New Approximation Techniques for Some Ordering Problems

Satish Rao * Andréa W. Richa †

Abstract

We describe logarithmic approximation algorithms for the NP-hard graph optimization problems of minimum linear arrangement, minimum containing interval graph, and minimum storage–time product. This improves on the best previous approximation bounds of Even, Naor, Rao, and Schieber for these problems by an $\Omega(\log \log n)$ factor.

We use the lower bound provided by the volume W of a spreading metric for each of the ordering problems above (as defined by Even et al.) in order to find a solution with cost at most a *logarithmic factor times* W for these problems. We develop a recursion where at each level we identify cost which, if incurred, yields subproblems with reduced spreading metric volume. Specifically, we present a divide-and-conquer strategy where the cost of a solution to a problem at a recursive level is C plus the cost of a solution to the subproblems at this level, *and* where the spreading metric volume on the subproblems is less than the original volume by $\Omega(C/\log n)$. This ensures that the resulting solution has cost $O(\log n)$ times the original spreading metric volume.

We note that this is an existentially tight bound on the relationship between the spreading metric volume and the true optimal values for these problems.

For planar graphs, we combine a structural theorem of Klein, Plotkin, and Rao with our new recursion technique to show that the spreading metric cost volumes are within an $O(\log \log n)$ factor of the cost of an optimal solution for the minimum linear arrangement, and the minimum containing interval graph problems.

1 Introduction.

We describe approximation algorithms that apply to the NP-hard graph optimization problems of finding a minimum linear arrangement, of finding a minimum containing interval graph, and of finding a minimum storage-time product [5]. An *α-approximation algorithm* is an algorithm that finds a solution to the respective problem whose cost is at most α times the cost of an optimal solution to the problem.

All the ideas that we use for approximating the minimum containing interval graph and the minimum storage–time product problems can be illustrated by the algorithms for the minimum linear arrangement problem. Thus, we restrict our exposition primarily to the minimum linear arrangement problem, which we define as follows: Let G be a graph with associated edge weights.

*Computer Science Division, Soda Hall, University of California, Berkeley, CA 94720-1776, E-mail: satishr@cs.berkeley.edu.

†Supported in part by NSF CAREER Award No. CCR–9985284 and NSF grant No. CCR 9900304. Department of Computer Science and Engineering, Arizona State University, Tempe AZ 85287, E-mail: aricha@asu.edu.

A preliminary version of this work appeared in *Proceedings of Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 1998.

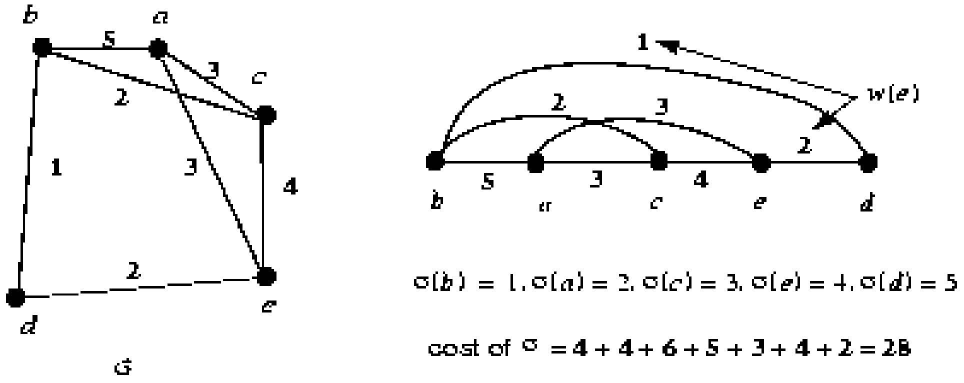

Figure 1: A graph G and a minimum linear arrangement σ of G.

Informally, a minimum linear arrangement (MLA) of G is an embedding[1] of G in the linear array such that (i) we have a one-to-one mapping from the nodes of G to the nodes of the linear array, and (ii) the weighted sum of the lengths of the edges of G — that is, the *cost* of the linear arrangement — is minimum. The length of an edge of G in the embedding is given by the distance between its two endpoints on the linear array. In Figure 1, we show a linear arrangement σ for the graph G with cost 28 (in fact this linear arrangement is a minimum linear arrangement of G).

Finding a minimum linear arrangement is NP-hard, even for the case when all the edges have unit weight. We present a polynomial time $O(\log n)$-approximation algorithm for the minimum linear arrangement problem on a graph with n nodes. This improves the best previous approximation bound of Even, Naor, Rao, and Schieber [2] for this problem by an $O(\log \log n)$ factor. If the graph is planar (or, more generally, if it excludes $K_{r,r}$ as a minor, for fixed r, where $K_{r,r}$ is the $r \times r$ complete bipartite graph), we obtain an $O(\log \log n)$ approximation factor for the minimum linear arrangement problem — improving on the best previous bound of $O(\log n)$ for these graphs — using a variation of the algorithm presented for the general case. We obtain this improvement by combining the techniques used for the general case with the algorithm presented by Klein, Plotkin, and Rao [8] for finding separators in graphs that exclude fixed $K_{r,r}$-minors.

We extend our approximation techniques (and bounds) to two other problems that involve finding a linear ordering of the nodes of a graph: the minimum containing interval graph, and the minimum storage–time product problems. Using techniques from [14], we can view the minimum containing interval graph problem (which we define formally later) as a "node version" of the minimum linear arrangement problem (see [2]). Thus, we also obtain an $O(\log n)$-approximation algorithm for general graphs, and an $O(\log \log n)$-approximation algorithm for graphs that exclude fixed $K_{r,r}$-minors for this problem. This improves on previous bounds of $O(\log n \log \log n)$ [2] for general graphs, and $O(\log n)$ for graphs that exclude fixed $K_{r,r}$-minors.

We can also use techniques from [14] to extend our ideas to produce an $O(\log T)$-approximation for the minimum storage-time product problem (where T is the sum of the processing times of all tasks), improving on a previous approximation bound of $O(\log T \log \log T)$ in [2]. The minimum storage–time product problem can be viewed as a generalization of the minimum linear arrangement problem, as explained in Section 5.

[1] An *embedding* of a graph G into a graph H maps nodes of G to nodes of H, and edges of G to paths in H. Typically, a guest network G is emulated by a host network H by embedding G into H. (For a more complete discussion of emulations and embeddings, see [9].).

A small variation of our algorithm (as presented in [1]) can be used to provide $O(\log n)$-approximation to the minimum cut linear arrangement and the minimum pathwidth problems.

