Divide-and-Conquer Approximation Algorithms via Spreading Metrics[*]

Guy Even[†] Joseph (Seffi) Naor[‡] Satish Rao[§] Baruch Schieber[¶]

Abstract

We present a novel divide-and-conquer paradigm for approximating NP-hard graph optimization problems. The paradigm models graph optimization problems that satisfy two properties: First, a divide-and-conquer approach is applicable. Second, a fractional spreading metric is computable in polynomial time. The spreading metric assigns rational lengths to either edges or vertices of the input graph, such that all subgraphs on which the optimization problem is non-trivial have large diameters. In addition, the spreading metric provides a lower bound, τ, on the cost of solving the optimization problem. We present a polynomial time approximation algorithm for problems modeled by our paradigm whose approximation factor is $O(\min\{\log\tau\log\log\tau, \log k\log\log k\})$, where k denotes the number of "interesting" vertices in the problem instance, and is at most the number of vertices.

We present seven problems that can be formulated to fit the paradigm. For all these problems our algorithm improves previous results. The problems are: (1) linear arrangement; (2) embedding a graph in a d-dimensional mesh; (3) interval graph completion; (4) minimizing storage-time product; (5) subset feedback sets in directed graphs and multicuts in circular networks; (6) symmetric multicuts in directed networks; (7) multiway separators and ρ-separators (for small values of ρ) in directed graphs.

---

[*] A preliminary version of this paper appeared in the Proceedings of the 36th Symposium on Foundations of Computer Science, 1995, pp. 62-71.

[†] Univ. des Saarlandes, FB 14 Informatik, LS Prof. Paul, 66123 Saarbrücken, Germany. E-mail:`guy@cs.uni-sb.de`

[‡] Computer Science Dept., Technion, Haifa 32000, Israel. E-mail:`naor@cs.technion.ac.il`. Research supported in part by Grant No. 92-00225 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel.

[§] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540. E-mail: `satish@research.nj.nec.com`

[¶] IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598. E-mail:`sbar@watson.ibm.com`.

1 Introduction

In this paper we describe efficient divide and conquer techniques that yield improved approximation algorithms for a number of NP-Hard graph optimization problems. Our methods rely on a class of functions on graphs that we call *spreading metrics*. Informally, a spreading metric on a graph is an assignment of lengths to either its edges or vertices, so that subgraphs for which the optimization problem is non-trivial are spread apart in the associated metric space. The *volume* of a spreading metric on a graph is defined as the sum, taken over all edges (vertices), of the length of the edges (vertices) multiplied by their capacity. The volume of the spreading metric provides a lower bound on the cost of solving the optimization problem on the input graph. We use the spreading metric to find a cut in the graph whose cost depends on the volume of the spreading metric in one of the resulting subgraphs. Thus, we use a divide-and-conquer approach that divides the problem according to the *cost* of solving the optimization problem, rather than traditional methods that divide according to the *sizes* of the subproblems.

A b-balanced cut in a graph $G = (V, E)$ is a cut that partitions the graph into connected components (strongly connected components in the directed case), each of which contains at most $(1 - b) \cdot n$ vertices, where $n = |V|$. Leighton and Rao [LR88] presented an algorithm that finds a b-balanced cut, for any $b \leq 1/3$, whose capacity is $O(\log n)$ times the minimum capacity of a *bisector* (a 1/2-balanced cut). Often, when the cost of a minimum capacity b-balanced cut can be used to lower bound the optimal cost of the problem at hand, one can construct a log-square approximation factor algorithm by recursively dividing the problems using b-balanced cuts for some constant b. One logarithmic error term is due to the error in the separator procedure, and the other comes from the recursion depth. On the face of it, it seemed that the only way to "break" the log-square barrier in the framework described herein, is by improving the separator approximation.

Our methods allow us to break the log-square barrier by an almost logarithmic factor, without finding better separators than [LR88]. The idea is that spreading metrics can be used to give an estimate of the recursive cost of solving subproblems that arise in the divide-and-conquer methods. Thus, we can manipulate the cost of the cut to be higher when the recursion makes a lot of progress, and lower when it does not. In fact, we can do this to such an extent that we only pay a single logarithmic factor (along with a doubly logarithmic factor) for both the recursion and the error in the cut procedure. This generalizes the approach taken in by Seymour [Se95], in which the existence of small feedback sets in directed unweighted graphs is proved.

Recently, in [ENRS95], we were able to extend our spreading metrics technique to graph partitioning problems. We were able to apply simpler recursion and obtain simple algorithms and proofs for balanced cuts and multiway separators in which the approximation factor is logarithmic. However, the simpler recursion does not apply for the applications described in this paper.

1.1 Results

We consider problems that are amenable to a divide-and-conquer approach and where a spreading metric is computable in polynomial time. There are two main parameters that are attached to each problem instance. The first one is τ, the volume of the spreading metric on the graph which lower bounds the cost of the optimal solution and is usually obtained through a solution of a linear program; the second parameter is k, the number of "interesting" vertices, as defined by the problem instance, and is at most the number of vertices. We present a polynomial time approximation algorithm for several problems where the approximation factor is $O\left(\min\{\log \tau \log\log \tau, \log k \log\log k\}\right)$.

We demonstrate the applicability of this approach by describing applications of our methods to seven problems that fit our paradigm. For all of these problems, our approximation algorithm improve previous results. These problems are: (1) linear arrangement; (2) embedding a graph in a d-dimensional mesh;[1] (3) interval graph completion; (4) minimizing storage-time product; (5) (subset) feedback sets in directed graphs and multicuts in circular networks; (6) symmetric multicuts in directed networks. (7) k-multiway separators and ρ-separators in directed graphs. For the first four problems we improve the approximation factor from $O(\log^2 n)$ to $O(\log n \log\log n)$, where n denotes the number of vertices. (For these problems, $\tau \geq n$ and $k = n$.) For problems (5) and (6), we improve the approximation factor from $O\left(\min\{\log \tau \log\log \tau, \log n \log\log n, \log^2 k\right)$ to $O\left(\min\{\log \tau \log\log \tau, \log k \log\log k\}\right)$, where k denotes the size of the subset in the subset feedback set problems, and the number of source-sink pairs in the multicut problems. For Problem (7) our approximation algorithm finds a separator whose capacity is $O\left(\min\{\log n \log\log n, \log \tau' \log\log \tau'\} \cdot \tau'\right)$, where τ' denotes the optimal cost of a ν-separator, and $\nu = 1/k$ in the multiway separator case, or $\nu = \rho - \varepsilon$, for some fixed ε, in the ρ-separator case. As will be shown later, this improves on previous results, either when k is big enough, or when ρ is small enough.

Some optimization problems require balanced partitioning for applying a divide-and-conquer approach. However, the cuts we find are not guaranteed to be balanced. We present an additional technique for balancing the decomposition found by recursively finding cuts.

1.2 Comparison to prior work

Leighton and Rao [LR88] presented an $O(\log n)$ approximation algorithm for balanced partitions of graphs. Among other applications, this provided a basis for an algorithmic implementation of the decomposition tree framework of [BL84]. Other applications described by [LR88] are: an $O(\log^2 n)$ approximation algorithm for the minimum feedback edge set in directed graphs and an $O(\log^2 n)$ approximation algorithm for the minimum cut linear arrangement problem.

The problems of linear arrangement and graph embeddings in d-dimensional meshes were considered by Hansen [Ha89]. His algorithms rely on the algorithm of Leighton and Rao [LR88] to obtain $O(\log^2 n)$ approximation algorithms for these problems. Our algorithms improve the approximation factor to $O(\log n \log\log n)$. Ravi *et al.* [RAK91] considered a generalization of the linear arrangement problem, called the storage-time product minimization problem. In case the execution times of all tasks is the same their algorithm achieves an $O(\log^2 n)$ approximation factor. Again, our algorithm improves this factor to $O(\log n \log\log n)$.

---

[1] In fact, linear arrangement is a special case of embedding a graph in a d-dimensional mesh. We separate the problems for simplicity of exposition, since solving the more general problem requires extra ideas.

The problem of finding a super-graph of a given graph, such that the super-graph is an interval graph and contains as few as possible edges, was considered by Ravi *et al.* [RAK91]. They extended Hansen's technique for this problem and obtained an $O(\log^2 n)$ approximation factor. We present a novel linear programming formulation of this problem that enables us to use our paradigm to obtain an $O(\log n \log\log n)$ approximation factor.

Decompositions of directed graphs were considered in the works of [LR88, Se95, KPRT93, ENSS95]. Seymour [Se95] was the first to present a decomposition algorithm that does not rely on balanced cuts. His paper proves the existence of feedback vertex sets in unweighted directed graphs of cardinality $O(\tau \log \tau \log\log \tau)$. Klein *et al.* [KPRT93] considered symmetric multicuts in directed graphs. By extending the work of Garg *et al.* [GVY93] to the directed case, they obtained an $O(\log^2 k)$ approximation factor for this problem. Even *et al.* [ENSS95] considered feedback set problems and their generalizations in directed graphs, as well as multicuts in circular networks. They presented an $O(\min\{\log \tau \log\log \tau, \log n \log\log n, \log^2 k\})$ approximation factor for the subset feedback set problem, where k denotes the number of special vertices. The first two terms in the approximation factor were obtained by extending the work of Seymour [Se95], and the last term was obtained by extending the work of [GVY93]. We use our paradigm to obtain an $O(\min\{\log k \log\log k, \log \tau \log\log \tau\})$ approximation factor for all of these problems – an improvement over previous works for small values of k.

