Divide-and-Conquer Approximation Algorithms via Spreading Metrics

GUY EVEN

Tel-Aviv University, Tel-Aviv, Israel

JOSEPH (SEFFI) NAOR

Technion, Haifa, Israel

SATISH RAO

University of California, Berkeley, Berkeley, California

AND

BARUCH SCHIEBER

IBM, T.J. Watson Research Center, Yorktown Heights, NY

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 lengths to either edges or vertices of the input graph, such that all subgraphs for which the optimization problem is nontrivial 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

---

A preliminary version of this paper appeared in the *Proceedings of the 36th Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif., 1995, pp. 62–71.

The research of J. Naor was supported in part by Grant No. 92-00225 from the United States–Israel Binational Science Foundation (BSF), Jerusalem, Israel. His research was also supported by the Fund for the Promotion of Research at the Technion.

This work was done while S. Rao was with NEC Research Institute, 4 Independence Way, Princeton, NJ 08540.

Authors' present addresses: G. Even, Department of Electrical Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel, e-mail: guy@eng.tau.ac.il; J. Naor, Computer Science Dept., Technion, Haifa 32000, Israel, e-mail: naor@cs.technion.ac.il; S. Rao, University of California, Computer Science Division, Soda Hall, Berkeley, CA 94720-1776, e-mail: satish@cs.berkeley.edu; B. Schieber, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, e-mail: sbar@watson.ibm.com.

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) balanced partitions and ρ-separators (for small values of ρ) in directed graphs.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Program Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*; *network problems*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation algorithms, divide and conquer, feedback set, linear arrangement, multicut, spreading metrics

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, which 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 in which the optimization problem is nontrivial 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 a spreading metric of a problem instance provides a lower bound on the cost of solving the problem. A spreading metric allows us to apply divide-and-conquer in which the divide step is guided by the *volumes* of the subproblems 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 [1999] 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 this framework was 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 Leighton and Rao [1999]. The idea is that spreading metrics can be used to give an estimate of the recursive cost of solving subproblems that arise as a result of applying divide-and-conquer. 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 an additional doubly logarithmic factor) for both the recursion and the error in the cut procedure. This generalizes the approach taken in by Seymour [1995] in proving the existence of small feedback sets in directed unweighted graphs. We note that a spreading metric is computed only once, and in all the recursive calls we use the same spreading metric.

Recently, in Even et al. [1999], we were able to extend our spreading metrics technique to graph partitioning problems. We were able to apply a simpler recursion and thus obtain simple logarithmic factor approximation algorithms for balanced partitions and separators. However, the simpler recursion does not apply to the applications described in this paper.

1.1. RESULTS. We consider problems that are amenable to a divide-and-conquer approach and for which 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. This parameter τ is a lower bound on the cost of the optimal solution and is obtained in this paper by solving a linear program. The second parameter is k, the number of "interesting" vertices. This is 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 $O(\min\{\log \tau \log\log \tau, \log k \log\log k\})$.

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 improves upon 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) q-balanced partitions 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 $O(\min\{\log \tau \log\log \tau, \log n \log\log n, \log^2 k)$ to $O(\min\{\log \tau \log\log \tau, \log k \log\log k\})$, where k denotes the size of the subset in the subset feedback set problems or the number of source-sink pairs in the multicut problems. For Problem (7), let τ' denote the optimal cost of a ν-separator where $\nu = 1/q$ in the case of q-balanced partitions or $\nu = \rho - \varepsilon$, for some fixed ε, in the ρ-separator case. Our approximation algorithm finds a separator whose capacity is $O(\min\{\log n \log\log n, \log \tau' \log\log \tau'\} \cdot \tau')$. This improves on previous results for the cases in which q is big enough, or when ρ is small enough.

For some of the applications, for example, subset feedback set in a directed graph, a spreading metric can be obtained from the "natural" linear programming relaxation of the problem. In fact, this type of spreading metrics draws directly from past work on multi-commodity flow and feedback sets in directed graphs, for example, Leighton and Rao [1999], Seymour [1995], and Garg et al.

[1] Indeed, 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.

[1996]. However, for other problems, for example, linear arrangement and interval graph completion, computing a spreading metric involves an "indirect" linear programming formulation. The formulation is indirect in the sense that it captures certain properties of an optimal (integral) solution, yet an integral solution (to the formulation) does not provide an integral solution to the original problem, as opposed to standard linear programming relaxations. Nevertheless, the spreading metric obtained from the linear programming formulation still has all the required properties.

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 [1999] presented an $O(\log n)$ approximation algorithm for finding balanced partitions of graphs. Their algorithm is based on an approximation algorithm for a sparse cut, namely, a cut which approximately minimizes the ratio of the weight of the edges in the cut divided by the number of vertices in the smaller side of the cut. Some of the results that follow from this are: a basis for an algorithmic implementation of the decomposition tree framework of Bhatt and Leighton [1984], 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 [1989]. His algorithms rely on the algorithm of Leighton and Rao [1999] to obtain $O(\log^2 n)$ approximation algorithms for these problems. We present a novel linear programming formulation for these problems that enables us to use our paradigm to improve the approximation factor for these problems to $O(\log n \log\log n)$. Ravi et al. [1991] considered a generalization of the linear arrangement problem, called the storage-time product minimization problem. In case the execution time of all tasks is the same, their algorithm achieves an $O(\log^2 n)$ approximation factor. Again, our algorithm improves this factor to $O(\log n \log\log n)$.

