Bounds on the Geometric Mean of Arc Lengths for Bounded-Degree Planar Graphs

Mohammad Khairul Hasan, Sung-Eui Yoon, and Kyung-Yong Chwa

Division of Computer Science
Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea
shaon@tclab.kaist.ac.kr, sungeui@kaist.edu, kychwa@tclab.kaist.ac.kr

Abstract. Data access time becomes the main bottleneck in applications dealing with large-scale graphs. Cache-oblivious layouts, constructed to minimize the geometric mean of arc lengths of graphs, have been adapted to reduce data access time during random walks on graphs. In this paper, we present a constant factor approximation algorithm for the Minimum Geometric Mean Layout (MGML) problem for bounded-degree planar graphs. We also derive an upper bound for any layout of the MGML problem. To the best of our knowledge, these are the first results for the MGML problem with bounded-degree planar graphs.

1 Introduction

Large-scale graphs are extensively used in variety of applications including data mining, national security, computer graphics, and social network analysis. Advances in data-capturing and modeling technologies have supported the use of large-scale graphs. Massive polygonal models–which are typically bounded degree planar graphs–are easily constructed using various techniques developed in computer-aided design (CAD), scientific simulation, and e-Heritage [1]. Social network analysis is a critical component to understanding various social behaviors, urban planning, and the spread of infectious diseases [2]. With hundreds of millions of nodes, complexity becomes a major challenge in analyzing large-scale graphs.

To process large-scale graphs in a timely manner, efficient processing algorithms have utilized the high computational power of CPUs and Graphics Processing Units (GPUs). Over the last few decades, the widening gap between processing speed and data access speed has been a major computing trend. For nearly two decades, the CPU performance has increased at an annual rate of 60%, meanwhile the disk access time has been decreased by an annual rate of 7%-10% [3].

In order to avoid high-latency data accesses, system architectures increasingly utilize caches and memory hierarchies. Memory hierarchies have two main characteristics. First, lower level memory have higher storage capacities and longer data access times. Second, whenever a cache miss occurs, data is transferred in a block containing data between different memory levels. Typically, nodes and arcs of a graph are stored linearly in the memory space. There exists an exponential number of choices for storing nodes and arcs in the one dimensional memory space. An ordering of nodes or arcs within the one dimensional memory space is a *data layout*. Given a block-based caching architecture with slower data access times for lower memory levels, it is critical to store data that are likely to be accessed together very close in the data layout.

The geometric mean of arc lengths for a graph has been shown to have a high linear correlation with the number of cache misses occurred while accessing the graph in the data layout [4]. Therefore the geometric mean of arc lengths is a *cache-coherent metric*. Efficient layout construction algorithms that reduce the geometric mean of input graphs have been used to improve cache performance. The constructed *cache-coherent layouts* have been demonstrated to show significant performance improvement over other tested layouts. However, it has not been proven that the constructed layouts are optimal in terms of geometric mean.

In this paper we investigate an optimal algorithm that minimizes the value of the geometric mean of arc lengths for a class of bounded-degree planar graphs.

Problem statement and main results: We start by defining a Minimum Geometric Mean Layout (MGML) problem for a graph. Given a graph, $\mathscr{G} = (\mathscr{N}, \mathscr{A})$, where $n = |\mathscr{N}|$ and $m = |\mathscr{A}|$, a layout, $\varphi : \mathscr{N} \rightarrow [n] = \{1, ..., n\}$, is a one-to-one mapping of nodes to position indices.

The geometric mean, *GMean*, of arc lengths for a layout φ of graph $\mathscr{G}$ is:

$$GMean = \left(\prod_{(u,v)\in\mathscr{A}} |\varphi(u) - \varphi(v)| \right)^{\frac{1}{m}} = 2^{\log\left(\left(\prod_{(u,v)\in\mathscr{A}} |\varphi(u)-\varphi(v)|\right)^{\frac{1}{m}}\right)}$$
$$= 2^{\frac{1}{m}\sum_{(u,v)\in\mathscr{A}} \log(|\varphi(u)-\varphi(v)|)}$$

$\varphi^* : \mathscr{N} \rightarrow [n]$ is defined as an optimal layout in terms of the minimum geometric mean of arc lengths. The geometric mean with the optimal layout, $GMean^{OPT}$, is similarly defined by replacing φ with φ^*. Throughout this paper we will assume that the "log" function is of base 2.

