem. MLA was originally proposed in [26]. It was proven to be strongly NP-hard [22] and this was later shown to hold even for bipartite graphs [16] and interval graphs [9]. For general graphs, the fastest known exact algorithm is based on dynamic programming and runs in $\mathcal{O}(2^n \cdot m)$ time [30]. The best approximation factor known for general graphs is $\mathcal{O}(\sqrt{\log n} \log \log n)$ [5, 19]; however, better approximations are known for special graph classes such as interval graphs [9], planar graphs [35], and series-parallel graphs [15]. On the positive side, MLA is known to be solvable in polynomial time on trees [1, 8, 24]. Furthermore, for some restricted classes of graphs, optimal layouts are known explicitly [7, 26, 28].

There is vast literature on the problem of computing an ordering of a graph vertex set to minimize or maximize a given objective function. Here we only mention a few notable examples. The *minimum bandwidth problem* [13, 17, 20, 25, 37] is to find an ordering minimizing the maximum distance between any two vertices connected with an edge; that is, $\min_\pi \max |\pi(u) - \pi(v)|$. Finding a tree (path) decomposition with minimum treewidth (pathwidth) can be cast as the problem of finding a elimination order of the vertices [29]. Finally, we mention the *traveling salesman problem* [27] and its many variants [3, 33, 34], which have inspired ground breaking algorithmic research for over five decades.

2 Preliminaries

In this section we first discuss natural heuristics for MLogA and their (worst-case) approximation factors. Then we derive optimal arrangements for several graph classes.

2.1 Heuristics

2.1.1 Greedy

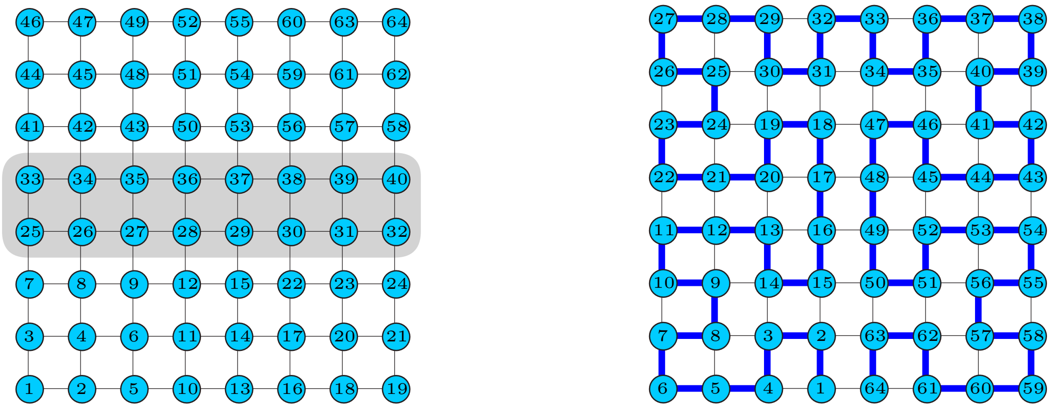
Arguably the easiest approach for MLogA is a greedy one. Start with a vertex, and iteratively add the next vertex that yields the lowest increase of the objective. There are several greedy criteria that we could use.

The simplest version of this approach that does not constrain in any way how we pick our vertices does not yield anything useful even in very simple instances: If the input is a path, the algorithm might pick every other vertex along the path and then the remaining vertices for a total cost of $\Omega(n \log n)$, whereas an optimal solution has cost $n - 1$.

One can refine the greedy criterion by asking that a newly added vertex is connected to one of the vertices already processed, and subject to this that the increase of the objective is minimized. Unfortunately, this also fails: If the input is a $2 \times n$ grid, the algorithm might pick the upper path of the grid followed by the lower path (in opposite direction) for a total cost of $\Omega(n \log n)$ whereas an optimal solution, which interleaves nodes from the top and bottom paths, has cost $\mathcal{O}(n)$.

2.1.2 MLA

It is tempting to apply an algorithm designed for MLA to solve MLogA. Recall that MLA admits $o(\log n)$-approximations. However such an approach may result in an $\Omega(\log n)$ gap between the output and the optimal solution. Consider the *square grid* graph; that is, the graph whose vertices correspond to the points in the plane with integer coordinates in the range $1, \ldots, h$ and two vertices connected by an edge whenever the corresponding points



(a) **(b)**

Figure 1 Layouts of the 8×8 grid graph optimizing for (a) MLA and (b) MLogA. The layout for MLA is constructed by an algorithm of Fishburn et al. [21] and contains $\Omega(h)$ consecutively numbered rows (shaded). The layout for MLogA is constructed following a space-filling curve as described in Lemma 7.

are at distance 1. The $h \times h$ grid is denoted by $G_{h,h}$ and contains $n = h^2$ vertices and $m = 2h(h-1)$ edges.

Fishburn et al. [21] describe the optimal arrangement, π, of the square grid; it contains t consecutively numbered rows with $t/h \to 1-\sqrt{2}$ as $h \to \infty$; see Figure 1a. The corresponding vertical edges between the rows in the grid have length h and there are $t \times h$ such edges. Summing up the contribution of the edges for MLogA, we get $\mathrm{LGA}_\pi \geq t \times h \times \lg h = \Omega(h^2 \times \log h) = \Omega(m \times \log n)$. However, as we show in Lemma 7, there is an $\mathcal{O}(m)$ solution for MLogA; see Figure 1b.

2.2 Lower bounds

Before proving a lower bound for the objective of MLogA, we show a simple fact about sums of logarithmic values. Due to lack of space, some of the proofs are deferred to Appendix B.