The problem of finding a super-graph of a given graph, such that the super-graph is an interval graph and contains as few edges as possible, was considered by Ravi et al. [1991]. 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 Leighton and Rao [1999], Seymour [1995], Klein et al. [1997], and Even et al. [1998]. Seymour [1995] was the first to present a decomposition algorithm that does not rely on balanced cuts. His paper proves the existence of feedback vertex sets of cardinality $O(\tau \log \tau \log\log \tau)$ in unweighted directed graphs. Klein et al. [1997] considered symmetric multicuts in directed graphs. By extending the work of Garg et al. [1996] to the directed case, they obtained an $O(\log^2 k)$ approximation factor for this problem. Even et al. [1998] considered the subset feedback set problem in directed graphs, as well as multicuts in circular networks. In the subset feedback set problem, the feedback set is only required to intersect a

subset of the directed cycles in the graph, characterized by a distinguished subset of the vertices called *special* vertices. 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 [1995], and the last term was obtained by extending the work of Garg et al. [1996]. 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 work for small values of k.

The *ρ-separator* problem in undirected (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 Even et al. [1999], where a logarithmic approximation factor algorithm is described for the undirected case, and 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 q-balanced partitions 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 q 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 q-balanced partition) by applying recursively the approximate separator algorithm in Leighton and Rao [1999] until all strongly connected components are small enough. (See also Leighton et al. [1990] and Simon and Teng [1997].) 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 $\nu < \max\{1/\log n, \tau'^{-\log\log \tau'/\log n}\}$. (A similar improvement is achieved for the q-balanced partitioning problem as detailed in Section 6.)

The recent papers of Linial et al. [1995] and Aumann and Rabani [1998] also consider metrics on graphs. They regard graphs with edge lengths as geometric objects and embed them in a high dimensional space with a logarithmic distortion of the edge lengths. The dimension of the space 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 nontrivial subproblem 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. For simplicity, we define the paradigm for undirected graphs, and consider the edge version. We later show how to extend the paradigm to directed graphs and to the vertex version.

2.1. What Kind of Problems Can We Approximate?

The paradigm finds approximate solutions of minimization problems on undirected graphs with edge capacities. A problem instance is represented by a pair of graphs (G, H), where $G = (V, E, c)$ 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 as G. The auxiliary graph is often a clique, for example, the linear arrangement problem, or a (partial) matching between pairs of terminals, for example, multicuts in directed circular networks.

Depending on the problem, we define a real-valued function scaler over the subgraphs of H. Let G' and H' denote subgraphs of G and H, respectively, defined over the same vertex set. We require that if H' contains edges, then $\text{scaler}(H') \geq 1$, otherwise $\text{scaler}(H') = 0$. The graph H and the scaler function have the following roles:

(1) If H' lacks edges (i.e., $\text{scaler}(H') = 0$), then the subproblem (G', H') is trivial (i.e., can be solved with zero cost).
(2) $\text{scaler}(H')$ is used in quantifying the cost of dividing the problem (G', H') into subproblems, as discussed in the paragraph on divide-and-conquer applicability.
(3) $\text{scaler}(H')$ serves as a lower bound on the diameter of G' induced by a spreading metric, as discussed in the paragraph on spreading metric applicability.

In all the applications discussed herein, the scaler function depends only on the size of the largest connected component in H (e.g., $\text{scaler}(H)$ equals the size of the largest connected component in H).

We can approximate problems that satisfy two properties: divide-and-conquer applicability and spreading metric applicability, defined below.

2.1.1. *Divide-and-Conquer Applicability*.

In general, divide-and-conquer applicability means that a problem can be solved by dividing it into subproblems, solving each subproblem independently, and then merging the solutions to obtain a solution for the original problem. For our paradigm we also require that the cost of merging the solutions is proportional to capacities of cuts in G as defined below. The definition of divide-and-conquer applicability in this paper is based on a decomposition tree representation, where a decomposition tree specifies a set of solutions to a problem instance. There is a cost associated with a decomposition tree which is an upper bound on the cost of every solution specified by it. We define a decomposition tree as follows:

Definition 1. A *decomposition tree* T of a graph $G(V, E)$ is a tree in with the following properties.

(1) Every tree node corresponds to a subset of the vertices V. Denote the subset of vertices corresponding to a tree node $t \in T$ by V_t.
(2) The subset of vertices corresponding to the tree root $r \in T$ is $V_r = V$.
(3) For every tree node $t \in T$, the set V_t is partitioned by the sets $V_{t_1}, V_{t_2}, \ldots, V_{t_d}$, where $t_1, \ldots, t_d$ denote the children of t.

Let $t \in T$ denote a node in a decomposition tree T. Let G_t and H_t denote the subgraphs of G and H, respectively, that are induced by V_t. We now define what it means that a decomposition tree T *fully decomposes* the problem represented by (G, H).

Definition 2. A decomposition tree T of $G(V, E)$ *fully decomposes* the problem (G, H) if for every leaf $t \in T$, the problem (G_t, H_t) is trivial.

Note that the translation of a decomposition tree that fully decomposes a problem instance to an actual solution of the problem instance depends on how solutions for subproblems are merged to a solution for the whole problem. This varies from one problem to another. For example, in the linear arrangement problem, a solution induced by a decomposition tree is an ordering of the vertices in their order of appearance as leaves of the decomposition tree. In all our applications, this translation is simple and can be computed in polynomial time. We elaborate more on this point in Section 6.

Every internal node $t \in T$ in the decomposition tree defines a cut in G as follows: Let $t_1, \ldots, t_d$ denote the children of t. The *cut corresponding to* t is the set of edges $F_t \subseteq E$ that have endpoints in different subsets V_{t_i} and V_{t_j}. Let $c(F_t)$ denote the sum of the edge capacities in F_t.

For our approximation paradigm to apply, we require that the cost of merging solutions of the subproblems represented by $(G_{t_1}, H_{t_1}), \ldots, (G_{t_d}, H_{t_d})$ to a solution of the problem (G_t, H_t) depends on $c(F_t)$ scaled by the coefficient $\mathsf{scaler}(H_t)$. The sum of all these costs constitutes the cost of T as defined below.