Our approximation techniques rely on a lower bound W on the cost of an optimal solution provided by a *spreading metric* (to be defined soon), for each of the problems considered: We find a solution to the general problem that has cost $O(W \log n)$ ($O(W \log T)$ for the minimum storage–time product problem). Alon and Seymour [15] proved that there exists a logarithmic gap between the spreading metric cost volumes, and the true optimal values for certain instances of the problems of minimum linear arrangement, minimum containing interval graph, and minimum storage–time product. Thus we provide an existentially tight bound on the relationship between the spreading metric cost volumes and the true optimal values for these problems.

1.1 Previous Work.

Leighton and Rao [10] presented an $O(\log n)$-approximation algorithm for balanced partitions of graphs. Among other applications, this provided $O(\log^2 n)$-approximation algorithms for the minimum feedback arc set, and for the minimum-cut linear arrangement problem. Hansen [6] used the ideas in [10] to present $O(\log^2 n)$-approximation algorithms for the minimum linear arrangement problem, and for the more general problem of graph embeddings in d-dimensional meshes. Ravi, Agrawal, and Klein [14] presented polynomial time approximation algorithms that deliver a solution with cost within an $O(\log n \log T)$ factor from optimal for the minimum storage–time product problem, where T is the sum of the processing times of all tasks, and within an $O(\log^2 n)$ factor from optimal for the minimum containing interval graph.

Seymour [15] was the first to present a directed graph decomposition divide-and-conquer approach that does not rely on balanced cuts. He presented a polynomial time $O(\log n \log \log n)$-approximation algorithm for the minimum feedback arc set problem. Even, Naor, Rao, and Schieber [2] extended the spreading metric approach used by Seymour to obtain polynomial time $O(\log n \log \log n)$-approximation algorithms for the minimum linear arrangement, and the minimum containing interval graph problems, and an $O(\log T \log \log T)$-approximation algorithm for the minimum storage–time product problem. Even et al. actually showed similar approximation results for a broader class of graph optimization problems, namely the ones that satisfy their "approximation paradigm": A graph optimization problem where their divide-and-conquer approach is applicable, and for which a spreading metric exists satisfies this paradigm. They presented polynomial time $O(\min\{\log W \log \log W, \log k \log \log k\})$-approximation algorithms for these problems, where k denotes the number of "interesting" nodes in the problem instance (clearly $k \leq n$), and W is a lower bound on the cost of an optimal solution for the optimization problem provided by a spreading metric. Examples of such problems, besides the ones already mentioned, are graph embeddings in d-dimensional meshes, symmetric multicuts in directed networks, multiway separators and ρ-separators (for small values of ρ) in directed graphs. For a detailed description of each of those problems, see [2].

Even, Naor, Rao, and Schieber [3] extended the spreading metric techniques to graph partitioning problems. They used simpler recursions that yield a logarithmic approximation factor for balanced cuts and multiway separators. However, they were not able to extend this simpler technique to obtain a logarithmic approximation bound for the other problems considered in [2].

Recently, Bornstein and Vempala [1] proposed an alternative unified approach for obtainining lower bounds for the problems considered in this work. They present a unified linear program framework which defines "flow metrics" for each of the problems considered. Flow metrics can be viewed as a special case of spreading metrics. By simply varying the objective function of the linear program, they obtain lower bounds for each of the problems, which they use in conjunction with the divide-and-conquer algorithm presented in this work (and preliminary in [13]) to match our approximation factors for the three problems considered. They also present a minor variation of our algorithm which can be used to obtain $O(\log n)$-approximations for the minimum cut linear arrangement and minimum pathwidth problems.

1.2 Spreading Metrics and Our Recursion.

Our algorithms use an approach that relies on *spreading metrics*. Spreading metrics have been used in recent divide-and-conquer techniques to obtain improved approximation algorithms for several graph optimization problems that are NP-hard [2]. These techniques perform the divide step according to the *cost* of a solution to the subproblems generated, rather than according to the *size* of such subproblems.

A spreading metric on a graph is an assignment of lengths to the edges or nodes of the graph that has the property of "spreading apart" (with respect to the metric lengths) all the nontrivial connected subgraphs. The *volume* of the spreading metric is the sum, taken over all edges (resp., nodes), of the length of each edge (resp., node) multiplied by its weight.

For each of the optimization problems considered in this paper, Even, Naor, Rao, and Schieber [2] showed how to find a spreading metric of volume W such that W is a lower bound on the cost of a solution to the problem. Our techniques are based on showing that a spreading metric of volume W can be used to find a solution to the respective problem with cost $O(W \log n)$ ($O(W \log T)$, for the minimum storage–time product problem).

In this paper, we develop a recursion where at each level we identify cost which, if incurred, yields subproblems with reduced spreading metric volume. Specifically, we present a divide-and-conquer strategy where the cost of a solution to a problem at a recursive level is C plus the cost of a solution to the subproblems, *and* where the spreading metric volume on the subproblems geenrated is less than the original volume by $\Omega(C/\log n)$ (resp., $\Omega(C/\log T)$ for the minimum storage–time product problem). We will show that this ensures that the resulting solution has cost $O(\log n)$ (resp., $O(\log T)$) times the original spreading metric volume.

The recursion is based on divide-and-conquer — that is, we find an edge or node set whose removal divides the graphs into subgraphs, and then recursively order the subgraphs. The cost of a recursive level is the cost associated with the edges (or nodes) in the cut selected at this level. Previous recursive methods and analyses proceeded by finding a small cutset where the maximum spreading metric volumes of the subproblems were quickly reduced. We proceed by finding a *sequence* of cutsets whose total cost can be upper bounded, say by a quantity C, and whose total spreading metric volume is $\Omega(C/\log n)$ (resp., $\Omega(C/\log T)$), as stated above. The crux of the argument is that the cost associated with an edge (or node) in a cutset can be bounded by the number of nodes between the previous and the next cutset in the sequence.

We point out that the methods in [2] applied to more problems, including the d-dimensional graph embedding problem, and the minimum feedback arc set problem [15]. We could not extend our methods to these other problems, since we were unable to find a suitable bound on the cost of a sequence of cutsets associated with any of these problems.

Finally, for planar graphs and other undirected graphs that exclude some fixed minors, we combine a structural theorem of Klein, Plotkin and Rao [8] with our new recursion techniques, to show that the spreading metric cost volumes are within an $O(\log \log n)$ factor of the cost of the optimal solution for the minimum linear arrangement, and the minimum containing interval graph problems.

1.3 Overview.

In Section 2, we present a formal definition of the minimum linear arrangement problem and define the spreading metric used for this problem. In Section 3, we present a polynomial time $O(\log n)$-approximation algorithm for the minimum linear arrangement problem on an arbitrary graph with n nodes and nonnegative edge weights. In Section 4, we show how to improve this approximation factor to $O(\log\log n)$, in case the graph has no fixed $K_{r,r}$-minors — e.g., the graph is planar. In Sections 5 and 6, we define and briefly discuss the algorithms for approximating the minimum storage–time product problem and minimum containing interval graph problem, respectively.