▶ **Lemma 1.** *For any integer $\ell \geq 1$ we have*

$$(\ell - 1) \cdot \lg(\ell + 1) < \sum_{i=1}^{\ell} (1 + \lfloor \lg i \rfloor) < (\ell + 1) \cdot \lg(\ell + 1).$$

It is clear that $\mathrm{LGA}(G) \geq m$ for every graph G, since the contribution of each edge to the objective is at least 1. The next lemma improves upon this trivial bound for dense graphs.

▶ **Lemma 2.** *Let $G = (V, E)$ be a graph with n vertices and m edges, then*

$$\mathrm{LGA}(G) \geq (m - n) \cdot \lg \frac{m}{n}$$

2.3 Specific Graph Classes

▶ **Lemma 3.** *Let K_n denote the complete graph with n vertices. Then*

$$\mathrm{LGA}(K_n) \leq \frac{n^2 \lg n}{2}.$$

Proof. The bound follows the observation that all layouts of a complete graph are equivalent, and applying Lemma 1 to each node and accounting for double counting:

$$\mathrm{LGA}(K_n) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n-1}(1+\lfloor \lg j \rfloor) \leq \frac{n^2 \lg n}{2}.$$

◀

▶ **Lemma 4.** *Let P_n and C_n denote the path and cycle with n vertices, respectively. Then*

$$\mathrm{LGA}(P_n) = n-1 = m \qquad \text{and} \qquad \mathrm{LGA}(C_n) = n + \lfloor \lg(n-1) \rfloor = m + \lfloor \lg(n-1) \rfloor.$$

Proof. The bound for the path is trivial. For the cycle, denote the lengths of the edges of C_n by $e_1, \dots, e_n$. Observe that for every ordering of C_n, there exist two edge-dis int paths connecting the first and the last vertices in the order. Hence, $e_1 + e_2 + \cdots + e_n \geq 2n-2$ and $e_i \geq 1$. Using an exchange argument, it is straightforward to show that given those constraints $\sum_{i=1}^{n}(1+\lfloor \lg e_i \rfloor)$ is minimized when $e_1 = \cdots = e_{n-1} = 1$ and $e_n = n-1$, which yields the claim. ◀

▶ **Lemma 5.** *Let $K_{1,\ell}$ denote a star with ℓ leaves. Then*

$$(\ell-2)\cdot(1+\lg\frac{\ell}{2}) \leq \mathrm{LGA}(K_{1,\ell}) \leq (\ell+2)\cdot \lg\frac{\ell+1}{2}.$$

Proof. The equality follows from the observation that the optimal layout is achieved when the central vertex of the star is placed in the middle of the order. The inequalities are the result of applying Lemma 1 to this sum. ◀

▶ **Lemma 6.** *Let T_n denote the k-level complete binary tree with $n = 2^k - 1$ vertices. Then*

$$\mathrm{LGA}(T_n) \leq \left\lceil \tfrac{5}{3}(2^k-1) \right\rceil - k - 1 \leq \tfrac{5}{3}n.$$

We conjecture that the bound given by Lemma 6 is optimal for complete binary trees; it has been verified computationally for trees with up to 15 vertices.

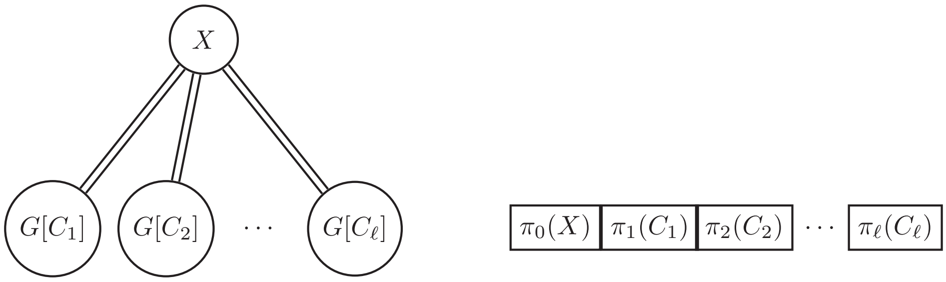
Next we explore MLogA on the $h \times h$ grid graph, $G_{h,h}$, and suggest using a space-filling curve to layout the vertices. A space filling curve is a continuous mapping from the unit interval $[0,1]$ to the unit square $[0,1]^2$. The idea is to overlay the grid over the unit square and then use the curve order to sort the vertices of the grid. We show that the layout obtained from the well-known Hilbert curve [38] yields a constant factor approximation for MLogA.

▶ **Lemma 7.** *Let $G_{h,h}$ denote the $h \times h$ grid graph with h being a power of two. Then*

$$\mathrm{LGA}(G_{h,h}) \leq 4h^2 = \mathcal{O}(m).$$

3 Balanced Separator

In this section we explore the performance of a divide-and-conquer algorithm based on balanced separators. Recall that a set of vertices $X \subseteq V$ is a *balanced vertex separator* for $G = (V,E)$ if every connected component of $G[V \setminus X]$ has at most $\left\lceil \frac{|V \setminus X|}{2} \right\rceil$ vertices. The *separation number* of G is the minimum integer k such that every subgraph of G has a balanced separator of order at most k. It is known that the separation number of a graph is linearly related to its treewidth [14, 36].



(a) A balanced separator X and the connected components of $G[V \setminus X]$.

(b) Partial layouts of each component are sequenced in an arbitrary order.

Figure 2 Algorithm BALANCED finds a small balanced separator X, recursively computes a partial layout π_i for each connected component C_i of $G[V \setminus X]$, and builds a full layout of G by concatenating these partial layouts and an arbitrary sequencing of X.