The ρ-separator problem in directed graphs is to find a minimum capacity subset of edges whose removal partitions the graph into strongly connected components each of which contains at most $\rho \cdot n$ vertices, where n is the total number of vertices. This problem for both the undirected and the directed cases was introduced in [ENRS95], where a logarithmic approximation factor algorithm is described for the undirected case and also for the directed case when $\rho \geq 1/2$. The algorithm presented here applies to the directed case for the range $\rho < 1/2$. It finds a separator whose capacity is $O(\min\{\log n \log\log n, \log \tau' \log\log \tau'\} \cdot \tau')$, where τ' denotes the minimum cost of a ν-separator, for some fixed $\nu < \rho$. Another problem that we consider is that of computing k-multiway separators in directed graphs. Here, we are interested in a minimum capacity subset of edges whose removal partitions the graph into strongly connected components that can be grouped into k parts of roughly equal size. We apply our ρ-separators algorithm to this problem and achieve an algorithm with the same approximation factor, as detailed in Section 6. Note that one can find a 2ν-separator (and hence, also a k-multiway separator) by applying recursively the approximate separator algorithm in [LR88] until all strongly connected components are small enough. (See also, [LMT90, ST95].) This approach yields a 2ν-separator whose capacity is $O(\log n \log(1/\nu)\tau')$. Hence, our ρ-separator algorithm is superior to the recursive one for $\rho < \max\{1/\log n, \tau'^{-\log\log\tau'/\log n}\}$. (A similar improvement is achieved for the k-multiway separator problem as detailed in Section 6.)

The recent papers of Linial *et al.* [LLR95] and Aumann and Rabani [AR94] also consider metrics on graphs. They regard graphs with edge lengths as geometric objects and map them into Euclidean spaces with a logarithmic distortion of the edge lengths. The dimension of the Euclidean spaces they map the graphs into is poly-logarithmic, and the "geometry" of the graph is then utilized for approximating multi-cuts. When we embed graphs into a Euclidean space (e.g. d-dimensional meshes), we consider only constant dimensionality. The main property we utilize for finding cuts is that the diameter of every subgraph, that corresponds to a non-trivial sub-problem, is sufficiently large.

The rest of the paper is organized as follows. Section 2 describes the approximation paradigm and states its performance. Section 3 gives an algorithm for partitioning a large diameter graph. This partitioning algorithm is used in the divide step of the divide-and-conquer algorithm described in Section 4. Section 5 extends the divide-and-conquer algorithm to problems where the divide step has to partition the graph into balanced parts. Finally, Section 6 describes the applications of the algorithm.

2 The approximation paradigm

In this section we describe our approximation paradigm and state its performance. First, we describe the "edge" version of our paradigm, and then we show how to modify it to get the "vertex" version.

The paradigm includes minimization problems on graphs (directed and undirected) with edge capacities. A problem instance, Π, is denoted by a pair of graphs (G, H), where $G = (V, E)$, is a graph with edge capacities $c(e) \geq 1$, for every edge $e \in E$. The graph $H = (V, E_H)$, called the *auxiliary graph*, is defined on the same vertex set. Our paradigm does not define solutions of the problems nor does it specify the cost of a solution. Instead, we demand that if the edge set E_H is empty, then the cost of problem Π on graph G, denoted by $\mathsf{cost}_{\Pi}(G)$, is zero. In addition, we require that the following two properties are satisfied.

Divide-and-conquer applicability. Problem Π can be solved by using a divide-and-conquer approach that partitions the problem into subproblems by either removing edges or removing vertices. Consider the edge version, and let $F \subseteq E$ denote a subset of edges (usually, we will use edge cuts). Define the capacity $c(F)$ to be the sum of the capacities of the edges in F. Suppose that removing the edges in E disconnects the graph into p connected components. Let $V_1, V_2, \ldots, V_p$ denote the p connected components of $G' = (V, E - F)$. Consider the subproblems corresponding to each connected component of G': let G_i denote the subgraph of G induced by V_i, and let H_i denote the subgraph of H induced by V_i.

The applicability of divide-and-conquer means that the cost of solving problem Π by partitioning it into subproblems satisfies the following recurrence inequality.

$$\mathsf{cost}(G, H) \leq \begin{cases} 0 & \text{if } E_H = \emptyset. \\ \sum_{i=1}^{p} \mathsf{cost}(G_i, H_i) + scaler(H) \cdot c(F) & \text{otherwise} \end{cases}$$

The coefficient $\mathsf{scaler}(H)$ is a cost-scaler of the contribution of $c(F)$ to the solution, and it depends only on the auxiliary graph H. We assume that whenever H contains edges, then $\mathsf{scaler}(H) \geq 1$. In all the applications discussed herein, the cost scaler is a function of the size of the largest connected component in H.

Spreading metric computability. A spreading metric is a function $\ell : E \rightarrow Q$ that assigns rational lengths to every edge $e \in E$. It has to be computed in polynomial time, and needs to satisfy two properties:

Lower-bound: The volume of the spreading metric, defined by $\sum_{e\in E} c(e)\ell(e)$, is a lower bound on $\mathsf{cost}(G, H)$.

Diameter guarantee: The distance induced by the spreading metric "spreads" the graph and all its subgraphs that correspond to non-trivial subproblems. More precisely, for vertices u and v, let $\mathsf{dist}(u, v)$ denote the length of the shortest path from u to v, where distances are measured with respect to edge lengths $\ell(e)$. Let $U \subseteq V$ be any subset of the vertex set. Denote by G_U and H_U the subgraphs of G and H induced by U, respectively. The diameter guarantee implies that for every subset $U \subseteq V$, if H_U has at least one edge, then there exist two vertices $u, v \in U$ that are connected in H_U, for which $\mathsf{dist}(u, v) \geq \mathsf{scaler}(H_U)$.

The main result presented in this paper is the following theorem.

Theorem 1: For any optimization problem Π that satisfies the above paradigm, it is possible to find an approximation algorithm with approximation factor $O\left(\min\{\log \tau \log\log \tau, \log k \log\log k\}\right)$, where τ is the value of an optimal (fractional) solution, and $k \leq n$ is the number of vertices that are not isolated in H.

Remarks:

1. For some problems an additional *balance-constraint* is required for applying a divide-and-conquer paradigm. The balance-constraint requires that all the values $\{\mathsf{scaler}(H_i)\}_{i=1}^{p}$ are upper bounded by a constant fraction of $\mathsf{scaler}(H)$. In applications where the auxiliary graph H is a complete graph and the scaler function depends only on the cardinality of the vertex-set in the auxiliary graph, then the balance constraint translates to the constraint that the cut used is a b-balanced cut, for some $b > 1/2$.

2. If H_U contains at least one edge, then the diameter guarantee together with the assumptions that $c(e) \geq 1$, and that the cost scaler is at least one, imply that the volume $\sum_{e\in E_U} c(e)\ell(e) \geq 1$.

3. In the "vertex" version of this paradigm: (i) the capacity function $c(v)$ is associated with each vertex $v \in V$, rather than each edge, (ii) the problem is partitioned in the divide step by removing vertices rather than edges, and (iii) the spreading metric assigns labels to the vertices.

4. We can relax the diameter guarantee by only requiring $\mathsf{dist}(u, v) \geq \alpha \mathsf{scaler}(H_U)$, where α is some parameter (constant or non-constant). For constant α, the resulting approximation factor is multiplied by a factor of $1/\alpha$.

3 Partitioning high diameter graphs

The main step in our divide-and-conquer algorithm is the divide step. This step is done by applying the cut procedure described in this section. Every cut partitions the graph into two parts. We would like to charge the capacity of the cut to the volume of the part whose volume is smaller. The cut procedure is based on a lemma that guarantees the existence of such a cut, provided that the diameter of the graph is large enough. Moreover, the lemma proves the existence of such a cut among a set of cuts that are defined by the layers of a shortest path tree rooted at a given vertex. Our definitions and techniques apply both to directed and undirected graphs.

Let $G = (V, E)$ denote a (connected) graph with edge capacities $c(e) \geq 1$ for all $e \in E$. Let $H = (V, E_H)$ denote the auxiliary graph corresponding to G, and let $\ell(e)$ denote the non-negative rational edge lengths obtained from a spreading metric computed on the original graph. (Note that we perform this procedure recursively, and thus G is a subgraph of the original graph.) Let $u, v \in V$ be two non-isolated vertices in the auxiliary graph H, for which the diameter property guarantees $\Delta \triangleq \mathsf{dist}(u,v) \geq \mathsf{scaler}(H)$. Denote the r-region of u under the metric $\ell(\cdot)$ by $N(u,r)$. Namely, $N(u,r)$ is the set of all vertices whose distance from u is at most r. Denote by $E(u,r)$ the set of edges for which both endpoints belong to $N(u,r)$, and by $C(u,r)$ the set of edges belonging to the cut $(N(u,r), V - N(u,r))$. Define $\mathsf{vol}(u,r)$, the *volume* of $N(u,r)$, to be:

$$\mathsf{vol}(u,r) \triangleq \sum_{e \in E(u,r)} c(e)\ell(e) \; + \sum_{e=(x,y) \in C(u,r)} c(e)(r - \mathsf{dist}(u,x))$$

Define $\mathsf{vol}_{rev}(v,r)$ to be the volume of the r-region of v in the reversed graph (in the undirected case the reversed graph is the same as the original graph). Let vol^* be the volume of the whole graph, that is, $\mathsf{vol}^* = \mathsf{vol}(u,\infty)$. Either $\mathsf{vol}(u,\Delta/2)$ or $\mathsf{vol}_{rev}(v,\Delta/2)$ is at most $\mathsf{vol}^*/2$. (In the "vertex" version we replace $\Delta/2$ by $\Delta/3$ to handle the case where there is a vertex at distance exactly $\Delta/2$ from u.) For simplicity, from now on we assume that $\mathsf{vol}(u,\Delta/2) \leq \mathsf{vol}^*/2$; otherwise, consider the reversed graph and swap u and v.

Define $N^<(u,r)$ to be the set of all vertices whose distance from u is strictly less than r. The cut procedure grows a region $N(u,r)$ using a single-source shortest-path algorithm until the capacity of the cut $(N^<(u,r), V - N^<(u,r))$ is bounded by an expression that depends mainly on $\mathsf{vol}(u,r)$, as stated in the following theorem.