2 The Minimum Linear Arrangement Problem

The *minimum linear arrangement (MLA) problem* is defined as follows: Given an undirected graph $G(V,E)$, with n nodes, and nonnegative edge weights $w(e)$, for all e in E, we would like to find a linear arrangement of the nodes $\sigma : V \to \{1,\ldots,n\}$ that minimizes the sum, over all $(i,j) \in E$, of the weighted edge lengths $|\sigma(i) - \sigma(j)|$. In other words, we would like to minimize the *cost*

$$\sum_{(i,j)\in E} w(i,j)\,|\sigma(i) - \sigma(j)|.$$

of a linear arrangement σ. In the context of VLSI layout, $|\sigma(i) - \sigma(j)|$ represents the length of the interconnection between i and j.

We now define the spreading metric used in the algorithms for the MLA problem presented in Sections 3 and 4. Analogous functions are used when approximating the minimum storage–time product problem (as presented in Section 5), and the minimum containing interval graph problem (see Section 6). Here we present the concept of spreading metrics in the context of the MLA problem (see [2] for a more general definition).

A *spreading metric* is a function $\ell : E \to \mathbf{Q}$ that assigns rational lengths to every edge in E, and that can be computed in polynomial time. It also satisfies the two properties below. The *volume* of a spreading metric ℓ is given by $\sum_{e\in E} w(e)\ell(e)$.

1. *Diameter guarantee:* Let the distances be measured with respect to the lengths $\ell(e)$. The distances induced by the spreading metric "spread" the graph and all its nontrivial subgraphs. In this application, this translates to "The diameter of every nontrivial connected subgraph U of V is $\Omega(|U|)$".

2. *Lower bound:* The minimum volume of a spreading metric is a lower bound on the cost of a MLA of G.

A solution ℓ to (1–3) is a spreading metric for the MLA problem (see [2]). Let $\mathcal{V}$ denote the set of all nontrivial connected subgraphs of V.

$$\begin{aligned} W = \quad & \min_{e\in E} w(e)\ell'(e) && (1)\\ \text{s.t.} \quad & \frac{(\sum_{u\in U} \mathsf{dist}(u,v))}{|U|} \geq \frac{|U|}{4}, \quad \forall v \in U, \forall U \in \mathcal{V} && (2)\\ & \ell'(e) \geq 0, \qquad \forall e \in E && (3)\end{aligned}$$

where $\mathsf{dist}(u,v)$ is the length of a shortest path from u to v according to the lengths $\ell'(e)$. The metric ℓ can be computed in polynomial time (see [2]) using, e.g., the ellipsoid method. (There

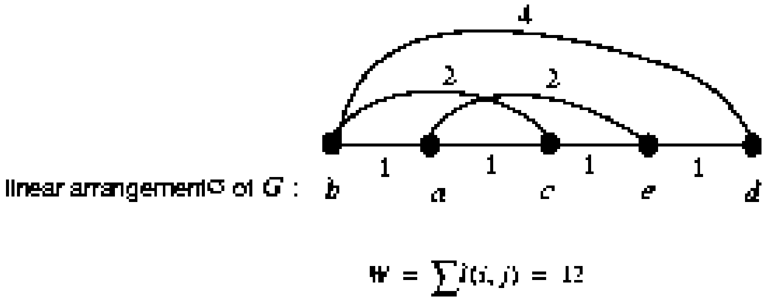

Figure 2: An assignment of lengths to the edges of G.

may be an exponential number of constraints in (2)). Note that (2) actually implies that $\ell(e) \geq 1$, for all e in E — simply consider the subsets U that consist of a single edge and its endpoints.

A solution ℓ to (1–3) is a *lower bound* on the cost of an MLA, since for any linear ordering σ of the nodes of G, the assignment of lengths to the edges of G given by $\ell'(i,j) = |\sigma(i) - \sigma(j)|$ satisfies (2– 3). The volume of such an assignment is exactly the cost of σ. In particular this is true for a MLA σ. Hence $W = \sum_{e \in E} w(e)\ell(e)$ is less than or equal to the cost of a MLA (Note that this lower bound is existentially tight, since there exist instances of this problem such that $\ell(i,j) = |\sigma(i) - \sigma(j)|$, where σ is a MLA of G, as for example, when G is a linear array.). We will use this fact later, when proving Theorems 1 and 2.

Figure 2 illustrates an assignment of lengths for the linear arrangement σ given by the ordering of the nodes of G from left to right (the lengths $\ell'(i,j)$ are the numbers associated with the edges in that picture; w.l.o.g.[2], assume that all the edge weights are 1).

Let ℓ be a spreading metric of volume $W = \sum_{e \in E} w(e)\ell(e)$ that satisfies (1–3). In the remainder of this paper all the distances in G are measured with respect to ℓ.

3 The Algorithm.

We now present our $O(\log n)$-approximation algorithm for the MLA problem on general graphs. Let $G(V,E)$ be a graph with nonnegative edge weights $w(e)$. Assume w.l.o.g. that G is connected (otherwise consider each connected component of G separately), and that all the edge weights $w(e)$ are at least equal to 1. For integer x we use the notation $[x]$ to denote the set $\{0, \ldots, x-1\}$.

In this paragraph, we introduce the notion of a *level* according to ℓ. Fix a node v in V. An edge (x,y) belongs to *level* i with respect to v if and only if $\mathsf{dist}(v,x) \leq i$ and $\mathsf{dist}(v,y) > i$, for any $i \in \mathbf{N}$. Note that an edge may belong to more than one level, and that there may be edges that do not belong to any level. Let the *weight of level* i, denoted by ρ_i, be the sum of the weights of the edges at level i. W.l.o.g., we will assume that $\log W$ is an integer. Let $\alpha_k = 2^k$, for all k in $[(\log W) + 1]$. Level i has *index* k, k in $[\log W]$, if and only if ρ_i belongs to the interval $I_k = (\alpha_k, \alpha_{k+1}]$.

It follows from (2) that we must have at least $n/4$ distinct levels with nonzero weight. Note that since $w(e) \geq 1$ and $\ell(e) \geq 1$, for all e, any level with nonzero weight must have weight at least 1. Since there are $\log W$ distinct level indices, there must be at least $n/(4 \log W)$ levels with same index k, for some k. Let κ be the exact number of levels of index k.