Our algorithm, which we call BALANCED, recursively finds a small balanced separator X, arbitrarily sequences X to get a partial layout $\pi_0(X)$, identifies the connected components $C_1, C_2, \ldots, C_\ell$ of $G[V \setminus X]$, recursively finds a layout $\pi_i(C_i)$ for each subgraph $G[C_i]$, and then concatenates all these layouts in arbitrary order, say $\langle \pi_0(X), \pi_1(C_1), \pi_2(C_2), \ldots, \pi_\ell(C_\ell)\rangle$.

For each problem $G' = (V', E')$ we find along the way, we assign a level value to the problem based on its size; more precisely, we say that the subproblem is in level i if $\frac{n}{2^i} \le |V'| < \frac{n}{2^{i-1}}$. Note that because of the balanced nature of the separators, a problem can only generate subproblems in lower levels. Thus, it follows that the collection of subproblems at level i forms a partition of a vertex subset of the input instance. Furthermore, since each subproblem at level i has cardinality at least $\frac{n}{2^i}$, it follows that we can have at most 2^i subproblems at level i. We extend the level assignment of subproblems to vertices as follows. If u belongs to the separator chosen for a subproblem at level i, then we say that u belongs to level i. Notice that once a vertex is chosen into a separator, it never again shows up in subproblems.

Now consider a separator X of a subproblem $G' = (V', E')$. We assign every edge in E' incident on X towards an endpoint in X (edges in $E'[X]$ can pick an endpoint arbitrarily). For each $u \in X$, let μ_u be the number of edges assigned to u in this way; note that $\mu_u > 0$ since every u must be connected to $V' \setminus X$, otherwise $X - u$ is also a balanced separator. Lastly, let L_i denote the set of nodes in level i and $\ell - 1$ be the deepest non-empty level. Since every edge in the input instance is assigned in this way, it follows that $\sum_u \mu_u = |E|$ and that $1 \le \mu_u \le n$.

On one hand, the cost of the layout is upper bounded by

$$\mathrm{UB}(\mu) = \sum_{i=0}^{\ell-1} \sum_{u \in L_i} \mu_u \cdot \left(1 + \lg \frac{n}{2^i}\right)$$

On the other hand, every node u with μ_u edges assigned to it needs at least as many bits to encode the edges as a star with μ_u leaves does. Using Lemma 5 we can infer that we need at least $(\mu_u - 2) \cdot \lg\left(1 + \frac{\mu_u}{2}\right)$ bits. Together with the fact that we always need at least μ_u bits, we get that $\frac{\mu_u}{4} \cdot (1 + \lg \mu_u)$ bits are always needed. Therefore, by ignoring the constant factor, we can use the following as the lower bound:

$$\mathrm{LB}(\mu) = \sum_{i=0}^{\ell-1} \sum_{u \in L_i} \mu_u \cdot (1 + \lg \mu_u)\,.$$

Therefore, the approximation ratio of the algorithm is bounded up to constant factors by $\frac{\mathrm{UB}(\mu)}{\mathrm{LB}(\mu)}$. The rest of this section is devoted to showing that if the graph has small balanced separators, this ratio is small.

Consider the auxiliary problem of finding an assignment μ and levels L_i that maximizes $\frac{\mathrm{UB}(\mu)}{\mathrm{LB}(\mu)}$ subject to the following constraints:

- $\sum_u \mu_u \geq n$, and $1 \leq \mu_u \leq n$ for all u,
- $|L_i| \leq k2^i$, where k is an absolute upper bound on the cardinality of the balanced separators we find along the way.

Strictly speaking the first constraint should be $\sum_u \mu_u = m$, but as we shall soon see, the worst bound of $\frac{\mathrm{UB}(\mu)}{\mathrm{LB}(\mu)}$ occurs when $m = n$. The second constraint follows from the fact that there are at most 2^i sub-problems at level i and that each of these has a separator of size at most k.

▶ **Lemma 8.** *For any assignment μ and levels L_i subject to the above constraints, the ratio $\frac{\mathrm{UB}(\mu)}{\mathrm{LB}(\mu)}$ is upper bounded by $\mathcal{O}(\log k)$.*

Proof. Define $\rho(\mu) = \frac{\mathrm{UB}(\mu)}{\mathrm{LB}(\mu)}$. First we identify further constraints that we can assume without loss of generality:

- $\sum_u \mu_u = n$. Otherwise, we can multiply μ by $\gamma = \sum_u \mu/n$, which cause $\mathrm{UB}(\mu)$ to scale down by a factor of γ, while $LB(\mu)$ decreases by a factor strictly greater than γ (due to its super-linear terms).
- $\forall u, v \in L_i$, we have $\mu_u = \mu_v$. Otherwise, average their values, which does not change $\mathrm{UB}(\mu)$ but decreases $\mathrm{LB}(\mu)$.
- $\forall u \in L_i, v \in L_j$, if $i < j$ then $\mu_u \geq \mu_v$. Otherwise, we can swap their values, increasing $\mathrm{UB}(\mu)$ without changing $\mathrm{LB}(\mu)$.
- $|L_i| = 2^i$ for all $i < \ell - 1$. Otherwise, if a level is not full we can promote a node for a lower level, which increases $\mathrm{UB}(\mu)$ but does not change $\mathrm{LB}(\mu)$.

In every case, the change increases $\rho(\mu)$, so we can assume all these properties without loss of generality.

We also assume that $|L_{\ell-1}| = 2^{\ell-1}$. Otherwise, if the level is not full we get rid of it altogether and scale up other values to add up to n. This can decrease the value of the solution a single time by a constant multiplicative amount; that is, at most 2.

Furthermore, we can assume that if we have two nodes $u \in L_i$ and $v \in L_{i+1}$ in consecutive layers and we increase/decrease μ_u and decrease/increase μ_v the change should not improve the ratio $\frac{\mathrm{UB}(\mu)}{\mathrm{LB}(\mu)}$, which we denote for brevity with ρ from now on. Out of this requirement we get the following property (see Appendix B for the proof).