Definition 3. Let T denote a decomposition tree that fully decomposes a problem (G, H). The *cost* of T equals the sum of the scaled capacities of the cuts corresponding to the internal decomposition tree nodes. Formally,

$$\text{cost}(T) = \sum_{t \in T} \text{scaler}(H_t) \cdot c(F_t).$$

The cost of a decomposition tree that fully decomposes a problem instance bounds the cost of every solution of the problem instance that is built up from the decomposition tree. Thus, if we can obtain a low-cost decomposition tree that fully decomposes a problem instance, then we can also find low-cost solutions of the problem instance.

2.1.2. *Spreading Metric Applicability*. Spreading metric applicability means that for the problem (G, H) there exists a spreading metric, defined as follows:

Definition 4. A *spreading metric* is a function $\ell: E \to \mathsf{R}^+$ that assigns nonnegative lengths to every edge $e \in E$. This function must satisfy the following two properties:

```
divide&conquer (G, H, ℓ)
begin
    If E_H is empty, then T = φ
    else /* divide */
        (V_1, V_2) = partition(G, H, ℓ)
        For i = 1, 2 do
            Let G_i and H_i denote the subgraphs of G and H induced by V_i.
            T_i = divide&conquer(G_i, H_i, ℓ_i)
        Let T denote a tree with a root t such that:
            V_t = V(G); and
            t has two children: the root of T_1 and the root of T_2.
    return(T)
end

generic_approximation_alg (G, H)
begin
    Compute a spreading metric ℓ : E → Q.
    T = divide&conquer(G, H, ℓ)
    Return(T)
end
```

FIG. 1. The generic approximation algorithm.

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

Diameter guarantee: The distance induced by the spreading metric "spreads" the graph and all its subgraphs that correspond to nontrivial subproblems. More precisely, for vertices u and v, let $dist_\ell(u, v)$ 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 requires that for every subset $U \subseteq V$, there exist two vertices $u, v \in U$ for which $dist_\ell(u, v) \geq scaler(H_U)$.

2.2. THE GENERIC APPROXIMATION ALGORITHM. The generic approximation algorithm is depicted in Figure 1. The partitioning procedure *partition* is described in Section 3. First, a spreading metric for (G, H) is computed. Then, a divide-and-conquer algorithm partitions the graph recursively until the subproblems defined by the resulting subgraphs are trivial. The generic approximation computes a decomposition tree that is translated to a solution of the problem.

2.3. THE MAIN RESULT. The main result presented in this paper is the following theorem:

THEOREM 1. *Let (G, H) denote an optimization problem that satisfies the paradigm defined in Section* 2.1. *If a spreading metric is computable in polynomial time, then the generic approximation algorithm finds, in polynomial time, a decomposition tree that fully decomposes the problem (G, H), the cost of which is $O(\tau \min\{\log \tau \log\log \tau, \log k \log\log k\})$, where τ is the cost of the spreading metric (and hence a lower bound on the cost of an optimal solution), and $k \leq n$ is the number of nonisolated vertices in H (i.e., vertices with positive degree).*

Recall that the cost of a decomposition tree that fully decomposes a problem instance is an upper bound on the cost of all the solutions that are specified by the decomposition tree. Moreover, a solution specified by a decomposition tree can be computed in polynomial time. The following corollary follows from Theorem 1.

COROLLARY 2. *Under the premises of Theorem* 1, *the generic approximation algorithm finds in polynomial time a solution of problem* (G, H), *the cost of which is* $O(\tau \min\{\log \tau \log\log \tau, \log k \log\log k\})$.

2.4. EXTENSIONS: VERTEX VERSION AND DIRECTED GRAPHS

(1) 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. The vertex version is applied in Section 6.4.

(2) The paradigm can be applied to directed graphs as well (see Sections 6.5–6.7). The modification for the directed case is that partitioning in the directed case means dividing the graph into strongly connected components instead of connected components. This means that a directed cut is attached to each divide step rather than a undirected cut.

(3) When the auxiliary graph is directed as well, the radius guarantee refers to distances (measured in G) between two vertices: a "source" and a "sink" in the auxiliary graph.

2.5. 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 bounded from above 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 $b > 1/2$. In Section 5, we extend our algorithms to deal with balance constraints.

(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 $\Sigma_{e \in E_U} c(e)\ell(e) \geq 1$. Therefore, the volume of a spreading metric of a non-trivial problem is at least 1.

(3) Suppose that the volume of the spreading metric satisfies the lower bound property, however, the diameter guarantee only satisfies $\mathsf{dist}_\ell(u, v) \geq \alpha \cdot \mathsf{scaler}(H_U)$, where $\alpha \leq 1$ is some parameter (constant or nonconstant). The approximation factor obtained in this case is multiplied by a factor of $1/\alpha$, since the spreading metric can be scaled by a factor of $1/\alpha$, yielding that the volume of the spreading metric (or lower bound guarantee) is also multiplied by a factor of $1/\alpha$.

3. *Partitioning High-Diameter Graphs*

The main step in the divide-and-conquer procedure is the partitioning step. Loosely speaking, the partitioning procedure searches for a subset of nodes U that satisfies two properties: (a) the volume associated with U is at most half the volume of G; and (b) the capacity of the cut $(U, V - U)$ is comparable with the volume associated with U. These two properties are essential for analyzing the cost of the solution, since one can charge the cost of separating U from the rest of the graph to the volume associated with U. We note that edge lengths remain the same during the recursive calls to the partitioning procedure.

3.1. General Setting and Notation.

Let $G = (V, E)$ denote a (connected) graph with edge capacities $c(e) \geq 1$ for all $e \in E$. Given a subset $U \subset V$, let $c(U)$ denote the capacity of the cut $(U, V - U)$. Let $H = (V, E_H)$ denote the auxiliary graph corresponding to G, and let $\ell(e)$ denote the spreading metric that assigns non-negative edge lengths. Let $\mathsf{vol}(G)$ denote the volume $\Sigma_{e \in E} c(e) \cdot \ell(e)$. Let $u, u' \in V$ be two vertices that belong to the same connected component in the auxiliary graph H, and for which the diameter property guarantees $\Delta \triangleq \mathit{dist}_\ell(u, u') \geq \mathsf{scaler}(H)$.