In this paper we propose a layout construction algorithm which, for a bounded degree planar graph, gives a layout with a geometric mean that is at most a constant factor worse than the geometric mean of the optimal layout. To the best of our knowledge, this is the first result in this area. We also show that the geometric mean of arc lengths for an arbitrary layout is $O(n)$, where n is the number of nodes in the graph.

We discuss works related to our problem in Sec. 2 followed by a background on computing cache-coherent layouts of graphs in Sec. 3. Our theoretical results are explained in Sec.4 and we conclude with related open problems in Sec. 5.

2 Related Work

In this section, we briefly introduce works related to our graph layout problem.

2.1 Graph Layout Optimization

For a graph layout, we can formulate a cost function which indicates how good the layout is. Layout problems with common cost functions have attracted much researcher attention in recent years. In most cases the problems are NP-complete [5–7] for general graphs. The most popular cost functions include arithmetic mean of arc lengths (also known as Minimum Linear Arrangement (MLA) problem), bandwidth, etc.

Arithmetic Mean: Minimizing the arithmetic mean or MLA, is NP-complete [5], thus, researchers are interested in approximation algorithms. The first approximation algorithm for MLA was derived by Hansen [8] with a factor of $O(\log n)^2$. Currently the best known algorithm for MLA [9] has a factor of $O(\sqrt{\log n}\log\log n)$. MLA has polynomial algorithms when the input graph is a tree [10] and linear algorithms when the input graph is a rectangular grid [11].

Bandwidth: Another interesting problem related to graph layout is the Bandwidth, where the objective is to minimize the maximum arc length. This problem is NP-Hard for general graphs [6], for trees with maximum degree 3 [12], and for grid and unit disk graphs [13]. This problem has polynomial algorithms for a variety of graph families. More details can be found in a recent survey of graph layout problems [14].

Geometric Mean: The geometric mean of arc lengths for a graph has recently been proposed to measure the cache-coherence of a graph layout in a cache-oblivious case [4]. Geometric mean is strongly correlated to the number of cache misses observed during the random access of the meshes and graphs in various applications. It is assumed that the block sizes encountered at runtime are power-of-two bytes. Then, the expected number of cache misses with those power-of-two bytes cache blocks is computed. More background information on the derivation of geometric mean will be presented in Sec. 3.

2.2 Space-Filling Curves

Space-filling curves [15] have been widely used in applications to improve cache utilization. Space-filling curves assist in computing cache-friendly layouts for volumetric grids or height fields. The layouts yield considerable performance improvements in image processing [16] and terrain or volume visualization [17, 18]. A standard method of constructing a layout for graph using a space-filling curve is to embed the graph in a geometrically uniform structure that contains the space-filling curve. To embed the graph in such a uniform structure (e.g., grid) geometrically, each node of the graph must contain geometric information.

Gotsman and Lindenbaum investigated the spatial locality of space-filling curves [19]. Motivated by searching and sorting applications, Wierum [20] proposed a logarithmic measure resembling the geometric mean of the graph for analyzing space-filling curve layouts of regular grids.

Although space-filling curves are considered a good heuristic for designing cache-coherent layouts, their applications are limited to regular geometric data structures like grids, images, or height fields. This is mainly due to the nature of the procedure of embedding onto uniform structures of space-filling curves. A multi-level layout construction method minimizing the geometric mean [1] can be considered as a generalization method of classic space-filling curves to arbitrary graphs.

3 Background on Cache-Coherent Layouts

In this section we give background information on computing cache-coherent layouts.

3.1 Block-based I/O Model

Caching schemes are widely used to bridge the widening gap between data processing speed and data access speed. The two main characteristics of modern memory hierarchies are, 1) varying data storage size and data access time between memory levels and 2) the block-based data fetching. Those characteristics are well captured in the *two-level I/O-model* defined by Aggarwal and Vitter [21]. This model assumes a fast memory called "cache," consisting of M blocks, and a slower infinite memory. Each cache block is B; therefore the total cache size is $M \times B$. Data is transferred between the different levels in blocks of consecutive elements.