Theorem 2: Let $G = (V,E)$ denote a graph with edge capacities $c(e) \geq 1$ and non-negative edge lengths $\ell(e)$. Let $\mathsf{dist}(u,v) = \Delta$ for $u, v \in V$. Suppose that $\mathsf{vol}(u,\Delta/2) \leq \mathsf{vol}^*/2$. Then, there exists a radius $\Delta/4 \leq r \leq \Delta/2$ such that

$$c(N^<(u,r), V - N^<(u,r)) \leq \frac{4\mathsf{vol}(u,r)}{\Delta} \ln\left(\frac{e \cdot \mathsf{vol}^*}{2\mathsf{vol}(u,r)}\right) \ln\ln\left(\frac{e \cdot \mathsf{vol}^*}{2\mathsf{vol}(u,\Delta/4)}\right),$$

and either $r = \mathsf{dist}(u,x)$ for some $x \in V$, or $r = \Delta/2$. (The base of all logarithms is e.)

Proof: We start by proving the first part of the theorem, namely, that there exists some radius $\Delta/4 \leq r \leq \Delta/2$, for which the inequality holds. We then show that if $r < \Delta/2$, and also $r \neq$

$\mathsf{dist}(u,x)$ for all $x \in V$, then the same inequality holds either for $\mathsf{dist}(u,y)$, where y is the closest vertex to u outside $N(u,r)$ whose distance from u is at most $\Delta/2$, or if no such vertex exists, then the inequality holds for $\Delta/2$.

Let M be a common multiple of all the denominators of the edge lengths. Observe that ΔM is an integer, and that for all $x \in V$, $\mathsf{dist}(u,x)$ is of the form i/M for some non-negative integer i. Consider the sequence $b_i = \mathsf{vol}(u, i/(4M))$, for $\Delta M \le i \le 2\Delta M$. Note that $b_{2\Delta M} \le \mathsf{vol}^*/2$.

To prove the theorem we need the following two lemmas that are proven below.

Lemma 3: For $\Delta M \le i < 2\Delta M$,

$$b_{i+1} - b_i \ge \frac{1}{4M} \cdot c\left(N^{<}(u, \frac{i+1}{4M}), V - N^{<}(u, \frac{i+1}{4M})\right).$$

Lemma 4: [Difference Lemma] Let $0 < a_1 < a_2, \ldots, a_\ell < 1$. Then, there exists an index i, $1 \le i \le \ell - 1$, for which

$$a_{i+1} - a_i < \frac{a_{i+1}}{\ell - 1} \cdot \ln\left(\frac{e \cdot a_\ell}{a_{i+1}}\right) \cdot \ln\ln\left(\frac{e \cdot a_\ell}{a_1}\right).$$

To complete the proof of the theorem, apply the Difference Lemma to the sequence $\{b_i\}$, and let i be the index for which

$$b_{i+1} - b_i < \frac{b_{i+1}}{\Delta M} \cdot \ln\left(\frac{e \cdot \mathsf{vol}^*}{2b_{i+1}}\right) \cdot \ln\ln\left(\frac{e \cdot \mathsf{vol}^*}{2b_{\Delta M}}\right).$$

Let $r = (i+1)/(4M)$, by Lemma 3, $b_{i+1} - b_i \ge \frac{1}{4M} \cdot c(N^{<}(u,r), V - N^{<}(u,r))$. Thus, the first part of the theorem holds for the above choice of r. Suppose that $r < \Delta/2$ and $r \ne \mathsf{dist}(u,x)$ for all $x \in V$. We claim that in this case the inequality holds also for $r' \triangleq \min\{\Delta/2, \min_{x \in V - N(u,r)}\{\mathsf{dist}(u,x)\}\}$. This is true since $N^{<}(u,r) = N^{<}(u,r')$, and hence, these regions define the same cuts. On the other hand, the right hand side of the inequality monotonically increases as a function of $r \le \Delta/2$. □

Proof of Lemma 3: Let $r = (i+1)/(4M)$. By our definition $b_{i+1} = \mathsf{vol}(u,r)$ and $b_i = \mathsf{vol}(u, r - 1/(4M))$. Define $M_r \triangleq N(u,r) - N(u, r - 1/(4M))$; that is, M_r is the set of all vertices x such that $r - 1/(4M) < \mathsf{dist}(u,x) \le r$. However, since all distances are of the form i/M for some non-negative integer i, M_r is the set of all vertices of distance exactly r from u, if such exist. Define

$$F(u,r) \triangleq \{(x,y) \in E(u,r) : \mathsf{dist}(u,x) < \mathsf{dist}(u,y) = r\}.$$

Namely, $F(u,r)$ contains the edges emanating vertices in $N^{<}(u,r)$ and entering vertices in M_r. By the volume definition

$$\begin{aligned} b_{i+1} &= \sum_{e \in E(u,r)} c(e)\ell(e) + \sum_{e=(x,y) \in C(u,r)} c(e)(r - \mathsf{dist}(u,x)) \\ &\ge \sum_{e \in E(u, r-1/(4M))} c(e)\ell(e) + \sum_{e \in F(u,r)} c(e)\ell(e) + \sum_{e=(x,y) \in C(u,r)} c(e)(r - \mathsf{dist}(u,x)). \end{aligned}$$

Note that strict inequality holds if there are edges in $M_r \times M_r$. If $e = (x,y) \in F(u,r)$, then since $\mathsf{dist}(u,y) = r$, we infer that $\ell(e) \geq r - \mathsf{dist}(u,x)$. The definition of $F(u,r)$ implies that $C(u, r - 1/(4M)) \subseteq C(u,r) \cup F(u,r)$, and hence,

$$b_{i+1} - b_i \geq \frac{1}{4M} \sum_{e=(x,y)\in C(u,r-\frac{1}{4M})} c(e).$$

To complete the proof observe that there are no vertices whose distance from u is strictly between $r - 1/(4M)$ and r, hence, $C(u, r-1/(4M)) = c(N^{<}(u,r), V - N^{<}(u,r))$, and the lemma follows.
□

Proof of Lemma 4: Let $\gamma = e \cdot a_\ell$. Consider the sequence of differences defined by

$$\left\{ \ln\ln\left(\frac{\gamma}{a_{i+1}}\right) - \ln\ln\left(\frac{\gamma}{a_i}\right) \right\}_{i=1}^{\ell-1}$$

There exists a difference which is greater than or equal to the average difference. Hence, there exists an index i for which,

$$\frac{\ln\ln\left(\frac{\gamma}{a_\ell}\right) - \ln\ln\left(\frac{\gamma}{a_1}\right)}{\ell - 1} \leq \ln\ln\left(\frac{\gamma}{a_{i+1}}\right) - \ln\ln\left(\frac{\gamma}{a_i}\right)$$

By the Mean Value Theorem (Lagrange's Theorem), there exists a $\theta \in (a_i, a_{i+1})$ which satisfies,

$$\ln\ln\left(\frac{\gamma}{a_{i+1}}\right) - \ln\ln\left(\frac{\gamma}{a_i}\right) = \frac{-(a_{i+1} - a_i)}{\theta \cdot \ln(\gamma/\theta)}$$

Combining these two equations, we get,

$$\ln\left(\frac{\ln(\gamma/a_\ell)}{\ln(\gamma/a_1)}\right) \cdot \frac{1}{\ell - 1} \leq \frac{-(a_{i+1} - a_i)}{\theta \cdot \ln(\gamma/\theta)}$$

Note that $\gamma/a_\ell = e$, hence,

$$a_{i+1} - a_i \leq \theta \cdot \ln\frac{\gamma}{\theta} \cdot \ln\ln\left(\frac{\gamma}{a_1}\right) \cdot \frac{1}{\ell - 1}$$

The function $x \ln(\gamma/x)$ is monotone increasing in the interval $(0, \gamma/e]$. But, $\gamma/e = a_\ell$, hence,

$$\theta \cdot \ln\frac{\gamma}{\theta} < a_{i+1} \cdot \ln\frac{\gamma}{a_{i+1}}.$$

□

4 Generic Approximation Algorithms

We consider a problem Π that has a polynomial time computable spreading metric for the pair G, H whose volume is at most $\mathsf{cost}(G,H)$. We also assume that the solution of Π given by the divide-and-conquer method can be bounded from above as follows.

$$\mathsf{cost}(G,H) \le \begin{cases} 0 & \text{if } E_H = \emptyset. \\ \sum_{i=1}^{p} \mathsf{cost}(G_i, H_i) + \mathsf{scaler}(H) \cdot c(L,R) & \text{otherwise} \end{cases}$$

We now prove Theorem 1, showing that there is a polynomial time algorithm for Π with approximation factor $O(\min\{\ln\tau \ln\ln\tau, \ln k \ln\ln k\})$, where τ is the volume of the optimal spreading metric for Π, and $k \le n$ is the number of vertices that are not isolated in H. The algorithm is a simple divide-and-conquer algorithm. First, we compute the spreading metric for Π, and then use Theorem 2 as the divide step, as long as the subproblems defined by the resulting subgraphs are non-trivial.

We distinguish between two cases: (i) $\tau \le k$, in which we prove that the algorithm achieves the approximation factor $O(\ln\tau \ln\ln\tau)$; and (ii) $k \le \tau$, in which we prove that the algorithm achieves the approximation factor $O(\ln k \ln\ln k)$.

The case $\tau \le k$. For $\beta > 0$, define $P(\beta)$ to be the maximum cost of solving Π on a subgraph G' whose volume is at most β using our algorithm. Define

$$L \triangleq \frac{4}{\ln 2} \cdot \ln\ln(2e\tau)\,, \text{ and } F(\beta) \triangleq L \cdot \ln(4\beta)\,.$$

Since $L = O(\log\log\tau)$ and $F(\tau) = O(\log\tau \log\log\tau)$, to prove Theorem 1 for this case, it suffices to prove the following upper bound on $P(\beta)$.

$$P(\beta) \quad \le \quad \max(\beta F(\beta), 0) \qquad (1)$$

We prove Inequality (1) by induction on $\lfloor 4\beta \rfloor$. The induction basis $\lfloor 4\beta \rfloor < 4$, or $\beta < 1$, follows from the observation that if an induced subgraph G_U has volume $\beta < 1$, then $\mathsf{cost}(G_U, H_U) = 0$. This is because for all subsets $U \subseteq V$, for which G_U has volume $\beta < 1$, the subgraph H_U consists only of isolated vertices. We prove this assertion by contradiction. To obtain a contradiction, assume that H_U contains a non-isolated vertex v. This implies that there exists another vertex $u \in U$ such that $\mathsf{dist}(u,v) \ge \mathsf{scaler}(H) \ge 1$. However, since the capacity of each edge is at least one, this implies that the volume of G_U is at least one; a contradiction.