▶ **Lemma 9.** *The worst ratio $\rho(\mu) = \frac{\mathrm{UB}(\mu)}{\mathrm{LB}(\mu)}$ is attained when for any two nodes $u \in L_i$ and $v \in L_{i+1}$ in consecutive layers we have*

$$\frac{\mu_u}{\mu_v} = 2^{\frac{1}{\rho(\mu)}}.$$

Thus, for the purposes of finding a bad assignment for our analysis, we can focus our attention on those obeying the above properties. To that end, we define μ_i to be the value of those nodes in level i. Therefore, without loss of generality, we focus on the following quantities

$$\widehat{\mathrm{UB}}(\mu) = n + \sum_{i=0}^{\ell-1} k2^i \mu_i \lg \frac{n}{2^i}$$

and

$$\widehat{\mathrm{LB}}(\mu) = n + \sum_{i=0}^{\ell-1} k 2^i \mu_i \lg \mu_i$$

Furthermore, using Lemma 9 we infer that

$$\mu_i = \frac{\mu_0}{2^{\frac{i}{\rho}}} \tag{1}$$

Let $\alpha = 2^{1-\frac{1}{\rho}}$. Note that since $\rho > 1$, it follows that $1 < \alpha < 2$. Plugging (1) into the upper and lower bounds we get

$$\widehat{\mathrm{UB}}(\mu) = n + k\mu_0 \sum_{i=0}^{\ell-1} \alpha^i \lg \frac{n}{2^i}$$

and

$$\widehat{\mathrm{LB}}(\mu) = n + k\mu_0 \sum_{i=0}^{\ell-1} \alpha^i \lg \frac{\mu_0}{2^{\frac{i}{\rho}}}$$

Approximating the value of the upper bound using integrals to get:

$$\begin{aligned}
\widehat{\mathrm{UB}}(\mu) &\le n + k\mu_0 \int_1^{\ell} \alpha^x \lg \frac{n}{2^x} \mathrm{d}x \\
&= n + k\mu_0 \left[\frac{\alpha^x}{\ln \alpha} \lg \frac{n}{2^x} + \frac{\alpha^x}{\ln^2 \alpha} \right|_1^{\ell} \\
&\le n + k\mu_0 \frac{\alpha^\ell}{\ln \alpha} \left(\lg \frac{n}{2^\ell} + \frac{1}{\ln \alpha} \right)
\end{aligned}$$

Approximating the value of the lower bound yields:

$$\begin{aligned}
\widehat{\mathrm{LB}}(\mu) &\ge n + k\mu_0 \int_1^{\ell-1} \alpha^x \lg \frac{\mu_0}{2^{x/\rho}} \mathrm{d}x \\
&= n + k\mu_0 \left[\frac{\alpha^x}{\ln \alpha} \lg \frac{\mu_o}{2^{x/\rho}} + \frac{\alpha^x}{\rho \ln^2 \alpha} \right|_0^{\ell-1} \\
& n + k\mu_0 \left[\frac{\alpha^{\ell-1}}{\ln \alpha} \left(\lg \frac{\mu_o}{2^{(\ell-1)/\rho}} + \frac{1}{\rho \ln \alpha} \right) - \left(\frac{\lg \mu_0}{\ln \alpha} + \frac{1}{\rho \ln^2 \alpha} \right) \right] \\
&\ge c \left[n + k\mu_0 \frac{\alpha^{\ell-1}}{\ln \alpha} \left(\lg \frac{\mu_o}{2^{(\ell-1)/\rho}} + \frac{1}{\rho \ln \alpha} \right) \right]
\end{aligned}$$

where the last inequality holds for a constant $c > 1/2$ assuming that $\rho > 2$ and $\ell > 1$. Both of these assumptions are safe to make for otherwise $\rho = \mathcal{O}(1)$.

Finally, note that $n = \sum_{i=0}^{\ell-1} k\mu_0 \alpha^i$, which yields $n \le k\mu_0 \frac{\alpha^\ell}{\alpha-1}$. Therefore,

$$\lg \frac{n}{2^\ell} \le \lg k + \lg \frac{\mu_0}{2^{\ell/\rho}} - \lg(\alpha - 1) \le \lg k + \lg \frac{\mu_0}{2^{(\ell-1)/\rho}} + 1,$$

where the last inequality holds for $\rho > 2$. Also, using the same assumption we note that $1/\lg\alpha = \frac{\rho}{\rho-1} \leq 2$, and so $\frac{1}{\ln\alpha} = \mathcal{O}(1)$.

Using the fact that $\mu_i \geq 1$ for all i, we get $\lg \frac{\mu_0}{2^{(\ell-1)/\rho}} \geq 0$. Therefore, the ratio $\frac{\widehat{\mathrm{UB}}(\mu)}{\widehat{\mathrm{LB}}(\mu)}$ is maximized when the previous inequality is tight, which yields that $\frac{\widehat{\mathrm{UB}}(\mu)}{\widehat{\mathrm{LB}}(\mu)} = \mathcal{O}(\log k)$. ◀

▶ **Theorem 10.** *For a graph with separation number at most k, algorithm* BALANCED *is an $\mathcal{O}(\log k)$-approximation for MLogA.*

Proof. The claim follows readily from Lemma 8. ◀

3.1 Implementation Details

In this section we discuss implementation details of BALANCED. While the guarantee in Theorem 10 is expressed in terms of the separation number of the input graph, we observe that finding a minimum balanced separator is an NP-hard problem [4]. However, we can get the same asymptotic guarantee by applying an approximation algorithm instead.