Define $N(u, r)$ by

$$N(u, r) \triangleq \{v \in V : \mathsf{dist}_\ell(u, v) < r\}$$

Note the strict inequality in the definition on $N(u, r)$, namely, a vertex v of distance r from u is not in $N(u, r)$.

Denote by $E(u, r)$ the set of edges for which both endpoints belong to $N(u, r)$, and by $\Gamma(u, r)$ 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:

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

If $\mathsf{vol}(u, \Delta/2) > \mathsf{vol}(G)/2$, then we reverse the roles of u and u' and consider $N(u', \Delta/2)$. Since $N(u, \Delta/2)$ and $N(u', \Delta/2)$ are disjoint, it follows that $\mathsf{vol}(u', \Delta/2) \leq \mathsf{vol}(G)/2$. Hence, we may assume without loss of generality that $\mathsf{vol}(u, \Delta/2) \leq \mathsf{vol}(G)/2$.

3.2. The Partitioning Procedure.

The partitioning procedure grows a region $N(u, r)$ using a single-source shortest-path algorithm until the capacity of the cut $\Gamma(u, r)$ is bounded by an expression justified by Theorem 3. The partitioning procedure is depicted in Figure 2.

3.3. Existence of a Good Cut.

We now prove a theorem that shows the existence of a good cut. Loosely speaking, a corollary to this theorem is that the partitioning procedure stops with a radius $r \leq \Delta/2$.

Theorem 3. *Let $G = (V, E)$ denote a graph with edge capacities $c(e) \geq 1$ and nonnegative edge lengths $\ell(e)$. Let u and u' in V be two vertices such that $\mathsf{dist}_\ell(u, u') = \Delta$. For any r_0, r_1 satisfying $0 < r_0 < r_1 < \Delta$, there exists a radius*

partition (G, H, ℓ)
begin
  Choose a pair of vertices u, u' such that:
    1. u and u' belong to the same connected component in H;
    2. $\Delta = \mathsf{dist}_\ell(u, u') \geq \mathsf{scaler}(H)$; and
    3. $\mathsf{vol}(u, \Delta/2) \leq \mathsf{vol}(G)/2$
  $U = N(u, \Delta/4)$ *(set the initial region)*
  $v =$ closest vertex to U in $V - U$.
  let $r = \mathsf{dist}_\ell(u, v)$ *(set initial radius)*
  while $c(U) > \frac{4}{\Delta} \cdot \mathsf{vol}(u, r) \ln\left(\frac{e\mathsf{vol}(G)}{2\mathsf{vol}(u,r)}\right) \cdot \ln\ln\left(\frac{e\mathsf{vol}(G)}{2\mathsf{vol}(u,\Delta/4)}\right)$ do
    begin *(grow the region)*
    $U = U \cup \{v\}$
    $v =$ closest vertex to U in $V - U$.
    $r = \mathsf{dist}_\ell(u, v)$ *(update radius)*
    end
  Return $(U, V - U)$
end

FIG. 2. The partitioning procedure.

$r_0 \leq r \leq r_1$ *such that*

$$c(N(u, r)) \leq \frac{\mathsf{vol}(u, r)}{r_1 - r_0} ln\left(\frac{e \cdot \mathsf{vol}(u, r_1)}{\mathsf{vol}(u, r)}\right) ln\, ln\left(\frac{e \cdot \mathsf{vol}(u, r_1)}{\mathsf{vol}(u, r_0)}\right). \tag{1}$$

(*The base of all logarithms is e.*)

PROOF. We prove the theorem using two lemmata: The first lemma, that follows Seymour [1995] and was used in a discrete fashion by Leighton and Rao [1999] shows that, except for at most $n - 1$ points, the derivative of the volume function, $d\mathsf{vol}(u, r)/dr$, equals $c(N(u, r))$. The second lemma, along the lines of Seymour [1995] shows that there exists a radius $r \in (r_0, r_1)$ for which $d\mathsf{vol}(u, r)/dr$ is not greater than the right-hand side of Eq. (1).

LEMMA 4. *Let* $I = \{\mathsf{dist}_\ell(u, v)\}_{v \in V}$. *Then for every* $r \in [r_0, r_1] - I$, *the derivative* $d\mathsf{vol}(u, r)/dr$ *is defined and satisfies*

$$\frac{d\mathsf{vol}(u, r)}{dr} = c(N(u, r)). \tag{2}$$

Lemma 4 follows by observing that in an interval (r', r'') in which no new vertices are encountered (i.e., $N(u, r') = N(u, r'')$), the function $\mathsf{vol}(u, r)$ is linear.

LEMMA 5. *Let* f: $[r_0, r_1] \to \mathsf{R}$ *be a nonnegative monotone increasing function that is differentiable almost everywhere. If the derivative* f' *is continuous almost everywhere, then there exists an* $r \in (r_0, r_1)$ *such that* $f'(r)$ *is defined and satisfies*

$$f'(r) \leq \frac{f(r)}{r_1 - r_0} \cdot ln\left(\frac{e \cdot f(r_1)}{f(r)}\right) \cdot ln\ ln\left(\frac{e \cdot f(r_1)}{f(r_0)}\right). \tag{3}$$

PROOF. By contradiction, suppose that for every $r \in (r_0, r_1)$

$$f'(r) > \frac{f(r)}{r_1 - r_0} \cdot \ln\left(\frac{e \cdot f(r_1)}{f(r)}\right) \cdot \ln\ln\left(\frac{e \cdot f(r_1)}{f(r_0)}\right).$$

Then,

$$\frac{f'(r)}{f(r) \cdot \ln\left(\frac{e \cdot f(r_1)}{f(r)}\right)} > \frac{1}{r_1 - r_0} \cdot \ln\ln\left(\frac{e \cdot f(r_1)}{f(r_0)}\right). \tag{4}$$