In a recursive step of the algorithm, we cut along the sequence of κ levels of index k — i.e., we remove all of the edges that belong to at least one of these levels, even if they also belong to some

[2]Without loss of generality.

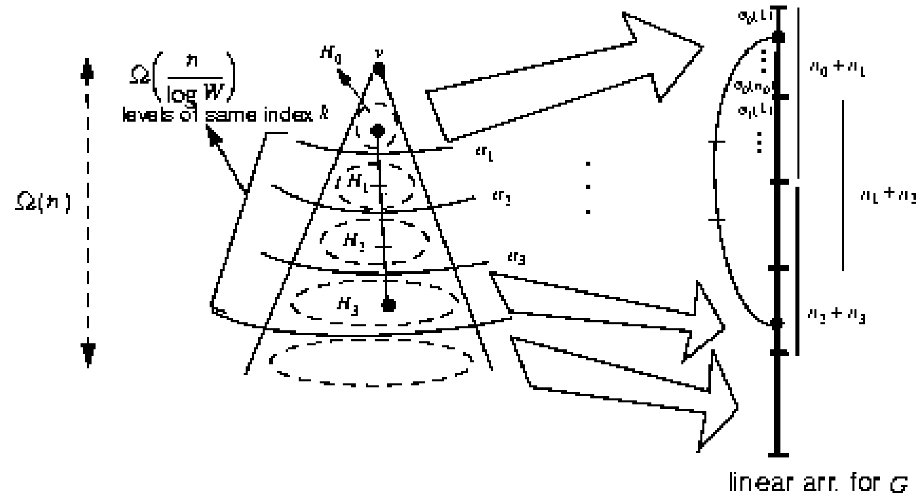

Figure 3: The algorithm and charging scheme.

other levels of index different from k. For all i, let level a_i be the ith level of index k, in increasing order of distances to v. Let H_i be the subgraph induced by the nodes that are at distance greater than a_i and at most a_{i+1} from v; let H_0 (resp., H_κ) be the subgraph induced by the nodes that are at distance at most a_1 (resp., greater than a_κ) from v. Let n_i denote the number of nodes in H_i. We recurse on each H_i, obtaining a linear arrangement σ_i for the n_i nodes in this subgraph. We combine the linear arrangements obtained for the H_i's, obtaining a linear arrangement σ for G, as follows:

$$(\sigma(1),\ldots,\sigma(n)) = (\sigma_0(1),\ldots,\sigma_0(n_0),\sigma_1(1),\ldots,\sigma_1(n_1),\ldots,\sigma_\kappa(1),\ldots,\sigma_\kappa(n_\kappa))$$

Each recursive step runs in polynomial time; at each recursive step, we decompose a connected component into at least two connected components. Hence the algorithm runs in polynomial time.

We use a *charging scheme* to account for the length of an edge e in the linear arrangement for G obtained by our algorithm (note that we will account for the *length of the edge in the linear arrangement*, rather than for the spreading metric length of the edge).

If some edge e in level a_i belongs to some other level of index k, say level a_j, then this edge also belongs to every level of index k between a_i and a_j. W.l.o.g. assume that $i < j$. Edge e will be "stretched over" all the nodes in $H_i \cup \ldots \cup H_{j-1}$, and may be "stretched over" some of the nodes in H_{i-1} and H_j, in the linear arrangement produced by our algorithm. Hence the length of such an edge in the final linear arrangement will be at most $n_{i-1} + \ldots + n_j$. Suppose we charge $n_{p-1} + n_p$ for the portion of the edge that is stretched over the nodes in $H_{p-1} \cup H_p$, when considering level a_p, for all $i \le p \le j$. Then the total charge associated with edge e is equal to $n_{i-1} + 2(n_i + \ldots n_{j-1}) + n_j$ — that is, edge e will be charged at least as much as its length in a final linear arrangement.

Figure 3 illustrates the algorithm and charging scheme described above.

We will now compute an upper bound on the cost of a linear arrangement obtained by our algorithm. Let $C(Z)$ be the maximum cost of a linear arrangement obtained by our algorithm for a subgraph of G whose volume of the spreading metric ℓ is at most Z. Since the sum of the weights of all edges in level a_i is ρ_{a_i}, and since the quantity $w(e)\ell(e)$ for an edge e which belongs to levels $a_i, \ldots, a_j$, $i \leq j$, satisfies $w(e)\ell(e) \geq w(e)(j-i+1)/2$, we have that the sum of the weights of all edges that belong to some level of index k is at least $\sum_{i=1}^{\kappa} \rho_{a_i}/2$. Note that this lower bound on the total sum of the weigths of the edges removed in this cut step is tight: Suppose there are two consecutive levels i and $i+1$ of index k and suppose there is an edge e which belongs to both these levels such that $\ell(e) = 1+\epsilon$, for some arbitrarily small $\epsilon > 0$; the quantity $w(e)\ell(e)$ is equal to $w(e)(1+\epsilon)$ which tends to $w(e)[(i+1)-i+1]/2 = w(e)$ as ϵ tends to zero. We charge for the length of an edge as described in the preceding paragraph, and thus derive the following recurrence relation for $C(W)$:

$$C(W) \leq C(W - \sum_{i=1}^{\kappa} \frac{\rho_{a_i}}{2}) + \sum_{i=1}^{\kappa} [\rho_{a_i}(n_{i-1} + n_i)]$$

We now show that $C(W) = O(W \log n)$. We first prove the following lemma:

Lemma 3.1 *$C(W) \leq cW \log W$, for some constant c.*

Proof: We will use induction on W. The base case $W = 0$ corresponds to a totally disconnected graph (a graph with no edges), and therefore $C(W) = 0$ in this case.

We can use induction on W here since, for any subgraph of G on x nodes whose volume of the spreading metric ℓ is at most Z ($Z \leq W$),

$$\sum_{i=1} \kappa \frac{\rho_{a_i}}{2} \geq \frac{\alpha_k x}{8 \log Z} \geq \frac{\alpha_k x}{8 \log W} \geq \frac{1}{8 \log W}$$

since $\rho_{a_i} > \alpha_k$ and $\kappa \geq x/(4 \log W)$. Thus, the recursive relation above will converge to the base case in at most $8W \log W$ steps.