3.2 Cache-Oblivious Layout Computation

There have been research efforts to design cache-coherent layouts that reduce the number of cache misses during accessing these graphs and meshes in various applications [4, 22, 1, 23]. Those approaches cast the problem of computing a cache-coherent layout of a graph to an optimization problem of minimizing a cost metric, which measures the expected number of cache misses for the layout during random walks on a graph. Those methods also utilize an input graph representing the anticipated runtime access pattern. Each node of the graph represents a data element and a direct arc (i, j) between two nodes indicates an anticipated access pattern; a node j may be accessed sequentially after a node i.

3.3 Geometric Mean

Cache-coherent layouts can be classified into cache-aware and cache-oblivious layouts. Cache-aware layout are constructed by optimizing the expected number of cache misses given a particular cache block size. On the other hand, cache-oblivious layouts are constructed without optimizing at a particular cache block size. Instead, cache-oblivious layouts are constructed by considering all the possible block sizes that may be encountered at runtime. Cache-oblivious layouts can yield significant benefits on architectures that employ various cache block sizes.

For the cache-aware layouts, it has been derived that the number of straddling arcs of an input graph in a layout determines the number of observed cache misses during random access at runtime [4]. The expected number of cache misses for a cache-oblivious layout is computed by considering power-of-two bytes block sizes. The result can be derived from that of the cache-aware case. In cache-oblivious layouts, the expected number of cache misses is reduced to the geometric mean of arc lengths of the graph. More importantly, the geometric mean has been demonstrated to have a strong correlation to the numbers of observed cache misses at runtime for various layouts.

3.4 Multi-level Construction Method

In order to construct cache-oblivious layouts that optimize the cache-oblivious metric (i.e., the geometric mean) multi-level construction methods have been introduced [4, 1]. The idea is to partition an input graph into a small number of subsets while minimizing the number of straddling arcs among partitioned sets.

After partitioning into subsets, orders of partitioned sets are computed while minimizing the geometric mean. This process is recursively performed on each partitioned set until each set has one node. As we partition each set and compute an ordering among the partitioned sets, the global ordering of nodes for the graph is progressively refined. Note that the overall process of multi-level construction methods is similar to that of space-filling curves like Hilbert-curve [15].

This method runs quite fast and produces high-quality cache-oblivious layouts. However, there have been no theoretical approaches investigating the optimality of the resulting layouts.

4 Theoretical Results

In this section, we present a constant factor approximation algorithm for the Minimum Geometric Mean Layout (MGML) problem for bounded-degree planar graphs. We also demonstrate that the geometric mean of arc lengths for an arbitrary layout of a planar graph is $O(n)$, where n is the number of nodes of the graph. More formally, our algorithm and its analysis are based on the assumption that the input graph is a n node planar graph whose maximum degree is bounded by a constant k.

Graphs satisfying the above assumptions are widely used in the field of computer graphics and are known as meshes. For the sake of clarity, we assume that the input graph is connected and the number of nodes in the input graph is a power of two. We can assume this without loss of generality because dummy nodes and arcs can be added as appropriate to make the total number of nodes a power of two and to make the graph connected. After applying our algorithm on the modified graph we can get the layout of the original graph simply by taking the relative ordering of real nodes from the generated layout without increasing the geometric mean of arc lengths for the original graph.

Throughout this paper we use the following notations. $\mathscr{G} = (\mathscr{N}, \mathscr{A})$ is our input graph with $n = |\mathscr{N}|$ and $m = |\mathscr{A}|$, where n is a power of 2 and $m = \theta(n)$. For arbitrary graph $G = (N, A)$ and $X \subseteq N$, $G[X]$ is the subgraph of G induced by X. Also for a graph $G = (N, A)$ and for any two node sets $P \subseteq N$ and $Q \subseteq N$, $Crossing(P, Q)$ is the set of arcs $A_x \subseteq A$ each of which has one node in P and another node in Q. For any graph G, $N(G)$ and $A(G)$ are the set of nodes and the set of arcs of the graph respectively.