We now integrate both sides over the interval (r_0, r_1). (This can be done since f' is continuous almost everywhere.) Since the right-hand side does not depend on r, we get that after integration the right hand side satisfies:

$$\frac{1}{r_1 - r_0} \cdot \ln\ln\left(\frac{e \cdot f(r_1)}{f(r_0)}\right) \cdot \int_{r_0}^{r_1} dr = \ln\ln\left(\frac{e \cdot f(r_1)}{f(r_0)}\right).$$

To integrate the left-hand side, one may verify by differentiation that

$$\int \frac{f'(r)}{f(r) \cdot \ln\left(\frac{e \cdot f(r_1)}{f(r)}\right)} dr = -\ln\ln\left(\frac{e \cdot f(r_1)}{f(r)}\right).$$

Therefore,

$$\int_{r_0}^{r_1} \frac{f'(r)}{f(r) \cdot \ln\left(\frac{e \cdot f(r_1)}{f(r)}\right)} dr = -\ln\ln\left(\frac{e \cdot f(r_1)}{f(r)}\right)\Bigg|_{r_0}^{r_1}$$

$$= \ln\ln\left(\frac{e \cdot f(r_1)}{f(r_0)}\right).$$

The contradiction now follows, since before we integrated both sides of Eq. (4), the left-hand side was greater than the right-hand side almost everywhere. However, after integration both sides are equal. The lemma follows. □

The proof of Theorem 3 follows by considering the function $f(r) = \mathsf{vol}(u, r)$ and applying the two lemmata. □

3.3.1. *The Correctness of the Partitioning Algorithm*. Theorem 3 proves the existence of a "good" radius in the interval $(\Delta/4, \Delta/2)$. This follows by assigning $r_0 = \Delta/4$, $r_1 = \Delta/2$, and the assumption that $\mathsf{vol}(u, \Delta/2) \leq \mathsf{vol}(G)/2$. However, the partitioning algorithm only considers radii from the set $I = \{\mathsf{dist}_\ell(u, v)\}_{v \in V}$. To prove that the partitioning procedure stops with a "good" radius, we need to show that I contains a "good" radius.

LEMMA 6. *There exists a radius $r' \in I$ that satisfies Eq.* (1).

PROOF. Let $r \in (\Delta/4, \Delta/2)$ be the radius that satisfies Eq. (1). The existence of such a radius is guaranteed by Theorem 3. Let r' denote the result of rounding the radius r up to the nearest radius in I. Note that $N(u, r') = N(u, r)$, and therefore, $c(N(u, r')) = c(N(u, r))$. Since the function $x \ln(ea/x)$ is increasing in the interval $(0, a]$, the radius r' also satisfies Eq. (1). □

Recall that the analysis of the approximation factor relies on the assumption that the cut is defined by a sphere $N(u, r)$ for which $\mathsf{vol}(u, r) \leq \mathsf{vol}(G)/2$. Since the partitioning procedure only considers radii from the set I, it may happen that the radius r' returned by the procedure is greater than $\Delta/2$. (This would happen when $r \in (\Delta/4, \Delta/2)$ is rounded up to $r' > \Delta/2$.) In this case it is not guaranteed that $\mathsf{vol}(u, r') \leq \mathsf{vol}(G)/2$. The following lemma shows that if $r' > \Delta/2$, then using a radius of $\Delta/2$ would result with the same cut. Therefore, either $\mathsf{vol}(u, r') \leq \mathsf{vol}(u, \Delta/2) \leq \mathsf{vol}(G)/2$ or we may use (only) in the analysis $r' = \Delta/2$.

LEMMA 7. *If the procedure returns a radius* $r' > \Delta/2$, *then* $N(u, r') = N(u, \Delta/2)$.

PROOF. Recall that r' is the result of rounding a radius $r \leq \Delta/2$ up to a radius $r' \in I$. Hence, for all $r \leq t \leq r$, $N(u, t) = N(u, r')$. □

4. *Analysis of the Approximation Factor*

In this section, we prove Theorem 1. We show that the cost of the decomposition tree T computed by the generic algorithm is $O(\tau \min\{\log \tau \log\log \tau, \log k \log\log k\})$, where τ is the cost of the spreading metric (and hence a lower bound on the cost of an optimal solution), and $k \leq n$ is the number of nonisolated vertices in H (i.e., vertices with positive degree). The proofs of the two approximation factors are given separately below.

4.1. AN $O(\log \tau \log\log \tau)$ APPROXIMATION FACTOR. The analysis in this case follows the analysis by Seymour [1995]. Consider an internal tree node $t \in T$. The cut F_t partitions the vertex subset V_t into V_ℓ and V_r, where ℓ and r denote the children of t. Let $\mathsf{vol}(G_t)$ denote the volume of the subgraph G_t according to the spreading metric computed in the first step. Recall that the cut F_t is computed by choosing a nonisolated vertex u in H_t and growing a sphere of radius r' in G_t centered at u. Let vol_ℓ denote the volume $\mathsf{vol}(u, r'')$, where $r'' = \min\{r', \Delta/2\}$. Lemma 7 implies that $V_\ell = N(u, r'')$. Note that vol_ℓ includes, in addition to the volume of the subgraph of G_t induced by V_ℓ, also a contribution from the edges of the cut F_t. The assumption that $\mathsf{vol}(u, \Delta/2) \leq \mathsf{vol}(G)/2$ implies that $\mathsf{vol}_\ell \leq \mathsf{vol}(G)/2$.