Combining the recurrence relation for $C(W)$ with $\alpha_k < \rho_{a_i} \leq \alpha_{k+1}$, for all i, we obtain:

$$\begin{aligned}
C(W) &\leq C(W - \frac{\alpha_k n}{8 \log W}) + \alpha_{k+1} \sum_i (n_{i-1} + n_i) \\
&\leq c[W - \frac{\alpha_k n}{8 \log W}] \log[W - \frac{\alpha_k n}{8 \log W}] + 2\alpha_{k+1} n \\
&\leq c[W - \frac{\alpha_k n}{8 \log W}] \log W + 2\alpha_{k+1} n \\
&\leq cW \log W + \alpha_{k+1} n [2 - \frac{c}{16}] \\
&\leq cW \log W
\end{aligned}$$

for a sufficiently large constant c ($c \geq 32$). The second inequality follows from the induction hypothesis; the fourth inequality follows since $\alpha_{k+1} = 2\alpha_k$. ■

We still need to show how to bring the approximation factor down from $O(\log W)$ to $O(\log n)$. We will do this by using standard techniques of rescaling and rounding down the edge weights (as in [4]).

Our goal will be to reduce, by rescaling and rounding down weights, our original input graph G to an "equivalent" input graph G' whose spreading metric volume is a polynomial in n. Consider the set E' of edges e such that $w(e) \leq W/(mn)$. Since an edge has length at most n in any linear arrangement for G, the contribution of the edges in E' to a MLA of G is at most W. Suppose we delete all those edges, and apply a ρ-approximation algorithm to the resulting graph.

We now round down each weight $w(e)$, for all e in $E \setminus E'$, to its nearest multiple of $W/(mn)$. The error incurred by this rounding procedure is again at most W. Furthermore, we scale the rounded weights by $W/(mn)$, obtaining new weights for the edges that are all integers in the interval $[0, mn]$. Note that we have only changed the units in which the weights are expressed.

The volume W' of the spreading metric for solving the MLA problem on $G' = G \setminus E'$ with integral weights that belong to $[0, mn]$ is at most a polynomial on n (since $W' \leq m^2n^2$). By Lemma 3.1, we have $C(W') \leq cW' \log(W') = c'W' \log n$, for some constant c'. Rescaling the edge weights back by multiplying $C(W')$ by $W/(mn)$, we obtain a linear arrangement for the original weights on G' with cost at most $c'W \log n$. Putting back the edges in E' into the linear arrangement obtained for G', we obtain a linear arrangement for G with cost at most $(c' \log n + 1)W$.

Since $W = w(e)\ell(e)$ is a lower bound on the cost of an MLA, by Lemma 3.1 and the considerations above, we have proved the following theorem:

Theorem 1 *The cost of a solution to the MLA problem, obtained by our algorithm is within an $O(\log n)$ factor of the cost of a MLA of G.*

4 Graphs with Excluded Minors

In this section we show how to obtain, in polynomial time, an $O(\log \log n)$-approximation for the MLA problem on a graph G with no fixed $K_{r,r}$-minors — e.g., on a planar graph G. We denote the $r \times r$ complete bipartite graph by $K_{r,r}$.

Definition 4.1 *Let H and G be graphs. Suppose that (i) G contains disjoint connected subgraphs A_v, for each node v of H; and (ii) for every edge (u, v) in H, there is a path $P_{(u,v)}$ in G with endpoints in A_u and A_v, such that any node in $P_{(u,v)}$ other than its endpoints does not belong to any A_w, w in H, nor to any $P_{(i,j)}$, (i, j) in $H \setminus (u, v)$. Then $\cup_v A_v$ is said to be an H-*minor *of G.*

Klein, Plotkin, and Rao [8], showed how to decompose (in polynomial time) an undirected graph with no $K_{r,r}$-minors into connected components of small diameter. In our application, this implies that each connected component has at most a constant fraction of the nodes in G, as shown in the next section.

4.1 The Algorithm.

We recursively solve the problem, as we do in the general case. We combine the partial solutions returned by each recursive step, and charge for each edge removed at a cut step in the same way as in the algorithm of Section 3. It is in the way we decompose the graph before a recursive step that the algorithm of Section 3 differs considerably from the one presented in this section. Before each recursive step, we will perform a series of *shortest path levelings*, to be defined soon, on each induced connected subgraph, until we can guarantee that the original graph has been decomposed into subgraphs that contain at most a fixed fraction (strictly less than one) of the nodes each. In the algorithm of Section 3, we perform only one shortest path leveling before each recursive step.

The algorithm proceeds in rounds. In each round we have a *cut step*, which corresponds to the series of cuts performed during the round, and a *recursive step*, which consists of recursing on the connected components that result from the cut step. Let $G(V, E)$ be a graph on n nodes that excludes $K_{r,r}$ as a minor, for some fixed $r > 0$. Let ℓ be a spreading metric for G of volume W that satisfies (2-3).

A single *cut step* in G will produce a series of subgraphs of G, $G = G_0, \ldots, G_t$, $t \leq r$, where each G_{i+1} results from cutting according to a shortest path leveling of G_i. Fix a node v in G_i. A *shortest path leveling (SPL)* of G_i rooted at v consists of an assignment of levels to the edges of G_i as follows: An edge (x, y) is at *level* j of this SPL if and only if $\mathsf{dist}(v, x)$ (in G_i) is at most j and $\mathsf{dist}(v, y)$ (in G_i) is greater than j, for all $j \in \mathbf{N}$. (An edge may be at more than one level.)

We group the levels of this SPL into *bands* of $2s$ consecutive levels as follows: *Band* i, for $i \in \mathbf{N}$, of the SPL consists of levels $2si$ through $2s(i+1) - 1$, where $s = n/b$, and b is a constant. Let $n(G_i)$ denote the number of nodes in G_i. The spreading metric diameter guarantee implies that this SPL has at least $n(G_i)/4$ levels. We will see later that $n(G_i) = \Theta(n)$, and that we can choose b such that $n(G_i)/4 \geq 2s$ (we need $b \geq 8$). Alternate coloring the bands "blue" and "red", in increasing order of the levels.

We will cut along a sequence of levels of the SPL; one of the connected components resulting from this cutting procedure will be G_{i+1}. W.l.o.g., assume that the subgraph induced by the blue bands has at least $n(G_i)/2$ nodes. We have $2s$ cuts of the following type: For $0 \leq j \leq 2s - 1$, a *leveled cut* j consists of all the edges in the jth level (with respect to distance from v) of every red band. For example, if the band consisting of the first $2s$ levels is colored blue, then the leveled cut j consists of the levels $2s + j, 6s + j, \ldots$, for all j.

We group the leveled cuts according to their indices, which we define below (the definition of an index in this context is slightly different from that in Section 3). We will see that there must exist at least an $\Omega(1/\log\log n)$ fraction of the leveled cuts with the same index k_i. Let $\beta_k = W2^k/(s \log n)$, for all k in $[1, 2\log\log n]$. Let $\beta_0 = 0$ and $\beta_{(2\log\log n)} = W$. The *weight* of leveled cut j is the sum of the weights of the levels in the cut (the weight of a level being the sum of the weights of the edges at that level). Leveled cut j has *index* k, k in $[2\log\log n]$, if and only if the weight of leveled cut j belongs to the interval $I_k = (\beta_k, \beta_{k+1}]$. Thus there must exist at least $s/\log\log n$ leveled cuts in G_i with same index k_i, since there exist at least $2s$ distinct leveled cuts.