4.1 Algorithm

We use a divide and conquer approach to find a layout φ given a graph. We use two algorithms, **Partition** and **MakeLayout**. Algorithm **Partition** takes a planar graph with a maximum degree bounded by a constant k as an input. Using the procedure described in [24] **Partition** algorithm computes a partition (P, Q) of nodes of the input graph such that $|P| = |Q|$ and $Crossing(P, Q)$ is bounded by $c \times \sqrt{k \times (|P| + |Q|)}$, where c is a constant. Theorem 1 and Corollary 1 ensure that such a partition can be generated.

Theorem 1 ([24]). *Let G be a planar graph with a maximum degree k. Then there exists a set of arcs of size $\leq 2\sqrt{2kn}$ such that by removing this arc set one can partition G into two subgraphs each having at most $\lceil \frac{2}{3} \times |N(G)| \rceil$ nodes. Furthermore such an arc set can be found in linear time.*

Corollary 1 ([24]). *For any planar graph G with a maximum degree k there exists an arc set R with size $\leq (6\sqrt{2}+4\sqrt{3})\sqrt{kn}$ whose removal divides the nodes of G into two sets of P and Q such that $|P| \leq \lceil |N(G)|/2 \rceil$, $|Q| \leq \lceil |N(G)|/2 \rceil$, and the arc set connecting P and Q belongs to R.*

Corollary 1 is based on both Theorem 1 and the divide-and-conquer partitioning method described in [25]. Assuming that n is a power of two, a n-node planar graph G with maximum degree bounded by k can be partitioned into two subgraphs with equal nodes by removing at most $(6\sqrt{2}+4\sqrt{3})\sqrt{kn}$ arcs in $O(n\log n)$ time.

The **MakeLayout** algorithm takes an input graph and two indices indicating the beginning and ending positions of the layout. Generating the layout for the entire input graph involves an initial call to **MakeLayout**$(\mathscr{G},1,n)$: other calls are made recursively. The algorithm first starts by checking if the input graph contains only one node. If so, it constructs a layout consisting of that node. Otherwise it splits the graph into two parts using the **Partition** algorithm. We recurse this process until a graph consisting of only one node is reached.

```
Algorithm Partition ($G$)
01      Input: A bounded degree planar graph $G$
02      Output: A partition $(P,Q)$ of $N(G)$
03      Begin
04          Find a partition $(P,Q)$ of $N(G)$ such that:
05              1. $|P| = |Q|$
06              2. $|Crossing(P,Q)| \leq c \times \sqrt{(k \times |N(G)|)}$
07          return $(P,Q)$
08      End.
```

```
Algorithm MakeLayout ($G,i,j$)
01      Input: A bounded degree planar graph $G$
02      Initial and end positions, $i$ and $j$, of any layout
03      Output: A layout of $G$
04      Begin
05          If $|N(G)| = 1$ then
06              Let $u$ be the only node of $G$
07              Set $\varphi(u) = i$
08          Else,
09              Let $(P,Q)$ be the partition of $N(G)$ found using Partition(G)
10              MakeLayout($G[P]$, $i$, $i+(\frac{j-i+1}{2})-1$)
11              MakeLayout($G[Q]$, $i+(\frac{j-i+1}{2})$, $j$)
12      End.
```

4.2 Analysis

First, we will show that the geometric mean of arc lengths for a layout generated by the **MakeLayout** algorithm is $O(1)$.

Because the number, n, of nodes in the input graph is a power of two, it is easy to see that the overall recursion tree of the **MakeLayout** algorithm is a complete binary tree with n leaves and exactly $\log n$ levels. Additionally, level l of this tree contains exactly 2^l nodes, i.e., 2^l calls to **MakeLayout**.

Each non-leaf node of the binary tree splits a graph into two subgraphs with equal nodes by removing crossing arcs between those two subgraphs. Let A_l be the set of arcs removed by all nodes of level l of the recursion tree and $Cost_l = \sum_{(u,v)\in A_l} \log(|\varphi(u) - \varphi(v)|)$. The following two lemmas give an upper bound for $Cost_l$ on each level l.

Lemma 1. *For each call of the algorithm* **MakeLayout(**G,i,j**)**, $j-i$ *is equal to* $|N(G)|-1$.

Proof. We will prove this lemma by induction on the level of recursion. At level 0 the algorithm is called with G as $\mathscr{G} = (\mathscr{N}, \mathscr{A})$, with i and j as 1 and n respectively. Since $|\mathscr{N}| = n$ the lemma is true for the level 0 call.