The cost associated with an internal tree node $t \in T$ is $\mathsf{scaler}(H_t) \cdot c(F_t)$. Since $\mathsf{scaler}(H_t) \leq \Delta$, Theorem 3 implies that

$$\mathsf{scaler}(H_t) \cdot c(F_t) \leq 4\mathsf{vol}_\ell \ln\left(\frac{e \cdot \mathsf{vol}(G_t)}{2\mathsf{vol}_\ell}\right) \ln\ln\left(\frac{e \cdot \mathsf{vol}(G_t)}{2\mathsf{vol}(u, \Delta/4)}\right).$$

Since the capacity of every edge is at least one and since $\Delta \geq 1$, it follows that $\mathsf{vol}(u, \Delta/4) \geq 1/4$. Therefore, the "ln ln" factor is bounded by $O(\log\log$

$\mathsf{vol}(G)$). For simplicity, we henceforth ignore this "ln ln" factor and take it into account only at the end of the analysis.

Let $f(x)$ denote the maximum cost of a decomposition tree of a problem (G, H) for which $\mathsf{vol}(G) \leq x$. From Theorem 3, it follows that the function $f(x)$ satisfies the following recurrence equation

$$f(x + y) \leq f(x) + f(y) + 4x \ln\left(\frac{e(x + y)}{2x}\right),$$

where $1/4 \leq x \leq y$. Remark 2 in Section 2.5 implies that if $x < 1$, then $f(x) = 0$.

Define the function $F(x)$ over the interval $[1/4, \infty)$ as follows:

$$F(x) \triangleq \frac{1}{\ln(2)} \cdot 4x \ln(4x).$$

Note that for $1/4 \leq x \leq y$

$$F(x + y) - F(x) - F(y) \geq \frac{1}{\ln(2)} \cdot 4x \ln\left(\frac{x + y}{x}\right).$$

To show that the function $F(x)$ is a solution of the recurrence equation, we only need to verify that

$$\frac{1}{\ln(2)} \cdot 4x \ln\left(\frac{x + y}{x}\right) \geq 4x \ln\left(\frac{e(x + y)}{2x}\right).$$

This follows since

$$\begin{aligned} \ln\left(\frac{e}{2}\right) &= \left(\frac{1}{\ln 2} - 1\right) \ln 2 \\ &\leq \left(\frac{1}{\ln 2} - 1\right) \ln\left(\frac{x + y}{x}\right). \end{aligned}$$

We conclude that the generic approximation algorithm finds a decomposition tree the cost of which is bounded by $F(\mathsf{vol}(G)) \cdot \ln \ln(2e \cdot \mathsf{vol}(G)) = O(\mathsf{vol}(G) \cdot \log(\mathsf{vol}(G)) \cdot \log \log(\mathsf{vol}(G)))$. Since $\mathsf{vol}(G)$ is a lower bound on the cost of an optimal solution to the problem (G, H), the first approximation factor follows.

4.2. An $O(\log k \log \log k)$ Approximation Factor. The generic approximation algorithm needs to be tuned to obtain the $O(\log k \log \log k)$ approximation factor. There are two modifications:

(1) The definition of the volume used for partitioning the graph. We assign a weight of $\mathsf{vol}(G)/k$ to every vertex that is nonisolated in H (all other vertices

are assigned zero weight). The modified volume of a sphere $N(u, r)$ is defined by

$$\mathsf{vol}'(u, r) = \mathsf{vol}(u, r) + \sum_{v \in N(u,r)} w(v).$$

Note that $\mathsf{vol}'(G) = 2\mathsf{vol}(G)$.

(2) The range of radii in which the cut is searched for is set to $[0, \Delta/2]$ (as opposed to $[\Delta/4, \Delta/2]$ in the previous analysis).

Let $t \in T$ denote an internal tree node. Using the same notation used for the first approximation factor, the implication of these changes is that the cost associated with the cut F_t is

$$\mathsf{scaler}(H_t) \cdot c(F_t) \leq 2\mathsf{vol}'_\ell \ln\left(\frac{e \cdot \mathsf{vol}'(G_t)}{2\mathsf{vol}'_\ell}\right) \ln\ln\left(\frac{e \cdot \mathsf{vol}'(G_t)}{2\mathsf{vol}(G)/k}\right).$$

We now bound the ln ln factor as follows:

$$\ln\ln\left(\frac{e \cdot \mathsf{vol}'(G_t)}{2\mathsf{vol}(G)/k}\right) \leq \ln\ln(ke).$$

Again, we ignore the "ln ln" factor till the end of the analysis.

Let $\hat{f}(x)$ denote the maximum cost of a decomposition tree of a problem (G, H) for which $\mathsf{vol}'(G) \leq x$. The function $\hat{f}(x)$ satisfies the following recurrence equation

$$\hat{f}(x + y) \leq \hat{f}(x) + \hat{f}(y) + 2x \ln\left(\frac{e(x + y)}{2x}\right),$$

where $\mathsf{vol}(G)/k \leq x \leq y$. Note that if $x < \mathsf{vol}(G)/k$, then $\hat{f}(x) = 0$ since the subgraph lacks nonisolated vertices.

Define the function $\hat{F}(x)$ over the interval $[\mathsf{vol}(G)/k, \infty)$ as follows:

$$\hat{F}(x) \triangleq \frac{1}{\ln 2} \cdot 2x \ln\left(\frac{x}{\mathsf{vol}(G)/k}\right).$$

The proof that the function $\hat{F}(x)$ is a solution of the recurrence equation bounding the growth of the function $\hat{f}(x)$ follows the proof of the first approximation factor.

We conclude that the generic approximation algorithm finds a decomposition tree the cost of which is bounded by $\hat{F}(\mathsf{vol}'(G)) \cdot \ln\ln(ke) = O(\mathsf{vol}(G) \cdot \log k \cdot \log\log k)$, and the second approximation factor follows.

5. *Balancing Decomposition Trees*

In this section, we show how to derive a balanced decomposition tree from an unbalanced decomposition tree with only a constant multiplicative factor increase in the cost of the tree. Thus, in the case where only a balanced decomposition tree can specify a solution to the problems at hand, we can first find an unbalanced decomposition tree, and then balance it. To simplify the

exposition, we assume that the auxiliary graph H is a clique graph on V and that $\mathsf{scaler}(H')$ depends only on the number of vertices in H'. We assume further that $\mathsf{scaler}(H')$ is monotone non-decreasing and bounded by a polynomial in the number of vertices in H'. All these assumptions hold for our applications (e.g., graph embeddings in d-dimensions).