If $k_i > 0$, then we cut along these at least $s/\log\log n$ leveled cuts of index k_i, and recurse on the resulting connected components. In this case, we let $t = i$, the cut step of this round is complete, and we recurse.

Otherwise, we first cut along only one of the leveled cuts of index $k_i = 0$ (chosen arbitrarily). Then we check whether there exists a resulting connected component G_{i+1} of G_i with at least $n(G_i)/2$ nodes. In case no such component exists, we let $t = i$ (the cut step of this round is complete), and we recurse. If $i = r - 1$, we also let $t = i$ (completing the cut step of this round) and recurse. If a component G_{i+1} with at least $n(G_i)/2$ nodes exist and if $i < r-1$, we proceed by performing a SPL on G_{i+1}, and by finding a suitable leveled cut(s) in G_{i+1} following the procedure just described for $i = i + 1$. If $i = r - 1$, we let $t = i$.

The number of nodes in G_i, $n(G_i)$, is proportional to n, for all i in $[t]$. This follows since $n(G_{i+1}) \geq n(G_i)/2$, by the choice of G_{i+1}, and since t $(\leq r)$ is a constant.

Suppose we just performed a series of t SPL's and corresponding cut procedures. The last cut performed, on G_{t-1}, generated a collection of connected components of G_{t-1}. Klein, Plotkin, and Rao [8] proved that the distance in G between any pair of nodes in any such component is $O(s)$ (where the constant in the $O(\cdot)$ notation depends only on r). Thus, for a large enough constant b, $b \geq 8$, we can ensure that the distance between any pair of nodes is at most $n/6$, in any such component.

We now show that it follows from the result by Klein, Plotkin, and Rao that any connected component that results from this cut step has at most $2n/3$ nodes. Fix any node u in G. It follows from (2), that any subgraph of G on $(n - x)$ nodes that contains u has a node at distance at least $(n - x)/4$ from u. Suppose we start with the graph G, and proceed by removing one node at a time, choosing always a node that has maximum distance to u among the remaining nodes. Thus, we need to remove at least one-third of the nodes before we are left only with nodes that are within distance $n/6$ from u in G. This implies that any resulting connected component of G_{t-1} has at most $2n/3$ nodes. Any other resulting connected component (of $G \setminus G_{t-1}$) has at most $n/2$ nodes, by the choice of the G_i's.

We distinguish between two types of cut steps: If k_{t-1} is equal to 0, then we have a cut step of *type I* in this round; otherwise k_{t-1} is not zero, and the cut step in this round is of *type II*. Note that in either type of cut step, $k_j = 0$, for all j in $[t-1]$.

Let $C(Z, x)$ denote the maximum cost of a linear arrangement obtained by our algorithm for a subgraph of G with x nodes, whose volume of the spreading metric ℓ is at most Z.

Lemma 4.1 *$C(W, n) \leq cW \log\log n$, for some constant c.*

Proof: We use induction on W and n (a similar argument to that in Lemma 3.1 shows that induction converges to the base case here in a finite number of steps). The base cases for $W = 0$ or $n = 0$ are trivial. Suppose we perform a cut step of type I. Thus, we perform a sequence of at most r cuts along a single leveled cut of each SPL in this step, and each of these leveled cuts has weight at most $2W/(s\log n)$. Let the connected components resulting from this cut step be $H_0, \ldots, H_p$. Then

$$\begin{aligned} C(W,n) &\leq \sum_{i=0}^{p} C(W_i, n_i) + r\frac{2W}{s\log n}n \\ &\leq \sum_i cW_i \log\log(2n/3) + \frac{2brW}{\log n} \\ &\leq cW\log\log n - \frac{cW}{3\log n} + \frac{2brW}{\log n} \\ &\leq cW\log\log n \end{aligned}$$

where W_i and n_i are the volume and number of nodes, respectively, associated with component H_i. We have shown that every n_i is at most $2n/3$. The second inequality above follows by induction. Note that $\log\log(2n/3) \leq \log\log n - 1/(3\log n)$: Thus the third inequality follows. The last inequality follows for a sufficiently large constant c ($c \geq 6br$).

If the cut step performed was of type II, then we performed a series of $t \leq r$ SPL's and respective cut procedures. The last term on the right-hand side of the first inequality below accounts for the first $(t-1)$th leveled cuts of index $k_i = 0$ used, one for each G_i, $0 \leq i < t-1$. The second term on the right-hand side of that inequality accounts for the tth group of leveled cuts of index $k_{t-1} > 0$ used for G_{t-1}. The charging scheme for the edges removed in the tth set of leveled cuts used in this cut step, and the lower bound on how much those edges contributed to the spreading metric volume W are analogous to the ones used in Section 3.

$$\begin{aligned} C(W,n) &\leq C(W - \frac{\beta_k}{2}\frac{s}{\log\log n}, n) + 2\beta_{k+1}n \\ &\quad +(r-1)\frac{2W}{s\log n}n \\ &\leq C(W - \frac{\beta_k s}{2\log\log n}, n) + (r+3)\beta_k n \\ &\leq c(W - \frac{\beta_k s}{2\log\log n})\log\log n + (r+3)\beta_k n \end{aligned}$$

$$
\begin{aligned}
&\leq cW \log\log n + \beta_k n(r+3-\frac{c}{2b}) \\
&\leq cW \log\log n
\end{aligned}
$$

when $c \geq 2b(r+3)$. The second inequality above follows from $\beta_k \geq 2W/(s \log n)$, and from $\beta_{k+1} = 2\beta_k$, $0 < k < 2\log\log n - 1$; the fourth inequality follows since $s = n/b$.

■

Since the volume W of the spreading metric ℓ is a lower bound on the cost of an MLA, by Lemma 4.1, we obtain the following theorem:

Theorem 2 *Given a graph G on n nodes that excludes fixed $K_{r,r}$-minors, the cost of a solution to the MLA problem obtained by the algorithm presented in this section is within an $O(\log\log n)$ factor of the cost of a MLA of G.*

5 Storage–Time Product

In this section, we sketch our approach for approximating the minimum storage–time product problem on a directed acyclic graph $G(V, E)$.