We now prove the inductive step. Let $\overline{\alpha} \triangleq 1 - \alpha$. Consider a graph G' of volume β that is divided into two subgraphs where one has volume at most $\alpha\beta$ and the other has volume at most $\overline{\alpha}\beta$ (where $\alpha \le 1/2$). Since the capacity of each edge is at least one, and since $\Delta \ge \mathsf{scaler}(H) \ge 1$, it follows that $\mathsf{vol}(u, \Delta/4) \ge 1/4$. By the algorithm (Theorem 2), $\mathsf{vol}(u, \Delta/2) \le \beta/2$. The radius of the region is between $\Delta/4$ and $\Delta/2$, hence, $1/4 \le \alpha\beta \le \beta/2$.

We apply the induction hypothesis to the subgraphs. Since the volume of the smaller part $\alpha\beta$

is at least $1/4$, we get that the cost of solving Π on the subproblems with volumes $\alpha\beta$ and $\overline{\alpha}\beta$ is bounded by $\alpha\beta F(\alpha\beta)$ and $\overline{\alpha}\beta F(\overline{\alpha}\beta)$, respectively.

Note that $F(\alpha\beta)$ increases as a function of α. Specifically,

$$F(\alpha\beta) = F(\beta) + L \cdot \ln(\alpha)$$

As long as $F(\alpha\beta)$ is positive (which is our case, since $\alpha\beta \geq 1/4$), the recursive cost of solving the two subproblems can be bounded as follows.

$$\begin{aligned} \alpha\beta F(\alpha\beta) + \overline{\alpha}\beta F(\overline{\alpha}\beta) &= \beta F(\beta) + (\alpha \ln(\alpha) + \overline{\alpha} \ln(\overline{\alpha})) \cdot \beta L \\ &\leq \beta F(\beta) + \alpha\beta L \ln(\alpha). \end{aligned}$$

We proceed by bounding the cost of the divide step in our algorithm. Recall that $\Delta \geq \mathsf{scaler}(H')$, where H' is the auxiliary graph corresponding to G'. Since $\mathsf{vol}(u, \Delta/4) \geq 1/4$, the capacity of the cut obtained using Theorem 2 is bounded by

$$\frac{4\alpha\beta}{\mathsf{scaler}(H')} \ln\left(\frac{e}{2\alpha}\right) \ln\ln\left(\frac{e\beta}{2 \cdot 1/4}\right).$$

Since $\beta \leq \tau$, it follows that the capacity of the cut is bounded by

$$\frac{\alpha\beta}{\mathsf{scaler}(H')} \ln\left(\frac{e}{2\alpha}\right) L \cdot \ln 2.$$

Since the cost incurred in the divide step equals the cost of the cut times $\mathsf{scaler}(H')$, we bound this cost by $\alpha\beta L \ln(2) \ln(e/(2\alpha))$.

A bound on the total cost is derived by combining the cost of solving the two subproblems with the cost of the divide step, to obtain

$$P(\beta) \leq \beta F(\beta) + \alpha\beta L \left(\ln\alpha + \ln(2) \ln\left(\frac{e}{2\alpha}\right)\right).$$

Since $\alpha \leq 1/2$, the expression in the parentheses is non-positive, and hence $P(\beta) \leq \beta F(\beta)$. Thus, Inequality (1) holds.

The proof of Theorem 1 is concluded in this case by noticing that the cost of solving Π on G is at most

$$P(\tau) = O\left(\tau \ln \tau \cdot \ln\ln \tau\right).$$

The case $k \leq \tau$. In this case we assume that for any subgraph of diameter Δ considered by our algorithm, if u is the vertex in this subgraph around which the region was grown, then $\mathsf{vol}(u, \Delta/4) \geq \lceil \tau/k \rceil$. This assumption can be made without loss of generality by modifying G and the spreading metric as follows. For every non-isolated vertex v of H, we add a self-loop in G touching the vertex v with capacity $4\lceil \tau/k \rceil/\Delta$. In the spreading metric we assign a length $\Delta/4$ to each such self-loop. So the volume associated with each self-loop equals $\lceil \tau/k \rceil$. These self-loops are removed as soon as the corresponding vertices in H become isolated. Since Theorem 2 guarantees the existence of a region whose radius is at least $\Delta/4$, it is easy to see that our assumption is justified provided that we always grow regions around non-isolated vertices of the current auxiliary graph (which we do by choosing u and v to be non-isolated vertices whose distance is at least the value of the scaler function). We note that the modification increased the volume of the whole graph by at most $(4\lceil\tau/k\rceil/\Delta)(\Delta k)/4 \leq k(1+\tau/k) < 2\tau$. This implies that $\mathsf{vol}^* < 3\tau$. Although vol^* may no longer be a lower bound on the cost of the solution, it is only off such a bound by at most a constant.

Similar to the previous case, define $P(\beta)$ to be the maximum cost of solving Π on a subgraph G' of volume β using our algorithm. Define now,

$$L \triangleq \frac{4}{\ln 2}\ln\ln\left(\frac{3e\cdot k}{2}\right), \text{ and } F(\beta) \triangleq L\cdot\ln\left(\frac{\beta}{\lceil\tau/k\rceil}\right).$$

Note that now $L = O(\log\log k)$ and $F(\tau) = O(\log\tau\log\log\tau)$. As before, to prove Theorem 1, it suffices to prove the upper bound 1 on $P(\beta)$; that is, $P(\beta) \leq \max(\beta F(\beta), 0)$.

We prove Inequality (1) for the case $\tau \geq k$ by induction on $\lfloor\frac{\beta}{\lceil\tau/k\rceil}\rfloor$. The induction basis $\lfloor\frac{\beta}{\lceil\tau/k\rceil}\rfloor = 0$ follows since if an induced subgraph G_U has volume $\beta < \lceil\tau/k\rceil$, then $\mathsf{cost}(G_U, H_U) = 0$. To see this, note that by our assumption, non-isolated vertices of the current auxiliary graph have self-loops of volume $\lceil\tau/k\rceil$. Hence, if G_U has volume $\beta < \lceil\tau/k\rceil$, then the corresponding subgraph H_U consists only of isolated vertices, and $\mathsf{cost}(G_U, H_U) = 0$.

Consider a graph G' of volume β that is divided into two subgraphs where one has volume at most $\alpha\beta$ and the other has volume at most $\overline{\alpha}\beta$ (where $\alpha < 1/2$ and $\overline{\alpha} = 1-\alpha$). By our assumptions, $\mathsf{vol}(u, \Delta/4) \geq \lceil\tau/k\rceil$, and $\mathsf{vol}(u, \Delta/2) \leq \beta/2$. The radius of the region is between $\Delta/4$ and $\Delta/2$, hence, $\lceil\tau/k\rceil \leq \alpha\beta \leq \beta/2$.

We apply the induction hypothesis to the subgraphs. Since the volume of the smaller part $\alpha\beta$ is at least $\lceil\tau/k\rceil$, the cost of solving Π on the subproblems with volumes $\alpha\beta$ and $\overline{\alpha}\beta$ is bounded by $\alpha\beta F(\alpha\beta)$ and $\overline{\alpha}\beta F(\overline{\alpha}\beta)$, respectively.

As in the previous case, the recursive cost of solving the two subproblems can be bounded as follows.

$$\alpha\beta F(\alpha\beta) + \overline{\alpha}\beta F(\overline{\alpha}\beta) \leq \beta F(\beta) + \alpha\beta L\ln(\alpha).$$

The cost of the divide step is bounded by

$$\frac{4\alpha\beta}{\mathsf{scaler}(H')}\ln\left(\frac{e}{2\alpha}\right)\ln\ln\left(\frac{e\beta}{2\lceil\tau/k\rceil}\right).$$

Since $\beta < 3\tau$, it follows that $\frac{e\beta}{2\lceil\tau/k\rceil} \leq \frac{3ek}{2}$, and we obtain the new bound

$$\frac{\alpha\beta}{\mathsf{scaler}(H')}\ln\left(\frac{e}{2\alpha}\right)L\ln 2.$$

Since the cost incurred in the divide step equals the cost of the cut times $\mathsf{scaler}(H')$, we bound this cost by $\alpha\beta L\ln(2)\ln(e/2\alpha)$. The proof from this point on is identical to the case $\tau \leq k$ and shows that the cost of solving Π on G is at most

$$P(3\tau) = O\left(\tau \ln\left(\frac{\tau}{\lceil \tau/k \rceil}\right) \ln\ln(k)\right).$$

5 Balanced decomposition trees

In this section we present a balancing technique that rearranges the cuts found according to Section 3. The balancing is required for applications in which divide-and-conquer methods are not useful for the problem in hand, unless each partition is *balanced*. Our balancing technique is applicable whenever the scaler function depends on the number of non-isolated vertices in the auxiliary graph. In such cases a balanced partition of a subproblem, $\Pi = (G, H)$, is a partitioning of the vertices of G that divides the subproblem Π into subproblems $\Pi_1 = (G_1, H_1), \Pi_2 = (G_2, H_2), \ldots$, such that the number of non-isolated vertices in each subgraph H_i is at most two thirds of the number of non-isolated vertices in H. (Note that fragmenting the problem into many small problems only helps us, however, we want to avoid large subproblems after the partitioning. This is why the balancing condition only sets upper bounds on the sizes of the subproblems).