Suppose the lemma is true for any call at a level p. Consider a level p execution of the algorithm **MakeLayout(**G,i,j**)**. If $|N(G)| > 1$, this execution will make two level $p+1$ executions **MakeLayout(**$G[P], i, i+(\frac{j-i+1}{2})-1$**)** and **MakeLayout(**$G[Q], i+(\frac{j-i+1}{2}), j$**)** in line 10 and 11 in the algorithm **MakeLayout** respectively. Let $G_0 = G[P], i_0 = i, j_0 = i+(\frac{j-i+1}{2})-1$ and $G_1 = G[Q], i_1 = i+(\frac{j-i+1}{2}), j_1 = j$. Then $j_0 - i_0 = \frac{j-i+1}{2} - 1 = \frac{(|N(G)|-1)+1}{2} - 1 = |N(G)|/2 - 1 = |P| - 1 = |N(G_0)| - 1$. Similarly, $j_1 - i_1 = j - i - \frac{j-i+1}{2} = \frac{j-i-1}{2} = \frac{j-i+1}{2} - 1 = |N(G_1)| - 1$. Since every level $p+1$ execution has been generated from a level p execution with a graph having more than one node, the lemma follows. □

Lemma 2. *For each level l, $Cost_l \le c \times \sqrt{kn} \times (\sqrt{2})^l \times \log \frac{n}{2^l}$, where n and k are the number of nodes and maximum degree of $\mathscr{G}$ respectively and c is a constant.*

Proof. If $l = \log n$ (i.e., the leaf level of the recursion) then $Cost_l = 0$ and also $\log \frac{n}{2^l} = 0$. Therefore, the lemma is trivially true. Let us assume that $0 \le l \le \log n - 1$. Consider an execution of **MakeLayout(**G,i,j**)** at a level l. Since $l \le \log n - 1$, $|N(G)| > 1$. Algorithm **MakeLayout** uses algorithm **Partition** to split the graph G into $G[P]$ and $G[Q]$ by removing arc set $Crossing(P,Q)$. Consider an arc $a = (u,v) \in Crossing(P,Q)$ such that $u \in P$ and $v \in Q$. It is easy to see that at the end of the algorithm $\varphi(u) \ge i$ and $\varphi(v) \le j$. Therefore, $\log(|\varphi(u) - \varphi(v)|) \le \log(j-i) = \log(|N(G)|-1) = \log(\frac{n}{2^l} - 1) < \log \frac{n}{2^l}$; the first equality follows from Lemma 1 and the last equality follows from the fact that in each recursive call each child graph has exactly half of the nodes of the parent graph.

Since there are exactly 2^l nodes at the level l of the recursion tree, $Cost_l < 2^l \times |Crossing(P,Q)| \times \log \frac{n}{2^l}$. The lemma follows from the fact that line 6 of the **Partition** algorithm ensures that $|Crossing(P,Q)| \leq c \times \sqrt{k \times |N(G)|}$ and $|N(G)| = \frac{n}{2^l}$. □

Lemma 3. $\sum_{l=0}^{\log n} Cost_l \leq \frac{c \times \sqrt{2k}}{(\sqrt{2}-1)^2} \times n$

Proof. The proof can be found in the appendix. □

Lemma 4. *The algorithm* **MakeLayout** *generates a layout of graph* $\mathscr{G} = (\mathscr{N}, \mathscr{A})$ *with a geometric mean of arc lengths =* $O(1)$.

Proof. The value of the geometric mean,

$$GMean = \left(\prod_{(u,v)\in\mathscr{A}} |\varphi(u) - \varphi(v)| \right)^{\frac{1}{m}} = 2^{\log\left(\left(\prod_{(u,v)\in\mathscr{A}} |\varphi(u)-\varphi(v)|\right)^{\frac{1}{m}}\right)}$$

$$= 2^{\frac{1}{m}\sum_{(u,v)\in\mathscr{A}} \log(|\varphi(u)-\varphi(v)|)} = 2^{\frac{1}{m}\sum_{l=0}^{\log n} Cost_l}$$

$$\leq 2^{\frac{1}{m}\frac{(c\times\sqrt{2k})}{(\sqrt{2}-1)^2}\times n} \quad \text{By Lemma 3}$$

$$= O(1)$$

The last equality follows from the fact that $m = \theta(n)$. □

Lemma 5. *The geometric mean of arc lengths for the optimal layout of a graph* $\mathscr{G} = (\mathscr{N}, \mathscr{A})$ *is* $\Omega(1)$