▶ **Lemma 11.** *Algorithm* BALANCED *can be implemented to run in polynomial time while maintaining an approximation factor of $\mathcal{O}(\log k)$, where k is the separation number of the input graph.*

Proof. Feige [18] provides a polynomial time algorithm finding a balanced separator of size $\mathcal{O}(k\sqrt{k})$ provided the input graph has a balanced separator of size k. Using the approximation algorithm for finding our balanced separators and Lemma 8, we get an approximation guarantee of $\mathcal{O}(\log(k\sqrt{k})) = \mathcal{O}(\log k)$.

Each node of the divide-and-conquer recursion tree performs a polynomial amount of work, therefore the overall running time is polynomial. ◀

We close this section by noting that once a balanced separator X of G is found, it is not important how the recursively-computed layouts of each component of $G[V \setminus X]$ and X itself are sequenced—this sequencing order does not affect the analysis. An optimized implementation, would benefit from engineering a good heuristic for ordering the components: Ideally, want to place components C close to the X that have large $|E[C, X]|$ and small $|C|$; however, these two metrics may be at odds with one another, so the heuristic would have to balance those two objectives.

3.2 Related Algorithms

Let us discuss the consequences of the analysis of Section 3 to other algorithms.

3.2.1 Bisection

The state-of-the-art approach for MLogA uses recursive graph bisection [11, 31]. Start with a given graph, G, and find a small almost balanced edge-cut, that is, a collection of edges whose removal yields two almost-equal-sized subgraphs. Then recursively layout each of the two subgraphs, and then concatenate the resulting orders.

It is natural to wonder if this is a good heuristic provided the balanced cuts found by the algorithm are relatively small. This is indeed the case, since the endpoints of the edges in an almost-balanced cut form an almost-balanced separator. Using a similar analysis technique to Theorem 10, one can show that if the bisection algorithm always finds almost-balanced cuts whose size is at most k then the solution found is $\mathcal{O}(\log k)$-approximate.

3.2.2 Centroid decomposition

Chung [8] proposed an optimal algorithm for MLA on trees that is based on the idea of removing the centroid of the tree, recursively finding a layout of each subtree and carefully concatenating these subtrees.

A similar algorithm (but without the need to be careful about how the subproblems are combined) is an $\mathcal{O}(1)$-approximation for MLogA on trees since the centroid is an almost-balanced separator.

4 Conclusions and Open Problems

In this paper we tackled a practical problem arising in graph compression. We studied approximation algorithms for MLogA, which was posed as an open question by Chierichetti et al. [6] and Dhulipala et al. [11]. Our main result, an approximation based on balanced separators, partially explains why the state-of-the-art heuristic (that uses a similar scheme) works well in practice.

There are several interesting open questions related to the problem. First, the complexity of MLogA on simple graphs and graphs of bounded treewidth is open. We emphasize that the related problem, MLA, can be solved on trees in polynomial time [1, 8, 24]. These algorithms rely on certain properties of optimally embedded trees for the linear objective, and it is unclear whether similar properties hold for the logarithmic objective. The complexity status of MLA on 2-trees (series-parallel graphs) is unsettled [15].

Another natural question is to design a constant-factor approximation algorithm for general graphs. We stress that Theorem 10 provides such an algorithm for graphs with a constant separation number. At the same time, graphs without small separators (e.g., with a constant conductance) have cost $\Omega(m \log n)$; thus, any order of the vertices yields a cost that is within a constant factor of the optimum. The challenge is to analyze the scenario between the two extremes.

Finally, we would like to see some progress on designing practical exact approaches for MLogA. To the best of our knowledge, there is no algorithm that works faster than the naive exhaustive search of $n!$ combinations. Can we solve the problem (exactly) in $\mathcal{O}(c^n)$ time for some constant $c > 0$? Is there an efficient integer programming formulation of the problem? We emphasize that the two questions are interesting even when the input graph is a tree.

Acknowledgments.

We thank Okke Schrijvers, Karthik Abinav Sankararaman and Riccardo Colini Baldeschi for fruitful discussions of the problem.