We address this balancing problem using the concept of a *decomposition tree*. Our partitioning algorithm for any problem Π defines a decomposition tree of the graph where each tree-node holds a subproblem. The root of the decomposition tree holds the original problem, and the children of each node t in the tree hold the subproblems obtained by partitioning the problem held by t. The leaves of the decomposition tree hold subproblems that can be solved directly without any further partitioning. We define the cost of an internal node in the tree that holds a subproblem (G_s, H_s) to be $\mathsf{scaler}(H_s)$ times the cut cost incurred in the divide step for this node. We define the cost of a leaf to be the cost of the subproblem it holds. The cost of the tree is the total cost of the nodes in the tree, and constitutes an upper bound on the cost of solving Π by the divide-and-conquer algorithm induced by the decomposition tree, provided that the decomposition tree is balanced.

We show how to derive a balanced decomposition tree from the unbalanced decomposition tree with only a constant multiplicative factor increase in the cost of the tree. Thus, we can find an unbalanced decomposition tree, produce a balanced tree from it, and then apply the divide-and-conquer algorithm using the partitions in the balanced decomposition tree. To simplify the exposition, we further assume that the auxiliary graph H is a clique graph on V, which suffices for our applications (e.g., graph embeddings in d-dimensions).

Given an unbalanced tree T, we obtain a balanced tree T' from it by coalescing tree-nodes. We first describe a procedure that finds a balanced partition of a given set of subproblems. Suppose that the problem held by tree-node $t' \in T'$ is partitioned into a set of subproblems $\Pi_{t_1}, \Pi_{t_2}, \ldots$, where subproblem Π_{t_i} is held by tree-node t_i in the unbalanced tree T. Let $n(s)$ denote the number of vertices contained in the subproblems held by a tree-node s. A straightforward partitioning approach is to construct two children of t', denoted by a and b, and divide the subproblems held by tree-node t' between a and b, so that $\max\{n(a), n(b)\} \leq 2/3 \cdot n(t')$. However, this approach fails if t' holds one relatively large subproblem. In such a case, we must partition a large subproblem into smaller subproblems. The partitioning of a subproblem Π_{t_j} is defined by the unbalanced decomposition tree T, namely, by the decomposition of subproblem Π_{t_j} into the subproblems held by the children of $t_j \in T$. The procedure for a balanced partition of subproblems held by tree-node t' is a combination of these two approaches, as described below.

Initially, a holds all the problems held by t' and b is empty. While $n(a) > 2/3 \cdot n(t')$, we pick a subproblem Π_{t_i} held by a and remove it from the set of subproblems held by a. We consider two cases: If $n(t_i) \leq n(t')/3$, then Π_{t_i} is added to the set of subproblems held by b. Otherwise, consider the partitioning of Π_{t_i} defined by the children of t_i in the unbalanced tree T. This partitioning divides Π_{t_i} into at least two subproblems. Add a smallest subproblem to the set of subproblems held by b, and add all the other subproblems to the set of subproblems held by a. Note that the subproblem added to b contains no more that $n(t_i)/2$ vertices. When this loop ends, $1/3 \cdot n(t') \leq n(a) \leq 2/3 \cdot n(t')$. The lower bound follows because when an iteration starts $n(a) > 2/3 \cdot n(t')$, and in each iteration $n(a)$ decreases by at most $n(a)/2$. This implies that $n(b) \leq 2/3 \cdot n(t')$, and hence a balanced partitioning of the subproblems held by t' is obtained. The balanced decomposition tree T of problem Π is obtained by assigning problem Π to the root of T' and recursively applying the procedure for a balanced partitioning of subproblems to the nodes of T'. This recursion is halted when a tree-node $t' \in T'$ holds only trivial subproblems, namely, subproblems that can be solved without any further partitioning.

Our assumption that the auxiliary graph H is a clique spanning all the vertices, implies that all the edges of the graph G belong to cuts in both trees, T and T'. For an edge $e \in E$, let $t(e)$ denote the tree-node in T whose cut contains e. Namely, e connects vertices that "belong" to different children of $t(e)$. Similarly, define $t'(e)$ relative to the balanced tree T'. The following lemma shows that $n(t'(e))/n(t(e))$ is bounded by a constant.

Lemma 5: For every edge $e \in E$, $n(t'(e)) < 3 \cdot n(t(e))$.

Proof: Let $\Pi_{t(e)}$ denote the subproblem corresponding to tree-node t in the unbalanced tree T. The construction of the balanced tree T' implies that the edge e belongs to the cut corresponding to $t'(e)$ following a partitioning of the subproblem $\Pi_{t(e)}$. This partitioning takes place only if $n(t(e)) > n(t'(e))/3$, and the lemma follows. □

Define $\beta \triangleq \sup \left\{ \frac{\mathsf{scaler}(x)}{\mathsf{scaler}(\frac{1}{3}x)} \right\}$. The value of β is well defined and constant if we assume that the function $\mathsf{scaler}(\cdot)$ depends only on the cardinality of the vertices in the auxiliary graph, it is monotone non-decreasing, and it is bounded by a polynomial. We are now ready to prove that the cost of the balanced decomposition tree T' almost equals the cost of an unbalanced decomposition tree T.

Theorem 6: The cost of the balanced decomposition tree is at most β times the cost of an unbalanced decomposition tree obtained by the separator procedure.

Proof: Lemma 5, the fact that $\mathsf{scaler}(\cdot)$ is monotone non-decreasing, and the definition of β, imply that for every edge $e \in E$,

$$\frac{\mathsf{scaler}(n(t'(e)))}{\mathsf{scaler}(n(t(e)))} \leq \frac{\mathsf{scaler}(n(t'(e)))}{\mathsf{scaler}(n(t'(e))/3)} \leq \beta$$

Hence,

$$\begin{aligned} \mathrm{cost}(T') &= \sum_{e\in E} \mathsf{scaler}(n(t'(e)))\cdot c(e) \\ &\le \sum_{e\in E} \beta\cdot\mathsf{scaler}(n(t(e)))\cdot c(e) \\ &= \beta\cdot\mathrm{cost}(T) \end{aligned}$$

□

6 Applications

In this section we present seven optimization problems and show how they can be cast in the paradigm of Section 2. For each problem we dedicate a subsection in which we define the problem and show how to cast it in our paradigm.

6.1 Linear arrangement

The linear arrangement problem is defined as follows.
Input: An undirected graph $G=(V,E)$ with a capacity $c(e)$ associated with each edge $e\in E$.
Output: A linear arrangement of the vertices $h:V\to\{1,\ldots,|V|\}$, that minimizes the total edge lengths, i.e., $\sum_{e=(i,j)\in E} c(e)\cdot|h(i)-h(j)|$.

In the context of VLSI layout, $|h(i)-h(j)|$ is referred to as the length of the interconnection between i and j. Finding an optimal linear arrangement is NP-hard. (See [GJ79, problem GT42, p. 200].)

We show how to cast this problem in our paradigm. The graph $H=(V,E_H)$ associated with $G=(V,E)$ is the clique graph $C_{|V|}$, and the scaler function is defined by $\mathsf{scaler}(H=(V,E_H))\triangleq|V|-1$. Clearly, if H contains no edges, then $|V|=1$ and $\mathsf{cost}(G,H)=0$. If $|V|>1$, then the length of every edge in the edge cut is at most $|V|-1$, therefore,

$$\mathsf{cost}(G,C_{|V|}) \quad\le\quad \mathsf{cost}(G_L,C_{|L|})+\mathsf{cost}(G_R,C_{|R|})+(|V|-1)\cdot c(L,R),$$

where G_L is the subgraph of G induced by L and G_R is the subgraph of G induced by R. This establishes the divide-and-conquer applicability.

We now show how to compute the spreading metric. Consider the following linear program.

$$\begin{aligned} \min\quad & \sum_{e\in E} c(e)\cdot\ell(e) \\ \text{s.t.}\quad & \forall U\subseteq V,\ \forall v\in U:\ \sum_{u\in U}\mathsf{dist}(u,v)\ge\frac{1}{4}(|U|^2-1) \\ & \forall e\in E:\ \ell(e)\ge 0 \end{aligned}$$

In the linear program, we follow our previous notation that regards $\ell(e)$ as edge lengths, and

$\mathsf{dist}(u,v)$ is the length of the shortest path from u to v.

Lemma 7: Let $\ell(e)$ denote a feasible solution of the linear program. For every subset $U \subseteq V$ ($|U| > 1$), and for every vertex $v \in U$ there is a vertex $u \in U$ for which $\mathsf{dist}(u,v) \geq \frac{1}{4}|U|$.

Proof: The average distance of a node $u \in U - \{v\}$ from v is at least $\frac{1}{4}(|U|+1)$, because of the constraint corresponding to U and v. Therefore, there exists a vertex $u \in U$ whose distance from v is at least the average distance from v, and the lemma follows. □

Note that the previous lemma comes short of the diameter guarantee by a factor of 4. Namely, the diameter of a subset U may not be greater than $\mathsf{scaler}(H_U)$, but it is greater than $\mathsf{scaler}(H_U)/4$. This only affects the constant in the approximation factor.

In the next lemma we prove that the volume of an optimal solution of the linear program satisfies the lower bound property.

Lemma 8: The cost of an optimal solution of the linear program is a lower bound on the cost of an optimal linear arrangement of G.