Proof. Since the optimal layout $\varphi^* : \mathscr{N} \rightarrow \{1, ..., n\}$ is a bijective function, for any arc $(u,v) \in \mathscr{A}$, $|\varphi^*(u) - \varphi^*(v)| \geq 1$. Thus,

$$GMean^{OPT} = \left(\prod_{(u,v)\in\mathscr{A}} |\varphi^*(u) - \varphi^*(v)|\right)^{\frac{1}{m}} \geq \left(\prod_{(u,v)\in\mathscr{A}} 1\right)^{\frac{1}{m}} = 1$$

□

Using Lemma 4 and Lemma 5 we get the following result.

Theorem 2. *The* **MakesLayout** *algorithm generates a layout for a graph* $\mathscr{G}$ *in* $O(n \log^2 n)$ *time such that the geometric mean of arc lengths of the graph is at most constant times the geometric mean of arc lengths of the optimal layout.*

4.3 Upper Bound of a Layout

Consider any arbitrary layout ψ of a graph $\mathscr{G} = (\mathscr{N}, \mathscr{A})$. Let $(u,v) \in \mathscr{A}$ be an arbitrary arc of $\mathscr{G}$. Then $1 \leq \psi(u) \leq n$, where $n = |\mathscr{N}|$. So, $\log(|\psi(u) - \psi(v)|) < \log n$. Then the geometric mean of arc lengths for this layout is:

$$\begin{aligned} GMean^{ANY} &= \left(\prod_{(u,v)\in\mathscr{A}} |\psi(u)-\psi(v)| \right)^{\frac{1}{m}} = 2^{\frac{1}{m}\sum_{(u,v)\in\mathscr{A}} \log(|\psi(u)-\psi(v)|)} \\ &< 2^{\frac{1}{m}\sum_{(u,v)\in\mathscr{A}} \log n} = 2^{\frac{\log n}{m} m} \\ &= n \end{aligned}$$

This observation along with Lemma 5 gives the following result.

Theorem 3. *Any algorithm which generates a layout for a planar graph is an $O(n)$ approximation algorithm for the MGML problem for planar graphs.*

5 Conclusion

We have presented a constant factor layout construction algorithm for the Minimum Geometric Mean Layout (MGML) with bounded-degree planar graphs problem. To the best of our knowledge, this is the first such result in this area. Our results are easily applicable to grids because grids belong to the class of bounded degree planar graphs.

Since the MGML problem has strong practical applications, especially for dealing with large-scale graphs, we seek to consider other classes of graphs. Particularly, we will be investigating an optimal algorithm and bounds for trees, a popular data structure used in various applications.

Acknowledgements

We would like to thank Peter Lindstrom, who first identified that the geometric means of space-filling curves of grids converge to a constant, as the grid size increases. This observation strongly supported that there is possibility that we can have a constant-factor approximation algorithm in terms of minimizing the geometric means. We also thank Samuel Brice and group members of the KAIST Theory of Computation for their English review and helpful feedbacks.

This project was supported in part by KAIST, MKE/MCST/IITA [2008-F-033-02], MKE/IITA u-Learning, MKE digital mask control, MCST/KOCCA-CTR&DP-2009, KRF-2008-313-D00922, MSRA E-heritage, and DAPA & ADD contract(UD080042AD).

References

1. Yoon, S.E., Lindstrom, P., Pascucci, V., Manocha, D.: Cache-Oblivious Mesh Layouts. ACM Transactions on Graphics (SIGGRAPH) **24**(3) (2005) 886–893