References

1. D Adolphson and T Ch Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.
2. Maciej Besta and Torsten Hoefler. Survey and taxonomy of lossless graph compression and space-efficient graph representations. *arXiv preprint arXiv:1806.01799*, 2018.
3. Avrim Blum, Prasad Chalasani, Don Coppersmith, William R. Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proc. of the 26th Annual ACM Symposium on Theory of Computing*, pages 163–171, 1994.
4. Thang Nguyen Bui and Curt Jones. Finding good approximate vertex and edge partitions is np-hard. *Inf. Process. Lett.*, 42(3):153–159, 1992.
5. Moses Charikar, Mohammad Taghi Hajiaghayi, Howard Karloff, and Satish Rao. L_2^2 spreading metrics for vertex ordering problems. *Algorithmica*, 56(4):577–604, 2010.
6. Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. On compressing social networks. In *Knowledge Discovery and Data Mining*, pages 219–228, 2009.
7. Fan-Rong King Chung. A conjectured minimum valuation tree. *SIAM Review*, 20(3):601–604, 1978.
8. Fan-Rong King Chung. On optimal linear arrangements of trees. *Computers & mathematics with applications*, 10(1):43–60, 1984.
9. Johanne Cohen, Fedor Fomin, Pinar Heggernes, Dieter Kratsch, and Gregory Kucherov. Optimal linear arrangement of interval graphs. In *International Symposium on Mathematical Foundations of Computer Science*, pages 267–279. Springer, 2006.
10. Nikhil R Devanur, Subhash A Khot, Rishi Saket, and Nisheeth K Vishnoi. Integrality gaps for sparsest cut and minimum linear arrangement problems. In *Symposium on Theory of Computing*, pages 537–546, 2006.
11. Laxman Dhulipala, Igor Kabiljo, Brian Karrer, Giuseppe Ottaviano, Sergey Pupyrev, and Alon Shalita. Compressing graphs and indexes with recursive graph bisection. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1535–1544, 2016.
12. Josep Díaz, Jordi Petit, and Maria Serna. A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, 34(3):313–356, 2002.
13. Markus Sortland Dregi and Daniel Lokshtanov. Parameterized complexity of bandwidth on trees. In *International Colloquium on Automata, Languages, and Programming*, pages 405–416, 2014.
14. Zdeněk Dvořák and Sergey Norin. Treewidth of graphs with balanced separations. *Journal of Combinatorial Theory, Series B*, 137:137–144, 2019.
15. Martina Eikel, Christian Scheideler, and Alexander Setzer. Minimum linear arrangement of series-parallel graphs. In *Proc. of the 12th International Workshop on Approximation and Online Algorithms*, pages 168–180. Springer, 2014.
16. Shimon Even and Yossi Shiloach. NP-completeness of several arrangement problems. Technical report 43, Dept. Computer Science, Technicon, Haifa, Isreal, 1975.
17. Uriel Feige. Approximating the bandwidth via volume respecting embeddings. *J. Comput. Syst. Sci.*, 60(3):510–539, 2000.
18. Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM J. Comput.*, 38(2):629–657, 2008.
19. Uriel Feige and James R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Inf. Process. Lett.*, 101(1):26–29, 2007.
20. Uriel Feige and Kunal Talwar. Approximating the bandwidth of caterpillars. *Algorithmica*, 55(1):190–204, 2009.
21. Peter Fishburn, Prasad Tetali, and Peter Winkler. Optimal linear arrangement of a rectangular grid. *Discrete Mathematics*, 213(1-3):123–139, 2000.

22 M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.

23 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

24 MK Goldberg and IA Klipker. Minimal placing of trees on a line. In *Physico-Technical Institute of Low Temperatures*. Academy of Sciences of Ukranian SSR, 1976.

25 Anupam Gupta. Improved bandwidth approximation for trees and chordal graphs. *J. Algorithms*, 40(1):24–36, 2001.

26 Lawrence Hueston Harper. Optimal assignments of numbers to vertices. *Journal of the Society for Industrial and Applied Mathematics*, 12(1):131–135, 1964.

27 Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees. *Oper. Res.*, 18(6):1138–1162, 1970.

28 Martin Juvan and Bojan Mohar. Optimal linear labelings and eigenvalues of graphs. *Discrete Applied Mathematics*, 36(2):153–168, 1992.

29 T. Kloks. *Treewidth: Computations and Approximations*. Springer-Verlag New York, 1994.

30 Yehuda Koren and David Harel. A multi-scale algorithm for the linear arrangement problem. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 296–309. Springer, 2002.

31 Joel Mackenzie, Antonio Mallia, Matthias Petri, J. Shane Culpepper, and Torsten Suel. Compressing inverted indexes with recursive graph bisection: A reproducibility study. In *Advances in Information Retrieval*, pages 339–352, Cham, 2019. Springer International Publishing.

32 Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data. In Pablo Barceló and Marco Calautti, editors, *Proc. of the 22nd International Conference on Database Theory*, volume 127, pages 12:1–12:18, 2019.

33 Julián Mestre, Sergey Pupyrev, and Seeun William Umboh. On the Extended TSP problem. In *Proc. of the 32nd International Symposium on Algorithms and Computation*, volume 212, pages 42:1–42:14, 2021.

34 Christos H. Papadimitriou and Mihalis Yannakakis. The traveling salesman problem with distances one and two. *Math. Oper. Res.*, 18(1):1–11, 1993.

35 Satish Rao and Andréa W. Richa. New approximation techniques for some linear ordering problems. *SIAM Journal on Computing*, 34(2):388–404, 2005.

36 Neil Robertson and Paul D Seymour. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986.

37 James B Saxe. Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM Journal on Algebraic Discrete Methods*, 1(4):363–369, 1980.

38 Wikipedia contributors. Hilbert curve. https://en.wikipedia.org/wiki/Hilbert_curve.

A Variants of MLogA

Most graph compression schemes build on *delta-encoding*, that is, sorting the adjacency lists (also called *posting lists*) so that the gaps between consecutive elements are positive, and then encoding these gaps using a variable-length integer code. For this reason, the *minimum logarithmic gap arrangement* (MLogGapA) problem is introduced [6, 11]. For a vertex $v \in V$ of degree k and an order π, consider the neighbors $out(v) = (v_1, \dots, v_k)$ of v such that $\pi(v_1) < \dots < \pi(v_k)$. Then the cost compressing the list $out(v)$ under π is related to $f_\pi(v, out(v)) = \sum_{i=1}^{k-1} \log|\pi(v_{i+1}) - \pi(v_i)|$. MLogGapA consists in finding an order π, which minimizes

$$\sum_{v \in V} f_\pi(v, out(v)).$$

Similarly to MLogA, the MLogGapA is known to be NP-hard [11]. Furthermore, Dhulipala et al. [11] experimentally verify that the cost of MLogGapA accurately predicts the compressed size of real-world instances for various modern encoding schemes.