The minimum storage–time product problem arises in a manufacturing or computational process, in which the goal is to minimize the total storage–time product of the process: We want to minimize the use of storage over time, assuming storage is an expensive resource. Let $G(V, E)$ be an acyclic directed graph on n nodes with edge weights $w(e)$, for all $e \in E$, and node weights $\tau(v)$, for all $v \in V$. The nodes of G represent tasks to be scheduled on a single processor. The time required to process task v is given by $\tau(v)$. The weight on edge (u, v), $w(u, v)$, represents the number of units of storage required to save intermediate results generated by task u until they are consumed at task v. The *minimum storage–time product problem* consists of finding a topological ordering[3] of the nodes $\sigma : V \to \{1, \ldots, n\}$ that minimizes

$$
\sum_{(i,j)\in E, \sigma(i)<\sigma(j)} \left\{ w(i,j) \left[\sum_{k \,:\, \sigma(i)\leq\sigma(k)\leq\sigma(j)} \tau(\sigma(k)) \right] \right\}.
$$

Figure 4 illustrates a topological ordering of the nodes of G (given from left to right on the rightmost representation of the graph) with minimum storage–time product of 177.

This problem generalizes the MLA problem: When all tasks have unit execution time, it becomes a directed version of the MLA problem. It is also a generalization of the single-processor scheduling problem, if we are minimizing the weighted sum of completion times (this problem is NP-complete [5, problem SS13, page 240]).

For the storage–time product problem, we use a spreading metric ℓ defined as follows. Let $E^R = \{(u, v) | (v, u) \in E\}$. We define $G' = (V, E \cup E^R)$. Let $\mathcal{V}$ denote the set of all nontrivial strongly connected subgraphs of G'. The spreading metric ℓ is a solution to

[3] An ordering σ of the nodes of G (where G is an acyclic directed graph) is said to be *topological* if and only if for every $(i, j) \in E$, $\sigma(i) < \sigma(j)$.

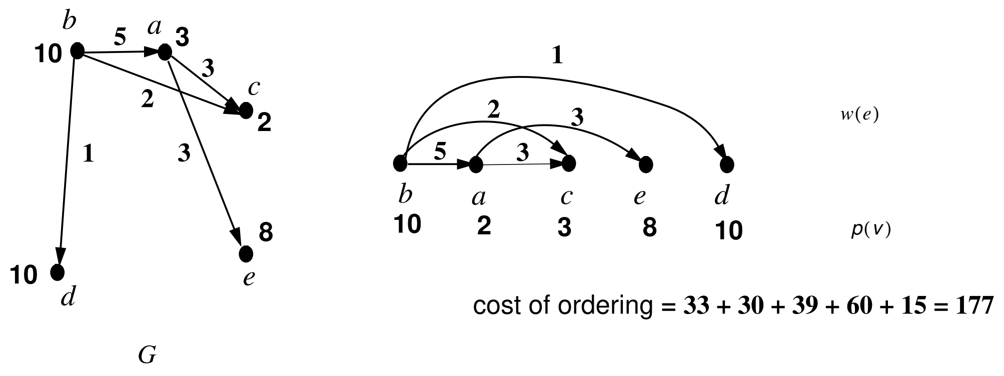

Figure 4: A minimum storage–time product of G.

$$\begin{aligned} W = \quad & \min w(e)\ell'(e) \\ \text{s.t.} \quad & \frac{\sum_{u\in U}\tau(u)\delta(u,v)}{|U|} \geq \frac{\sum_{u\in U}\tau(u)}{4}, \qquad \forall U \in \mathcal{V}, \forall v \in U \\ & \ell'(i,j) \geq \tau(i) + \tau(j), \qquad \forall (i,j) \in E \\ & \ell'(i,j) = 0, \qquad \forall (i,j) \in E^R \end{aligned}$$

where $\delta(u,v)$ is equal to $(\mathsf{dist}(u,v) + \mathsf{dist}(v,u))$. Here we define $\mathsf{dist}(u,v)$ to be the length of a shortest path from u to v in G' according to the lengths $\ell'(e)$ for $e \in E \cup E^R$. (Note that each $e \in E^R$ has length zero.)

For any linear ordering σ of V, the assignment of lengths to the edges given by $\ell'(i,j) = \sum_{k\,:\,\sigma(i)\leq\sigma(k)\leq\sigma(j)} \tau(\sigma(k))$, for all (i,j) in E, satisfies the constraints above. Thus the volume W of the spreading metric ℓ is a lower bound on the optimal cost of a solution to the storage–time product problem.

We can adapt the algorithm of Section 3 to this problem as follows. Let $T = \sum_{v\in V}\tau(v)$. There is a node v such that the out-tree, or the in-tree rooted at v has depth $\Omega(T)$. Thus, we can find a sequence $a_1, \ldots a_\kappa$ of $\kappa = \Omega(T/\log W)$ levels whose weights $\rho_{a_1}, \ldots \rho_{a_\kappa}$ are within a factor of two of each other (as in Section 3).

Laying out the resulting pieces successively, we obtain a solution where the cost is bounded by

$$C(W) \leq C(W - \sum_{i=1}^{\kappa}\frac{\rho_{a_i}}{2}) + \sum_{i=1}^{\kappa}[\rho_{a_i}(\tau_{i-1} + \tau_i)].$$

where τ_i is the sum of $\tau(v)$ over all nodes v that lie between levels a_{i-1} and a_i (τ_0 and τ_κ are defined accordingly).

This recursion can be upper bounded by $O(W \log W)$, as in Section 3. This cost can be reduced to $O(W \log T)$ using the standard techniques that were used in Section 3 to reduce $O(\log W)$ to $O(\log n)$.

Since the minimum storage–time problem is defined on a directed graph G, the techniques from Klein, Plotkin, and Rao [8] cannot be applied to this problem in order to get a better approximation factor for graphs with excluded $K_{r,r}$-minors.

6 Minimum containing interval graph

In this section, we sketch our approach to approximating the cost of a minimum containing interval graph of a graph $G(V, E)$.

We first introduce interval graphs. An *interval graph* is a graph whose nodes can be mapped to distinct intervals in the real line such that two nodes in the graph have an edge between them if and only if their corresponding intervals overlap. A completion of a graph G into an interval graph results in an interval graph with same node set as G that contains G as a subgraph.

We use the following characterization of interval graphs, due to [12]. An undirected graph $G(V, E)$ on n nodes is an interval graph if and only if there exists a linear ordering $\sigma : V \to \{1, \ldots, n\}$ of the nodes in V such that if an edge $(u, v) \in E$, where $\sigma(u) < \sigma(v)$, then every edge (u, w), for w such that $\sigma(u) < \sigma(w) < \sigma(v)$, also belongs to E. This characterization implies that, for any given ordering σ of the nodes in G, there exists a *unique* way of completing G into an interval graph $\overline{G}$ by adding as few edges to G as possible.