Definition 5. Let $n(t)$ denote $|V_t|$ for an internal tree node t. A tree node t is *balanced* if for every child t_i of t, $n(t_i) \leq (2/3) \cdot n(t)$. A tree T is *balanced* if every internal tree node $t \in T$ is balanced.

Given an unbalanced tree T, we construct a balanced tree T' from T. Loosely speaking, the balanced tree T' is constructed from T in a top-down fashion, namely, we balance the tree nodes from the root to the leaves. An internal tree node t is balanced by cutting off "small" subtrees from "large" subtrees rooted at children of t. The cut off subtrees are then "promoted" and become children of t. More formally, a subtree rooted at t_i is small if $n(t_i) \leq (2/3) \cdot n(t)$ and large otherwise. The balancing of an internal tree node t proceeds by traversing large subtrees rooted at children of t in preorder (i.e., first the node and then recursively its children). As soon as a small subtree is reached, this small subtree is promoted to become a child of t. A traversal of a large subtree and cutting off of small subtrees from its continues until this subtree becomes small. After the balancing procedure is applied to t, it is applied to the children of t. We note that: (a) When the balancing procedure is applied to t, $n(t)$ does not change. However, $n(t)$ may decrease when the balancing procedure is applied to an ancestor of t. (b) The balanced tree T' may not be a binary tree even if T is. If a balanced and binary decomposition tree is sought, then children t_1, t_2 of $t \in T'$ for which $n(t_1) \leq n(t_2) \leq (1/3)n(t)$ may be coalesced to make T' a binary tree without rendering T' unbalanced. The set of children of the tree node obtained by coalescing t_1 and t_2 is the union of the children of t_1 and t_2. Such a coalescing of siblings can be applied in preorder to make T' a balanced binary tree.

Our assumption that the auxiliary graph H is a clique spanning all the vertices, implies that every edge belongs to a cut in the decomposition tree, namely, $\bigcup_{t \in T} F_t = \bigcup_{t' \in T'} F_{t'} = E$. For an edge $e \in E$, let $t(e)$ denote the tree node in T whose cut contains e. Namely, $t(e)$ is the least common ancestor in T of the endpoints of 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 8. *For every edge* $e \in E$, $\frac{2}{3} \cdot n(t'(e)) < n(t(e))$.

PROOF. Fix an edge $e = (u, v)$ and assume that the leaf corresponding to u precedes the leaf corresponding to v in preorder traversal. If $n(t'(e))$ in the beginning of the balancing of $t'(e)$ is not greater than $n(t(e))$ (in T), then $n(t'(e)) \leq n(t(e))$ also after the balancing of T, and the lemma follows.

Suppose that at the beginning of the balancing of $t'(e)$, $n(t'(e)) > n(t(e))$. Figure 3 depicts this situation. Since the subtree rooted at $t'(e)$ at the beginning of the balancing of $t'(e)$ is isomorphic to a subtree of T, and since u and v are leaves of this subtree, it follows that $t(e)$ is a descendent of $t'(e)$ at the beginning of the balancing of $t'(e)$. Let t_u denote a descendent of $t(e)$ that is promoted to become a child of $t'(e)$ for which $u \in V_{t_u}$. Note that such a node must exist since the subtree rooted at $t(e)$ is large (otherwise, $t'(e)$ would have been the same as $t(e)$), and u precedes v in the preorder traversal that starts at $t(e)$. Since $t(e)$

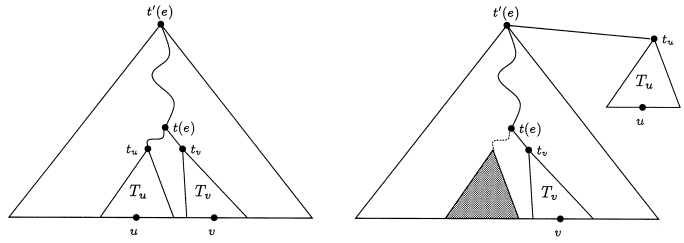

FIG. 3. The effect of balancing $t'(e)$, where $e = (u, v)$. (a) The subtree rooted at $t'(e)$ before balancing $t'(e)$; (b) The subtree rooted at $t'(e)$ after t_u is "promoted."

was not promoted, it follows that $n(t(e)) > (2/3) \cdot n(t'(e))$, and the claim follows. □

By our assumptions $\mathsf{scaler}(H')$ depends only on the number of vertices in H'. Thus, the function scaler can be viewed as a function defined over the natural numbers. Since $\mathsf{scaler}(x)$ is assumed to be monotone nondecreasing and bounded by a polynomial, the value

$$\beta \triangleq \sup_{x \in \mathbf{N}+} \left\{ \frac{\mathsf{scaler}(x)}{\mathsf{scaler}(\lceil (2/3) \cdot x \rceil)} \right\}$$

is a well-defined constant. 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 9. *Assume that*: (*a*) *the auxiliary graph H is a clique*; *and* (*b*) $\mathsf{scaler}(H')$ *depends only on the number of vertices in H' and is monotone nondecreasing and bounded by a polynomial in the number of vertices in H'.*

Let T denote a decomposition tree and let T' denote the balanced decomposition tree obtained by the balancing procedure. Then

$$\mathsf{cost}(T') \leq \beta \cdot \mathsf{cost}(T).$$

PROOF. Lemma 8, the fact that $\mathsf{scaler}(\cdot)$ is monotone non-decreasing, and the definition of β, imply that for every edge $e \in E$,

$$\frac{\mathsf{scaler}(n(t'(e)))}{\mathsf{scaler}(n(t(e)))} \leq \frac{\mathsf{scaler}(n(t'(e)))}{\mathsf{scaler}(\lceil (2/3) \cdot n(t'(e)) \rceil)} \leq \beta.$$

Hence,

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

□

6. *Applications*

In this section we present seven optimization problems to which we apply our paradigm. For each problem, we dedicate a subsection in which we define the problem and show how to cast it in the paradigm of Section 2. The spreading metrics for the various problems are computed by formulating an appropriate linear programming relaxation. We first explain how to solve these linear programs optimally in polynomial time.