For some applications, such as index compression, it is convenient to study a generalization of MLogA and MLogGapA by considering a bipartite graph with *query* and *data* vertices. To this end, let $G = (\mathcal{Q} \cup \mathcal{D}, E)$ be an undirected unweighted bipartite graph with disjoint sets of vertices $\mathcal{Q}$ and $\mathcal{D}$. The goal is to find a permutation, π, of data vertices, $\mathcal{D}$, so that the following objective is minimized:

$$\sum_{q \in \mathcal{Q}} \sum_{i=1}^{\deg_q - 1} \log(\pi(u_{i+1}) - \pi(u_i)),$$

where $\deg_q$ is the degree of query vertex $q \in \mathcal{Q}$, and q's neighbors are $\{u_1, \dots, u_{deg_q}\}$ with $\pi(u_1) < \cdots < \pi(u_{deg_q})$. The optimization problem is called *bipartite minimum logarithmic arrangement* (BiMLogA). Notice that BiMLogA is different from MLogGapA in that the latter does not differentiate between data and query vertices. It is easy to see that the new problem generalizes both MLogA and MLogGapA: to model MLogA, add a query vertex for every edge of the input graph; to model MLogGapA, add a query for every vertex of the input graph.

While according to Chierichetti et al. [6] and Dhulipala et al. [11] MLogGapA and BiMLogA are arguably more important for applications than MLogA, we find that the latter problem is interesting on its own from a theoretical point of view. Adjusting our results for the more general variants is a possible future direction.

B Omitted Proofs

▶ **Lemma 1.** *For any integer $\ell \geq 1$ we have*

$$(\ell - 1) \cdot \lg(\ell + 1) < \sum_{i=1}^{\ell} (1 + \lfloor \lg i \rfloor) < (\ell + 1) \cdot \lg(\ell + 1).$$

Proof. We can use integrals to prove the upper bound:

$$\sum_{i=1}^{\ell} (1 + \lfloor \lg i \rfloor) \leq \int_{1}^{\ell+1} 1 + \lg x \, \mathrm{d}x \leq (\ell + 1) \cdot \lg(\ell + 1).$$

And the lower bound:

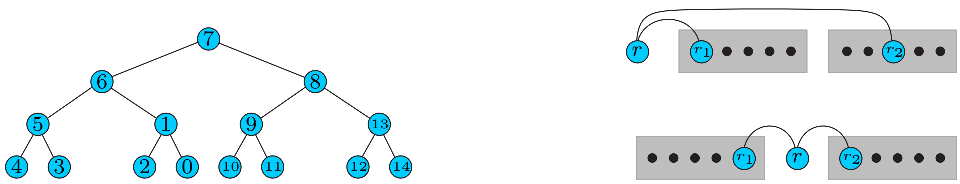
$$\sum_{i=1}^{\ell} (1 + \lfloor \lg i \rfloor) \geq \int_{1}^{\ell+1} \lg x \, \mathrm{d}x \geq (\ell - 1) \cdot \lg(\ell + 1).$$

◀

▶ **Lemma 2.** *Let $G = (V, E)$ be a graph with n vertices and m edges, then*

$$\mathrm{LGA}(G) \geq (m - n) \cdot \lg \frac{m}{n}$$

Proof. Consider a vertex, $v \in V$, and all incident edges. The optimal layout of the star subgraph is achieved when v is placed in the middle of the order and the neighbors occupy



(a) A complete binary tree with 4 levels numbered according to Lemma 6

(b) Two ways of embedding a binary tree, *side-based* (top) and *center-based* (bottom)

Figure 3 Embedding a complete binary tree with $\mathrm{LGA} \le \frac{5}{3}n$.

consecutive intervals to the left and to the right of v. Thus the edges incident on v contribute to the objective at least

$$\sum_{i=1}^{\lfloor \deg(u)/2 \rfloor} (1 + \lfloor \lg i \rfloor) + \sum_{i=1}^{\lceil \deg(u)/2 \rceil} (1 + \lfloor \lg i \rfloor) \ge (\deg(u) - 2) \cdot \lg \frac{\deg(u)}{2}$$

where the inequality follows from applying Lemma 1 to each sum.

Summing over all vertices and observing that every edge is counted twice, gives a global lower bound of

$$\mathrm{LGA}(G) \ge \sum_{u \in V} \frac{\deg(u) - 2}{2} \cdot \lg \frac{\deg(u)}{2} \ge (m - n) \cdot \lg \frac{m}{n}.$$

where the last inequality follows from Jensen's inequality and the fact $f(x) = (x-2) \cdot \lg x$ is a concave function, which means that the sum is minimized when all n terms are equal. ◀

▶ **Lemma 6.** *Let T_n denote the k-level complete binary tree with $n = 2^k - 1$ vertices. Then*

$$\mathrm{LGA}(T_n) \le \left\lceil \tfrac{5}{3}(2^k - 1) \right\rceil - k - 1 \le \tfrac{5}{3}n.$$

Proof. Consider a complete binary tree, T_n, with k levels such that $n = 2^k - 1$. Let r be the root of the tree connected to two copies of a complete tree with $k-1$ levels; see Figure 3a. In order to prove the claim, we consider two ways of embedding T_n: a *side-based* layout in which r is the rightmost (or leftmost) in the order, and a *center-based* layout in which r is positioned between the two copies of the subtrees, T_{n-1}. See Figure 3b for an illustration and observe that the vertices of each of the subtrees do not overlap in the resulting order.