Proof: Consider a linear arrangement $h : V \rightarrow \{1, \ldots, |V|\}$ of G. Define $\ell(e) = |h(i) - h(j)|$ for $e = (i,j) \in E$. Clearly, the cost $\sum_{e \in E} c(e) \cdot \ell(e)$ equals the cost of the linear arrangement $h(\cdot)$. Feasibility of $\ell(\cdot)$ is proven as follows. Consider a subset $U \subseteq V$, and a vertex $v \in U$. Let U_L denote the vertices of U who are to the left of v, namely, $U_L \triangleq \{u \in U : h(u) < h(v)\}$. Similarly, define $U_R \triangleq \{u \in U : h(u) > h(v)\}$. Now,

$$\begin{aligned}
\sum_{u \in U} \mathsf{dist}(u,v) &= \sum_{u \in U_L} \mathsf{dist}(u,v) + \sum_{u \in U_R} \mathsf{dist}(u,v) \\
&\geq \sum_{i=1}^{|U_L|} i + \sum_{i=1}^{|U_R|} i \\
&\geq \sum_{i=1}^{\lfloor \frac{|U|-1}{2} \rfloor} i + \sum_{i=1}^{\lceil \frac{|U|-1}{2} \rceil} i \\
&\geq \frac{|U|^2 - 1}{4}
\end{aligned}$$

Hence, $\ell(\cdot)$ is a feasible solution and the lemma follows. □

Finally, we show that the spreading metric can be computed in polynomial time.

Lemma 9: The linear program can be solved in polynomial time.

Proof: The linear program has an exponential number of constraints. However, it can be solved using the Ellipsoid algorithm. This follows by observing that a violated constraint of a given non-feasible solution can be found in polynomial time. Simply, run a single-source shortest paths algorithm from every vertex v, and check that for all values of k, $1 \leq k \leq |V|$, the k closest vertices to v satisfy the appropriate constraint in the linear program. □

6.2 Graph embeddings in d-dimensions

The second graph embedding problem we consider is graph embedding in d-dimensional meshes defined as follows.

Input: An undirected graph $G = (V, E)$ with capacity $c(e)$ associated with each edge $e \in E$.

Output: A one-to-one mapping, h, of G to a subgraph containing $|V|$ vertices of the d-dimensional grid. The value of the embedding is $\sum_{(u,v)\in E} d(h(u), h(v))$, where $d(x, y)$ is the number of mesh-edges in the shortest path between x and y in the mesh.

A related graph embedding problem we consider is linear arrangement with d-dimensional cost.

Input: An undirected graph $G = (V, E)$ with capacity $c(e)$ associated with each edge $e \in E$.

Output: A linear arrangement of the vertices $h : V \to \{1, \ldots, |V|\}$, that minimizes the total d-dimensional edge cost, i.e., $\sum_{e=(i,j)\in E} c(e) \cdot |h(i) - h(j)|^{1/d}$.

We refer to the cost function in the latter problem as the d-dimensional cost of a linear ordering. The following lemma reduces the problem of graph embeddings in d-dimensional meshes to the problem of linear arrangements with d-dimensional cost.

Lemma 10: Given a linear ordering of G of d-dimensional cost-value A, one can produce a d-dimensional embedding of G of cost-value $O(d \cdot |A|)$.

Proof sketch: This follows directly from the existence of a curve through a d-dimensional grid, where two vertices that differ by k along the curve are at distance $O(dk^{1/d})$ in the d-dimensional mesh. A Peano curve is an example of such a curve. See [S94] for a description of space filling curves. $\square$

Note that due to Lemma 10, the dimensionality of the mesh, namely d, increases the approximation factor by a linear factor in d.

We define the auxiliary graph $H = (V, E_H)$ associated with $G = (V, E)$ to be the clique graph $C_{|V|}$, and the scaler function by $\mathsf{scaler}(H) \triangleq (|V| - 1)^{1/d}$. As in the linear arrangement problem, the length of an edge of a cut is at most $(|V| - 1)$. Therefore, the cost associated with each cut-edge is at most $(|V| - 1)^{1/d}$, and the divide-and-conquer algorithm satisfies the following recurrence inequality:

$$\mathsf{cost}(G, C_{|V|}) \leq \mathsf{cost}(G_L, C_{|L|}) + \mathsf{cost}(G_R, C_{|R|}) + (|V| - 1)^{1/d} \cdot c(L, R),$$

where G_L is the subgraph of G induced by L and G_R is the subgraph of G induced by R.

The spreading metric is obtained by solving the following linear program.

$$\begin{array}{ll} \min & \sum_{e\in E} c(e) \cdot \ell(e) \\ \text{s.t.} & \forall U \subseteq V,\ \ \forall v \in U : \sum_{u\in U} \mathsf{dist}(u, v) \geq \frac{1}{4} \cdot (|U| - 1)^{1+1/d} \\ & \forall e \in E : \ \ell(e) \geq 0 \end{array}$$

The following lemma is analogous to Lemma 7, and proves that the diameter guarantee is satisfied upto a constant of 4.

Lemma 11: Let $\ell(e)$ denote a feasible solution of the linear program. For every subset $U \subseteq V$ ($|U| > 1$), and for every vertex $v \in U$ there is a vertex $u \in U$ for which $\mathrm{dist}(u, v) \geq \frac{1}{4} \cdot (|U| - 1)^{1/d}$.

The following lemma is analogous to Lemma 8, and proves that the lower bound property is satisfied.

Lemma 12: The cost of an optimal solution of the linear program is a lower bound on the cost of a minimum d-dimensional cost of any linear ordering of G.

Proof: The proof follows the proof of Lemma 8. Consider a linear ordering $h(\cdot)$ of the vertices of G into a mesh. For every edge $(i, j) \in E$, define $\ell(i, j)$ to be $|h(j) - h(i)|^{1/d}$. Clearly, the cost $\sum_{e \in E} c(e) \cdot \ell(e)$ equals the d-dimensional cost of the embedding $h(\cdot)$. Feasibility of $\ell(\cdot)$ is proved as follows. Consider a subset $U \subseteq V$, and a vertex $v \in U$. Vertices in U are at integral distances from v and at most two vertices are at any particular distance. We can thus derive the following inequality.

$$\begin{aligned}
\sum_{u \in U} \mathrm{dist}(u, v) &\geq \sum_{i=1}^{\lfloor \frac{|U|-1}{2} \rfloor} i^{1/d} + \sum_{i=1}^{\lceil \frac{|U|-1}{2} \rceil} i^{1/d} \\
&\geq \int_0^{\lfloor \frac{|U|-1}{2} \rfloor} x^{1/d} dx + \int_0^{\lceil \frac{|U|-1}{2} \rceil} x^{1/d} dx \\
&\geq 2 \frac{d}{1+d} \cdot \left(\frac{|U| - 1}{2} \right)^{1+1/d} \\
&\geq \frac{1}{4} (|U| - 1)^{1+1/d} \quad \text{(assuming } d \geq 1\text{)}.
\end{aligned}$$

□

The proof of the following lemma follows the proof of Lemma 9.

Lemma 13: The linear program can be solved in polynomial time.

Remark: An alternative method for finding a d-dimensional embedding is by following Hansen's technique [Ha89]. However, this requires finding a balanced partitioning of the graph so as to maintain a constant aspect ratio of the meshes in the recursive calls. This can be done using balanced decomposition trees that are presented in Section 5.

6.3 Minimizing storage-time product

Following Ravi *et al.* [RAK91], we consider the storage-time product minimization. To make the presentation clearer we consider here only the special case in which all tasks require unit time. However, our algorithm applies also to the more general case where the times differ.

Input: A directed acyclic graph $G = (V, E)$ with edge capacities $c(e)$. The vertices of G represent tasks to be scheduled. An edge $e = (u \rightarrow v)$ with capacity $c(e)$ corresponds to $c(e)$ units of storage generated by task u and consumed by task v.

Output: A linear ordering $h : V \rightarrow \{1, \ldots, |V|\}$ of the vertices such that if $(u \rightarrow v) \in E$, then

$h(u) < h(v)$. This ordering corresponds to a scheduling of the tasks that obeys the precedence constraints. The goal is to minimize the product storage-time. Assuming that all tasks are of unit length, this translates to minimizing the cost of the ordering defined by $\mathsf{cost}(h) \triangleq \sum_{u \to v}(h(v) - h(u)) \cdot c(u \to v)$.

The divide-and-conquer condition is applied in the same manner as it is applied to the undirected linear arrangement problem. The graph $H = (V, E_H)$ associated with $G = (V, E)$ is the clique graph $C_{|V|}$, and the scaler function is defined by $\mathsf{scaler}(H = (V, E_H)) \triangleq |V| - 1$. Since the graph here is directed we need to modify the way the spreading metric is computed. Following Ravi *et al.* [RAK91], we augment the graph by adding reversed edges, $E^R \triangleq \{v \to u : u \to v \in E\}$ with infinite capacity. The spreading metrics is defined by the following linear program:

$$\begin{aligned} \min \quad & \sum_{e \in E \cup E^R} c(e) \cdot \ell(e) \\ \text{s.t.} \quad & \forall U \subseteq V, \ \ \forall v \in U : \ \sum_{u \in U} (\mathsf{dist}(u,v) + \mathsf{dist}(v,u)) \geq \frac{1}{4}(|U|^2 - 1) \\ & \forall e \in E \cup E^R : \ \ell(e) \geq 0 \end{aligned}$$

We use the convention that $\infty \cdot 0 = 0$, and hence, infinite capacity edges are assigned zero length.

The required properties of the spreading metric are proved in the following Lemmata, which are analogous to Lemmata 7-9, and are also proved similarly.

Lemma 14: Let $\ell(e)$ denote a feasible solution of the linear program. For every subset $U \subseteq V$ ($|U| > 1$), and for every vertex $v \in U$ there is a vertex $u \in U$ for which

$$\max\{\mathsf{dist}(u,v), \mathsf{dist}(v,u)\} \geq \frac{1}{8} \cdot |U|.$$

Note that the bound on the diameter of a subset U is $\frac{1}{8} \cdot |U|$, which is smaller than the value $|U| - 1$ attached by the scaler function. However, the ratio is at most 8, and hence, this affects only the constant in the approximation factor.

Lemma 15: The cost of an optimal solution of the linear program is a lower bound on the cost of an optimal linear ordering of G.

Proof: The proof follows the proof of Lemma 8, the only difference is that reversed edges are given zero length. □

Lemma 16: The linear program can be solved in polynomial time.

6.4 Interval graph completion

The interval graph completion problem is defined as follows.
Input: A connected undirected graph $G = (V, E)$.
Output: A minimum cardinality set of edges F such that $G = (V, E \cup F)$ is an interval graph.