The *minimum containing interval graph problem* seeks for a completion of a graph G into an interval graph $\overline{G}$ with *minimum total number of edges.* Thus, in this problem, the cost of a completed graph is given by the *total* number of edges in the graph. This cost can be viewed as the sum over nodes of the maximum backward stretch of the node — i.e., of the distance to the farthest lower-numbered node to which the node is connected. This is very similar to the MLA problem, except that the nodes are stretched along the order rather than the edges (see [2]). Thus, our techniques in Sections 3 and 4 also apply to this problem.

The containing interval graph problem arises in several areas, from computer science, to biology (see [11]), to archaeology (e.g., when finding a consistent chronological model for tool use while making as few assumptions as possible [7]).

The spreading metric ℓ that we use (due to [2]) assigns lengths to the nodes of the graph, rather than to its edges, as in the minimum linear arrangement and in the minimum storage–time product problems. Let $\mathcal{V}$ denote the set of all nontrivial connected subgraphs of G. The metric ℓ is a solution to

$$\begin{array}{rll} W = & (\sum_{v \in V} \ell'(v))/2 & \\ \text{s.t.} & \sum_{v \in U} \mathsf{dist}(u, v) \geq \frac{1}{4}(|U|^2 - 1), & \forall u \in U,\ \forall U \in \mathcal{V} \\ & \ell'(v) \geq 0, & \forall v \in V \end{array}$$

where $\mathsf{dist}(u, v)$ is the shortest length — given by $[\ell'(u) + \sum_{i=0}^{p} \ell'(x_i) + \ell'(v)]$ — of a path $u, x_0, \ldots, x_p, v$, $x_i \in V$, from u to v in G.

Let $\overline{G}(V, \overline{E})$ be a completion of G into an interval graph. If we let $\ell'(v)$ be the degree of node v in $\overline{G}$, the cost $(\sum_{v \in V} \ell'(v))/2$ clearly gives the number of edges in $\overline{E}$. Also this assignment of lengths to the nodes satisfies the constraints above. Hence the volume W of the metric ℓ is a lower bound on the number of edges in a minimum containing interval graph of G.

The recurrence relations that bound the cost of a solution obtained for the minimum containing interval graph problem are analogous to the ones for the MLA problem, both for the general case and for the excluded $K_{r,r}$-minors case.

A closely related problem to the minimum containing interval graph problem is that of the *minimum interval graph completion problem.* In this problem, the cost of a completion of a graph G into an interval graph $\overline{G}$ is given by the number of edges in $\overline{G} - G$. While an optimal solution to the minimum containing interval graph problem on a graph G is also an optimal solution to the minimum interval graph completion problem on G, an α-approximate solution $\overline{G}$ to the minimum containing interval graph completion problem on G is not enough to guarantee the existence of solution to the minimum interval graph completion problem on G with an α-approximation factor.

7 Conclusion

We provided an existentially tight bound on the relationship between the spreading metric cost volumes and the true optimal values for the problems of minimum linear arrangement, minimum containing interval graph, and minimum storage–time product.

It would be interesting to extend our techniques to obtain $O(\log n)$-approximation algorithms for other problems. In particular, it seems natural to extend our techniques to improve the best known approximation factors for other problems that satisfy the "approximation paradigm" of [2]. We would then provide an existentially tight bound — on the ratio between the value of an optimal solution and the spreading metric volume — for any such problem.

Another interesting problem would be to extend the ideas used for approximating the minimum containing interval graph problem to obtain better approximations for the minimum containing chordal graph problem (interval graphs are a particular subclass of chordal graphs). The minimum containing chordal graph problem has not been addressed in [2].

However, since the approach used here depends on the structure of linear-ordering graph problems, new ideas may be required.

References

[1] C. F. Bornstein and S. Vempala, *Flow metrics*, in Proceedings of LATIN 2002: Theoretical Informatics, 5th Latin American Symposium, vol. 2286 of Lecture Notes in Computer Science, Springer, 2002, pp. 516–527.

[2] G. Even, J. Naor, S. Rao, and B. Schieber, *Divide-and-conquer approximation algorithms via spreading metrics*, in Proceedings of the Thirty-Sixth Annual Symposium on Foundations of Computer Science, Oct. 1995, pp. 62–71.

[3] ——, *Spreading metric based approximate graph partitioning algorithms*, in Proceedings of the Eigth Annual ACM–SIAM Symposium on Discrete Algorithms, Jan. 1997, pp. 639–648.

[4] G. Even, J. Naor, B. Schieber, and M. Sudan, *Approximating minimum feedback sets and multicuts in directed graphs*, in Integer programming and combinatorial optimization-Lecture Notes in Computer Science, E. Balas and J. Clausen, eds., no. 920, Springer–Verlag, New York, 1995, pp. 14–28. Full version appears in IBM Research Report RC 20074 (88796).

[5] M. R. Garey and D. S. Johnson, in Computers and Intractability: A guide to the theory of NP-Completeness, W. H. Freeman and Company, New York, NY, 1979.

[6] M. Hansen, *Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems*, in Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science, Oct. 1989, pp. 604–609.

[7] D. G. Kendall, *Incidence matrices, interval graphs, and seriation in archeology*, Pacific J. Math., 28 (1969), pp. 565–570.

[8] P. Klein, S. Plotkin, and S. Rao, *Excluded minors, network decomposition and multicommodity flow*, in Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing, Oct. 1993, pp. 682–690.

[9] R. Koch, T. Leighton, B. Maggs, S. Rao, and A. Rosenberg, *Work-preserving emulations of fixed-connection networks*, in Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, May 1989, pp. 227–240.

[10] F. T. Leighton and S. Rao, *An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms*, in Proceedings of the Twenty-Ninth Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Oct. 1988, pp. 422–431.

[11] J. Meidanis and J. C. Setubal, *Introduction to Computational Molecular Biology*, PWS Publishing Co., Boston, MA, 1997.

[12] G. Ramalingam and C. P. Rangan, *A unified approach to domination problems in interval graphs*, Information Processing Letters, 27 (1988), pp. 271–174.

[13] S. Rao and A. W. Richa, *New aprroximation techiniques for some ordering problems*, in Proceedings of the Ninth Annual ACM–SIAM Symposium on Discrete Algorithms, Jan. 1998, pp. 211–218.

[14] R. Ravi, A. Agrawal, and P. Klein, *Ordering problems approximated: Single processor scheduling and interval graph completion*, in Proceedings of the Eighteenth International Colloquium on Automata, Languages and Programming, July 1991, pp. 751–762.

[15] P. D. Seymour, *Packing directed circuits fractionally*, Combinatorica, 15 (1995), pp. 281–288.