Define the cost of embedding a complete binary tree with k levels using the side-based and center-based approaches by $S(k)$ and $C(k)$, respectively. It follows directly from the construction that

$$\begin{aligned}
C(k) &= 2 + 2S(k-1) && \text{and} \\
S(k) &= 2 + \lfloor \lg(2^{k-1} + 2^{k-2} - 1) \rfloor + S(k-1) + C(k-1) && \\
&= 1 + k + S(k-1) + C(k-1) && \text{for } k \ge 2 \text{ and} \\
C(1) &= S(1) = 0. &&
\end{aligned}$$

We claim that $C(k) = \left\lceil \frac{5}{3}(2^k - 1) \right\rceil - k - 1$ and $S(k) = C(k) + \left\lfloor \frac{k}{2} \right\rfloor$. By induction, the two bounds clearly hold for $k = 1$. For $k \ge 2$, we have

$$C(k) = 2 + 2S(k-1) = 2 + 2\Big(C(k-1) + \left\lfloor \tfrac{k-1}{2} \right\rfloor\Big) = 2 + 2\Big(\left\lceil \tfrac{5}{3}(2^{k-1} - 1) \right\rceil - k + \left\lfloor \tfrac{k-1}{2} \right\rfloor\Big).$$

Observe that for even k, we have $2^k \bmod 3 = 1$, while for odd k, it holds $2^k \bmod 3 = 2$. Thus, when $k = 2t$ is even, $2\lceil \frac{5}{3}(2^{k-1}-1)\rceil = \lceil \frac{5}{3}(2^k-1)\rceil - 1$; therefore,

$$C(k) = 2 + \lceil \tfrac{5}{3}(2^k-1)\rceil - 1 - 4t + 2\lfloor \tfrac{2t-1}{2}\rfloor = \lceil \tfrac{5}{3}(2^k-1)\rceil - 2t - 1.$$

Similarly, when $k = 2t+1$, we have $2\lceil \frac{5}{3}(2^{k-1}-1)\rceil = \lceil \frac{5}{3}(2^k-1)\rceil - 2$; therefore,

$$C(k) = 2 + \lceil \tfrac{5}{3}(2^k-1)\rceil - 2 - 2(2t+1) + 2\lfloor \tfrac{2t}{2}\rfloor = \lceil \tfrac{5}{3}(2^k-1)\rceil - (2t+1) - 1.$$

The inductive step for $S(k)$ is verified analogously.

Finally, observe that $\mathrm{LGA}(T_n) \le \min(C(k), S(k))$, which proves the desired bound. ◀

▶ **Lemma 7.** *Let $G_{h,h}$ denote the $h \times h$ grid graph with h being a power of two. Then*

$$\mathrm{LGA}(G_{h,h}) \le 4h^2 = \mathcal{O}(m).$$

Proof. At a very high level, the Hilbert curve orders the points in the unit square by recursively dividing it into four smaller squares, visiting each of smaller square in turn and concatenating the partial traversals. This construction yields a hierarchical decomposition of the grid. At the top level of the hierarchy we have a single square holding all h^2 points. One level down, at level 1, we have 4 smaller squares of $h/2 \times h/2$ each holding $h^2/4$ points. In general, level i has 4^i squares each holding $h^2/4^i$ points; see Figure 1b.

We say that an edge (u, v) is *cut at level i* if u and v belong to the same square at level i but different squares at level $i+1$. Notice that this means that the distance between u and v is no larger than the size of the the squares at level i, namely, $|\pi(u) - \pi(v)| \le h^2/4^i$. Furthermore, notice that there are $2h$ edges cut at level 0, $4h$ edges at level 1, and in general, $2^{i+1}h$ edges cut at level i.

Therefore, the objective value of the layout is upper bounded by

$$\begin{aligned}
\mathrm{LGA}(G_{h,h}) &\le \sum_{i=0}^{\lg h-1} 2^{i+1}h \cdot (1 + \lfloor \lg \frac{h^2}{4^i} \rfloor) \\
&\le 2h^2 + 2h \sum_{i=0}^{\lg h-1} 2^i \lg \frac{h^2}{4^i} \\
&\le 2h^2 + 2h \int_{i=1}^{\lg h} 2^x \lg \frac{h^2}{4^x} \mathrm{d}x \\
&\le 2h^2 + \frac{2}{\lg 2} h^2 \\
&\le 4h^2
\end{aligned}$$

Finally, we note that the grid contains $2h^2 - 2h$ edges, so the layout is at least 2-approximate. ◀

▶ **Lemma 9.** *The worst ratio $\rho(\mu) = \frac{\mathrm{UB}(\mu)}{\mathrm{LB}(\mu)}$ is attained when for any two nodes $u \in L_i$ and $v \in L_{i+1}$ in consecutive layers we have*

$$\frac{\mu_u}{\mu_v} = 2^{\frac{1}{\rho(\mu)}}.$$

Proof. Consider the operation of deviating slightly from the give vector μ to another vector increasing μ_u by a small δ amount while decreasing μ_v by the same amount. Let us denote with $\mu|\delta$ this new vector. And let $f(\delta) = \frac{\mathrm{UB}(\mu|\delta)}{\mathrm{LB}(\mu|\delta)}$

Assuming that μ is the vector maximizing the ratio, we expect that $f'(0) = 0$; for otherwise, we can deviate from μ and improve the ratio (either with $\delta > 0$ or $\delta < 0$ depending on the sign of $f'(0)$).

In order to derive the equation $f'(0) = 0$, we first compute the derivatives of the numerator $g(\delta) = \mathrm{UB}(\mu|\delta)$ and the denominator $h(\delta) = \mathrm{LB}(\mu|\delta)$:

$$g'(\delta) = \left(1 + \lg \frac{n}{2^i}\right) - \left(1 + \lg \frac{n}{2^{i+1}}\right) = 1$$

$$h'(\delta) = (2 + \lg(\mu + \delta)) - (2 + \lg(\mu_v - \delta)) = \lg \frac{\mu_u + \delta}{\mu_v - \delta}$$

We can write the constraint $f'(0) = 0$ in terms of these functions as follows

$$g'(0)\mathrm{LB}(\mu) - \mathrm{UB}(\mu)h'(0) = 0,$$

which we can re-write as

$$\frac{1}{\lg \frac{\mu_u}{\mu_v}} = \frac{\mathrm{UB}(\mu)}{\mathrm{LB}(\mu)} = \rho(\mu),$$

which in turn is equivalent to the relation shown in the lemma statement. ◀