We present a novel method for obtaining a lower bound on the optimal interval completion problem that is based on a linear programming formulation. We rely on the characterization of interval graphs due to [RP88]: A graph is an interval graph if and only if there exists a linear ordering of the vertices such that if a vertex u with index i has an edge to vertex v with index j, where $i < j$, then every vertex whose index is between i and j also has an edge to vertex v. Note that such an ordering of the vertices of G uniquely defines a completion of G into an interval graph. Thus, it suffices to show how to compute such an optimal ordering.

The following lemma uses the above characterization to establish a relation between the difference in indices of pairs of vertices and the sum of vertex degrees along any path connecting them.

Lemma 17: Let $H = (V, E)$ be an interval graph. Let $h : V \to \{1, \ldots, |V|\}$ be a linear ordering of its vertices with the property guaranteed by the above characterization. Then, for every path P, $p = v_1, v_2, \ldots, v_p$ in H,

$$|h(v_p) - h(v_1)| \leq \sum_{i=1}^{p} \mathsf{deg}(v_i),$$

where $\mathsf{deg}(v)$ denotes the degree of vertex v.

Proof: Without loss of generality assume that $h(v_p) \geq h(v_1)$. Consider only the set of "right-going" edges in path P, denoted by P_r. For these edges $h(v_{i+1}) \geq h(v_i)$. The above characterization ensures that the right endpoint of each such edge (v_i, v_{i+1}) satisfies $\mathsf{deg}(v_{i+1}) \geq h(v_{i+1}) - h(v_i)$. Thus,

$$\begin{aligned} h(v_p) - h(v_1) &\leq \sum_{(v_i, v_{i+1}) \in P_r} (h(v_{i+1}) - h(v_i)) \\ &\leq \sum_{(v_i, v_{i+1}) \in P_r} \mathsf{deg}(v_{i+1}) \\ &\leq \sum_{i=1}^{p} \mathsf{deg}(v_i). \end{aligned}$$

□

We apply the "vertex-version" of the paradigm as follows. Each vertex is assigned a unit capacity. The graph $H = (V, E_H)$ associated with $G = (V, E)$ is the clique graph $C_{|V|}$, and the scaler function is defined by $\mathsf{scaler}(H = (V, E_H)) = |V| - 1$. Following Ravi *et al.* [RAK91], we show that nested dissection satisfies the required recurrence inequality. Namely, for every vertex separator U that separates V into two sets L and R, if we order the vertices of L and R recursively, and place the ordering of L before R (or vice-versa), followed by any ordering of U, then

$$\mathsf{cost}(G, C_{|V|}) \leq \mathsf{cost}(G_L, C_{|L|}) + \mathsf{cost}(G_R, C_{|R|}) + \quad (|V| - 1) \cdot |U|$$

where G_L and G_R are the subgraphs of G induced by L and R. The above inequality holds because at most $(|V| - 1)$ edges need to be added per vertex in U. This establishes the divide-and-conquer applicability.

The spreading metric is obtained by solving the following linear program. Since we are using a vertex version of the paradigm, we attach a length $\ell(v)$ to each vertex v, where $\ell(v)$ can be viewed as the fractional degree of v; and $\mathsf{dist}(u,v)$ denotes the shortest path connecting u and v with respect to the vertex lengths $\ell(v)$.

$$\begin{array}{ll} \min & \frac{1}{2}\sum_{v\in V}\ell(v) \\ \text{s.t.} & \forall U\subseteq V, \forall v\in U : \sum_{u\in U}\mathsf{dist}(u,v)\geq \frac{1}{4}(|U|^2-1) \\ & \forall v\in V : \ell(v)\geq 0 \end{array}$$

The following three lemmata are analogous to Lemmata 7-9, and are proved similarly.

Lemma 18: Let $\ell(e)$ denote a feasible solution of the linear program. For every subset $U\subseteq V$, and for every vertex $v\in U$, there exists a vertex $u\in U$ for which $\mathsf{dist}(u,v)\geq\frac{1}{4}|U|$.

Lemma 19: The cost of an optimal solution of the linear program is a lower bound on the minimum number of edges in an interval graph completion of G.

Proof: Consider a completion of G to into an interval graph $G^*(V,E^*)$. For every $v\in V$, let $\ell(v)$ be the degree of v in G^*. Clearly, the cost $\frac{1}{2}\sum_{v\in V}\ell(v)$ equals the number of edges in E^*. We show feasibility of this solution by considering a linear ordering $h(\cdot)$ of G^* with the property guaranteed by the characterization above. Fix a subset $U\subseteq V$ and a vertex $v\in U$. Consider a shortest path P, with respect to $\ell(\cdot)$, that connects v and u, i.e. $\mathsf{dist}(u,v)=\sum_{x\in P}\ell(x)$. Since $\ell(x)$ is the degree of x in G^*, by Lemma 17 we have $\sum_{x\in P}\ell(x)\geq|h(u)-h(v)|$. Summing over all vertices in U and using Lemma 8, we get

$$\sum_{u\in U}\mathsf{dist}(u,v)\geq\sum_{u\in U}|h(u)-h(v)|\geq\frac{1}{4}(|U|^2-1)$$

and the lemma follows. □

Lemma 20: The linear program can be solved in polynomial time.

6.5 Feedback sets in directed graphs and multicuts in circular networks

In this subsection we consider a generalization of the weighted feedback set called the weighted subset feedback set problem. This problem has two versions, subset feedback edge set (SUBSET-FES), and subset feedback vertex set (SUBSET-FVS), which were shown to be equivalent in [ENSS95]. Thus, we consider here only the SUBSET-FES problem defined as follows.

Input: A directed graph $G=(V,E)$ with capacities $c(e)$ associated with each edge $e\in E$. A subset $X\subseteq V$ of "special" vertices, where $|X|=k$.

Output: A minimum capacity subset of edges that intersects every interesting cycle, i.e., a cycle containing a vertex from X.

In [ENSS95] it is shown that this problem is equivalent to the problem of finding a multicut in circular network which is defined as follows.

Input: A directed graph $G = (V, E)$ with a capacity $c(e)$ associated with every $e \in E$, and a set of k *source-sink pairs* $\{(s_i, t_i)\}_{i=1,\dots,k}$. For every source-sink pair, (s_i, t_i), there is an infinite capacity edge $t_i \rightarrow s_i$.

Output: A minimum capacity subset of edges $F \subseteq E$ that intersects every path in G from a source to its corresponding sink.

We show how to cast the multicut problem in our paradigm. The directed graph $H = (V, E_H)$ associated with $G = (V, E)$ is the "demands" graph, defined by the set of edges $E_H = \{(s_i, t_i)\}_{i=1,\dots,k}$. The scaler function is defined by $\mathsf{scaler}(H = (V, E_H)) = 1$ if E_H is non-empty, and 0 otherwise. Clearly, if $E_H = \emptyset$, then all the source-sink pairs are separated and $\mathsf{cost}(G, H) = 0$. Suppose that $E_H \neq \emptyset$ and we add a cut (L, R) of finite weight to the multicut, then the following recurrence inequality holds.

$$\mathsf{cost}(G, H) \leq \mathsf{cost}(G_L, H_L) + \mathsf{cost}(G_R, H_R) + c(L, R),$$

where G_L and H_L are the subgraphs of G and H induced by L, and G_R and H_R are the subgraphs of G and H induced by R. Note that this recurrence does not hold for arbitrary directed multicut problems, but holds only for circular networks which have infinite capacity edges from each sink to its corresponding source. This establishes the divide-and-conquer applicability.

The spreading metric is defined by the following linear program.

$$\begin{array}{ll} \min & \sum_{e \in E} c(e) \cdot \ell(e) \\ \text{s.t.} & \text{For } 1 \leq i \leq k, \text{ for every path } P \text{ from } s_i \text{ to } t_i\text{: } \sum_{e \in P} \ell(e) \geq 1 \\ & \forall e \in E : \ \ell(e) \geq 0 \end{array}$$

The following three lemmata prove that the spreading metric meets the required properties. The next lemma is immediate.

Lemma 21: Let $\ell(e)$ denote a feasible solution of the linear program. For every subset $U \subseteq V$, if U contains a source-sink pair, then the distance from the source to the sink is at least 1.

Lemma 22: The cost of an optimal solution of the linear program is a lower bound on the cost of an optimal multicut of G.

Proof: Given a multicut of G, denoted by F, define $\ell(e)$ to be the indicator function of F. Clearly, $\ell(\cdot)$ is feasible and its cost equals the capacity of F. □

Lemma 23: The linear program can be solved in polynomial time.

Proof: The proof follows the proof of Lemma 9. Given edge lengths, one can check feasibility by checking whether the length of a shortest paths from s_i to t_i is at least 1. □

6.6 Symmetric multicuts in directed networks

Klein *et al.* [KPRT93] considered a variant of network decomposition, called symmetric multicuts:

Input: A directed graph $G = (V, E)$ with a capacity $c(e)$ associated with every $e \in E$, and a set of k *terminal pairs* $\{(s_i, t_i)\}_{i=1,\ldots,k}$.

Output: A minimum capacity subset of edges $F \subseteq E$ such that every terminal pair is separated into two different strongly connected components in the graph $G' = (V, E - F)$.

The symmetric multicut problem is cast in our paradigm similarly to the multicut problem. The only difference is that the auxiliary graph is undirected, and has edges connecting terminal pairs.

The spreading metric is computed by the following linear program.

$$\begin{array}{ll} \min & \sum\limits_{e \in E} c(e) \cdot \ell(e) \\ \text{s.t.} & \text{For } 1 \le i \le k \text{, for every cycle } C \text{ that intersects } s_i \text{ and } t_i\text{: } \sum\limits_{e \in C} \ell(e) \ge 1 \\ & \forall e \in E : \ \ell(e) \ge 0 \end{array}$$

The following three lemmata are similar to Lemmata 21-23.