2. Eubank, S., Kumar, V.S.A., Marathe, M.V., Srinivasan, A., Wang, N.: Structural and algorithmic aspects of massive social networks. In: SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2004) 718–727
3. Hennessy, J.L., Patterson, D.A., Goldberg, D.: Computer Architecture, A Quantitative Approach. Morgan Kaufmann (2007)
4. Yoon, S.E., Lindstrom, P.: Mesh layouts for block-based caches. IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization) **12**(5) (2006) 1213–1220
5. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified np-complete problems. In: STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing, New York, NY, USA, ACM (1974) 47–63
6. Papadimitriou, C.H.: The NP-completeness of the bandwidth minimization problem. **16**(3) (1976) 263–270
7. Gavril, F.: Some np-complete problems on graphs. In: 11th Conference on Information Science and Systems. (1977) 91–95
8. Hansen, M.: Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. Foundations of Computer Science, 1989., 30th Annual Symposium on (Oct-1 Nov 1989) 604–609
9. Feige, U., Lee, J.R.: An improved approximation ratio for the minimum linear arrangement problem. Inf. Process. Lett. **101**(1) (2007) 26–29
10. Goldberg, M.K., Klipker, I.A.: An algorithm for minimal numeration of tree vertices. Sakharth. SSR Mecn. Akad. Moambe **81**(3) (1976) 553–556 (In Russian).
11. Muradyan, D.O., Piliposyan, T.E.: Minimal numberings of a rectangular lattice. Akad. Nauk. Armyan. SRR **1**(70) (1980) 21–27 (In Russian).
12. Garey, M.R., Graham, R.L., Johnson, D.S., Knuth, D.E.: Complexity results for bandwidth minimization. **34**(3) (May 1978) 477–495
13. Díaz, J., Penrose, M.D., Petit, J., Serna, M.: Approximating layout problems on random geometric graphs. Journal of Algorithms **39** (2001) 2001
14. Díaz, J., Petit, J., Serna, M.: A survey of graph layout problems. ACM Comput. Surv. **34**(3) (2002) 313–356
15. Sagan, H.: Space-Filling Curves. Springer-Verlag (1994)
16. Velho, L., de Miranda Gomes, J.: Digital halftoning with space filling curves. In: ACM SIGGRAPH. (1991) 81–90
17. Lindstrom, P., Pascucci, V.: Visualization of large terrains made easy. IEEE Visualization (2001) 363–370
18. Pascucci, V., Frank, R.J.: Global static indexing for real-time exploration of very large regular grids. In: Supercomputing. (2001)
19. Gotsman, C., Lindenbaum, M.: On the metric properties of discrete space-filling curves. IEEE Transactions on Image Processing **5**(5) (1996) 794–797
20. Wierum, J.M.: Logarithmic path-length in space-filling curves. In: 14th Canadian Conference on Computational Geometry. (2002) 22–26
21. Aggarwal, A., Vitter, J.S.: The input/output complexity of sorting and related problems. Communications of ACM **31** (1988) 1116–1127
22. Yoon, S.E., Manocha, D.: Cache-efficient layouts of bounding volume hierarchies. Computer Graphics Forum (Eurographics) **25**(3) (2006) 507–516
23. van Emde Boas, P.: Preserving order in a forest in less than logarithmic time and linear space. Inf. Process. Lett. **6** (1977) 80–82
24. Diks, K., Djidjev, H., Sýkora, O., Vrto, I.: Edge separators of planar and outerplanar graphs with applications. J. Algorithms **14**(2) (1993) 258–279
25. Lipton, R.J., Tarjan, R.E.: A separator theorem for planar graphs. SIAM Journal on Applied Mathematics **36**(2) (1979) 177–189

Appendix

Proof of Lemma 3

$$\begin{aligned}
\sum_{l=0}^{\log n} Cost_l &\leq \sum_{l=0}^{\log n} c \times \sqrt{kn} \times (\sqrt{2})^l \times \log \frac{n}{2^l} \quad \text{By Lemma 2} \\
&= c\sqrt{kn} \times [\sum_{l=0}^{\log n} (\log n \times (\sqrt{2})^l - \sum_{l=0}^{\log n} l \times (\sqrt{2})^l] \\
&= c\sqrt{kn} \times \frac{(\sqrt{2})^{\log n+1} - (\sqrt{2}-1) \times \log n - (\sqrt{2})}{(\sqrt{2}-1)^2} \leq c\sqrt{kn} \times \frac{(\sqrt{2}) \times (\sqrt{n})}{(\sqrt{2}-1)^2} \\
&= \frac{(c \times \sqrt{2k})}{(\sqrt{2}-1)^2} \times n
\end{aligned}$$

□