Lemma 24: Let $\ell(e)$ denote a feasible solution of the linear program. For every subset $U \subseteq V$, if U contains a terminal pair (s_i, t_i), then either the distance from s_i to t_i is at least $1/2$, or the distance from t_i to s_i is at least $1/2$.

Lemma 25: The cost of an optimal solution of the linear program is a lower bound on the cost of an optimal symmetric multicut of G.

Lemma 26: The linear program can be solved in polynomial time.

6.7 Multiway separators and ρ-separators in directed graphs

We first discuss the directed ρ-separator problem. The problem of finding ρ-separators in undirected and directed graphs is introduced in [ENRS95]. We consider here only directed graphs since the approximation factor given in [ENRS95] for undirected graphs is better than the approximation factor described here. However, both cases can be cast into our paradigm. For simplicity, we consider here only the case in which all vertices have unit weights (although edges may have different capacities); the weighted case is described in [ENRS95].

For $0 < \rho \le 1$, recall that a ρ-separator in a directed graph $G = (V, E)$ is a subset of edges whose removal partitions G into strongly connected components, each of which contains at most $\rho \cdot n$ vertices, where $n = |V|$. The directed ρ-separator problem is defined as follows:

Input: A directed graph $G = (V, E)$ with a capacity $c(e)$ associated with every $e \in E$, and a parameter $0 < \rho \le 1$.

Output: A minimum capacity directed ρ-separator.

The performance of the approximation algorithm for this problem differs from the other problems treated in this paper, and is sometimes called a "pseudo-approximation algorithm" or a "bi-criteria approximation algorithm". The input contains instead of a single parameter ρ, $0 < \rho \leq 1$, two parameters ρ and ν, $0 < \nu < \rho \leq 1$. The algorithm finds a ρ-separator, and the cost of the solution is compared with the cost of an optimal ν-separator. Hence, we compare the cost of the found solution with the cost of an optimal solution to a tighter problem. We need the difference between ρ and ν in order to guarantee a diameter, as described in Lemma 27. To summarize, we find a ρ-separator whose capacity is $O\left(\frac{\rho}{\rho-\nu} \cdot \min\{\log n \log\log n, \log \tau \log\log \tau\} \cdot \tau\right)$, where τ denotes the cost of an optimal ν-separator.

In [ENRS95] we relate the ρ-separator problem to the problem of partitioning a graph into k roughly equal parts, which is called the k-multiway separator problem. Specifically, we show that if every strongly connected component contains at most $\rho \cdot n$ vertices, then every maximal grouping of strongly connected components (such that each group still contains at most $\rho \cdot n$ vertices) contains less that $2/\rho$ groups. Hence, we can compute k-multiway separators by finding a $2/k$-separator and grouping the resulting components.

The k-multiway separator problem was considered by Leighton *et al.* [LMT90] and by Simon and Teng [ST95]. They proposed applying recursively the approximate separator algorithm in [LR88] until all strongly connected components are small enough. This approach yields a k-multiway separator in which the number of vertices in each part is bounded by $2n/k$. The capacity of the separator is $O(\log n \log k \cdot \tau')$, where τ' denotes the minimum cost of a $1/k$-separator. Our algorithm is more flexible and yields a partitioning in which each part contains at most $(1+\varepsilon)n/k$ vertices, for any $\varepsilon > 0$. The capacity of the separator is $O((1/\varepsilon) \cdot \min\{\log n \log\log n, \log \tau' \log\log \tau'\} \cdot \tau')$. If we compare these results for $\varepsilon = 1$, then our algorithm is superior to the recursive algorithm when $k > \min\{\log n, \tau'^{\log\log \tau' / \log n}\}$.

We now show how to cast the directed ρ-separator problem into our paradigm. First, we need to extend the definition of the auxiliary graph to hypergraphs. The hyperedges of the auxiliary hypergraph $H = (V, E_H)$ associated with $G = (V, E)$ are defined as follows: $X \subseteq V$ is a hyperedge if $|X| > \nu \cdot n$ and the subgraph G_X of G induced by X is strongly connected. For every sub-hypergraph H_U of the hypergraph H induced by a subset of vertices U, the scaler function is defined by $\mathsf{scaler}(H_U) = 1$ if H_U contains at least one hyperedge, and 0 otherwise.

Clearly, if H_U lacks hyperedges, then all the strongly connected components in G_U are small, and hence, $\mathsf{cost}(G_U, H_U) = 0$. Divide and conquer now applies as in Section 6.5 by removing the edges of the directed cut in each divide step.

In the sequel we show that an optimal solution to the linear program given below is a spreading metric.

$$\begin{array}{lll}
\min & \sum_{e \in E} \ell(e) & \\
\text{s.t.} & \forall U \subseteq V, \forall v \in U : \sum_{u \in U} (\mathsf{dist}(v,u) + \mathsf{dist}(u,v)) \geq |U| - \nu \cdot n & \\
 & \forall e \in E : \ \ell(e) \geq 0 &
\end{array}$$

Note that the constraints for small subsets of vertices (namely, $|U| \leq \nu \cdot n$) are trivial and may be omitted.

Lemma 27: If $U \subseteq V$ satisfies $|U| > \rho \cdot n$, then for every vertex $v \in U$ there exists a vertex $u \in U$, such that

$$\mathsf{dist}(v,u) + \mathsf{dist}(u,v) > \frac{\rho - \nu}{\rho}$$

The proof of this lemma is similar to the proof of Lemma 7. Note however, that our relaxation of the diameter guarantee adds a factor of $\frac{2\rho}{\rho-\nu}$ to the approximation factor. Since we cannot guarantee a diameter of at least $\frac{\rho-\nu}{2\rho}$ unless there are at least $\rho \cdot n$ vertices, we obtain a ρ-separator rather than a ν-separator.

Lemma 28: The cost of an optimal solution of the linear program is a lower bound on the cost of an optimal ν-separator.

Proof: Consider an optimal ν-separator $F \subseteq E$. Define $\ell(e)$ to be the indicator function of F. In order to show that $\ell(e)$ is a feasible solution of the linear program, consider an arbitrary constraint for $U \subseteq V$ and $v \in U$. Define $\mathsf{comp}_F(v)$ to be the set of vertices in the strongly connected component of $G' = (V, E - F)$ that contains vertex v. Since F is a ν-separator, it follows that $|\mathsf{comp}_F(v)| \leq \nu \cdot n$. If $u \notin \mathsf{comp}_F(v)$, then either every path from u to v contains an edge from E, or every path from v to u contains an edge of E. Hence, $\mathsf{dist}(v,u) + \mathsf{dist}(u,v) \geq 1$, and

$$\begin{aligned}
\sum_{u \in U} (\mathsf{dist}(v,u) + \mathsf{dist}(u,v)) &\geq \sum_{u \in U - \mathsf{comp}_F(v)} (\mathsf{dist}(v,u) + \mathsf{dist}(u,v)) \\
&\geq |U - \mathsf{comp}_F(v)| \\
&\geq |U| - |\mathsf{comp}_F(v)| \\
&\geq |U| - \nu \cdot n
\end{aligned}$$

□

The proof of the following lemma follows the proof of Lemma 9.

Lemma 29: An optimal solution to the linear program is computable is polynomial time.

References

[AR94] Y. Aumann and Y. Rabani. "An $O(\log n)$ approximate min-cut max-flow theorem and approximation algorithm", To appear: SIAM Journal on Computing.

[BL84] S.N. Bhatt and F.T. Leighton, "A framework for solving VLSI graph layout problems", JCSS, Vol. 28, pp. 300-343, (1984).

[ENRS95] G. Even, J. Naor, S. Rao and B. Schieber, "Spreading Metric Based Approximate Graph Partitioning Algorithms", manuscript, Nov. 1995.

[ENSS95] G. Even, J. Naor, B. Schieber and M. Sudan, "Approximating minimum feedback sets and multicuts in directed graphs", *4th IPCO*, pp. 14-28, 1995. Full version appears in IBM Research Report RC 20074 (88796).

[GJ79] M.R. Garey and D.S. Johnson, "Computers and intractability: a guide to the theory of NP-completeness", W. H. Freeman, San Francisco, California, 1979.

[GVY93] N. Garg, V.V. Vazirani and M. Yannakakis, "Approximate max-flow min-(multi) cut theorems and their applications," *25th STOC,* pp. 698-707, 1993.

[Ha89] M. Hansen, "Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems," *30th FOCS*, pp. 604-609, 1989.

[KPRT93] P.N. Klein, S.A. Plotkin, S. Rao and É. Tardos, "Bounds on the Max-Flow Min-Cut Ratio for directed multicommodity flows," Unpublished manuscript, 1993.

[LMT90] F.T. Leighton, F. Makedon, and S. Tragoudas, "Approximation algorithms for VLSI partition problems", *IEEE International Symposium on Circuits and Systems*, 1990.

[LR88] F.T. Leighton and S. Rao, "An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms," *29th FOCS,* pp. 422-431, 1988. Directed graphs and applications are dealt with in manuscript, Feb., 1992.

[LLR95] N. Linial, E. London, and Y. Rabinovich. "The geometry of graphs and some of its algorithmic applications", Combinatorica, 15(2), pp. 215-245, 1995.

[NN94] Y. Nesterov and A. Nemirovskii, "Interior-point polynomial algorithms in convex programming", SIAM, Philadelphia, 1994.

[RAK91] R. Ravi, A. Agrawal and P. Klein, "Ordering problems approximated: single processor scheduling and interval graph completion", *18th ICALP*, pp. 751-762, 1991.

[RP88] G. Ramalingam and C. Pandu Rangan, "A unified approach to domination problems in interval graphs", Information Processing Letters, Vol. 27, pp. 271-274, 1988.

[S94] H. Sagan. "Space-filling curves", Springer-Verlag, 1994.

[Se95] P. D. Seymour, "Packing Directed Circuits Fractionally," Combinatorica, 15(2), pp. 281-288, 1995.

[ST95] H. D. Simon and S-H. Teng, "How good is recursive bisection", SIAM J. Scientific Computing, to appear, 1995.