6.1. Polynomial-Time Solvability. Let $G = (V, E)$ be a graph, directed or undirected, let $c(\cdot)$ be a capacity function associated with the edge set E. A spreading metric attaches lengths $\{\ell(e)\}_{e \in E}$, to the edges. Denote by $\mathsf{dist}(u, v)$ the distance from u to v with respect to the metric induced by $\ell(\cdot)$.

Our linear programming formulations fall into one of two generic categories according to the structure of the auxiliary graph H. For the subset feedback set problem, the multicut problem in directed graphs, and the symmetric multicut problem in directed graphs, the auxiliary graph H is a set of edges that match pairs of terminals. In this case, $\mathsf{scaler}(H) = 1$. This means that the spreading metric linear program attaches lengths to the edges, such that the distance between pairs of terminals is at least one, while minimizing the volume of the graph. This version of spreading metrics draws directly from past work on multicommodity flow and feedback sets in directed graphs, for example, Leighton and Rao [1999], Seymour [1995], and Garg et al. [1996]. Thus, the linear program has the following form.

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e) \cdot \ell(e) \\ \text{s.t.} \quad & \text{For each pair of terminals } t, t': \qquad \mathsf{dist}(t, t') \geq 1 \\ & \forall e \in E : \ell(e) \geq 0 \end{aligned}$$

The linear program can be solved optimally in polynomial-time using the Ellipsoid algorithm [Grötschel et al. 1988]. This follows by observing that for a given solution, a shortest path computation between each pair of terminals can determine that, either, all constraints are satisfied, or discover a violated constraint. More on compact formulations of such linear programs and efficient solutions of such linear programs can be found in Shmoys [1997] and Even et al. [1998].

The following problems have an auxiliary graph H which is a clique: linear arrangement, embedding a graph in a d-dimensional mesh, interval graph completion, minimizing storage-time product, q-balanced partitions and ρ-separators in directed graphs. Here, $\mathsf{scaler}(H)$ depends on the number of vertices in H. Consider the following linear program.

$$\begin{aligned} \min \quad & \sum_{e \in E} c(e) \cdot \ell(e) \\ \text{s.t.} \quad & \forall U \subseteq V, \quad \forall v \in U : \sum_{u \in U} \mathsf{dist}(u, v) \geq f(|U|) \\ & \forall e \in E : \ell(e) \geq 0 \end{aligned}$$

Note that the linear program guarantees the diameter of every subset $U \subseteq V$ by introducing an "averaging" constraint. Namely, the diameter of U measured from v is at least $f(|U|)/(|U| - 1)$. Hence, the diameter guarantee is satisfied if $f(|U|)$ is defined by $f(|U|) = (|U| - 1) \cdot \mathsf{scaler}(H_U)$.

An optimal solution to the linear program can be computed in polynomial time using the Ellipsoid Algorithm [Grötschel et al. 1988]. This follows by observing that for a given candidate solution, we can determine that either all constraints are satisfied, or discover a violated constraint. We 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. More details on efficient solutions of such linear programs can be found in Shmoys [1997] and Even et al. [1998].

6.2. 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: $V \rightarrow \{1, \ldots, |V|\}$, that minimizes the total edge lengths, that is, $\Sigma_{e=(i,j)\in E} c(e) \cdot |h(i) - h(j)|$.

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 Garey and Johnson 1979, 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 $\mathsf{scaler}(H = (V, E_H)) \triangleq |V| - 1$. To establish the divide-and-conquer applicability we consider any binary decomposition tree T that fully decomposes the problem. Note that there is a 1-1 correspondence between the leaves of T and the vertices of G. The solution to the linear arrangement problem that is represented by T is given by arranging the vertices of G in the order of their appearance as leaves of T. The cost of the tree T is $\mathsf{cost}(T) = \Sigma_{t\in T}(|V_t| - 1)c(F_t)$, where V_t and F_t are the set of vertices and cut corresponding to the tree node t. We need to show that this cost bounds the cost of solutions built up from T. For this we prove that for every tree node t the cost of the subtree rooted at t, denoted T_t, bounds the cost of solutions built up from T_t to the linear arrangement problem for the subgraph of G induced by the set of vertices V_t. We prove the claim by induction on the level of the tree nodes. The claim clearly holds for all leaves of T. Consider an internal tree node $t \in T$ and denote its two children by t_L and t_R. By induction the claim holds for both t_L and t_R. The solution represented by T_t is given by concatenating the solutions represented by T_{t_L} and T_{t_R}. Note that the additional cost is at most $|V_t| - 1 = \mathsf{scaler}(H_t)$ times the capacity of the cut F_t that separates V_{t_L} from V_{t_R}. We get

$$\mathsf{cost}(T_t) \leq \mathsf{cost}(T_{t_L}) + \mathsf{cost}(T_{t_R}) + (|V_t| - 1) \cdot c(F_t).$$

The inductive claim follows.

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

$$\min \quad \sum_{e \in E} c(e) \cdot \ell(e)$$

$$\text{s.t.} \quad \forall U \subseteq V, \quad \forall v \in U: \sum_{u \in U} \mathsf{dist}_\ell(u, v) \geq \frac{1}{4}(|U|^2 - 1)$$

$$\forall e \in E: \ell(e) \geq 0$$

In the linear program, we follow our previous notation that regards $\ell(e)$ as edge lengths, and $\mathsf{dist}_\ell(u, v)$ is the length of the shortest path from u to v.

LEMMA 10. *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 $\mathsf{dist}_\ell(u, v) \geq \frac{1}{4}(|U| - 1)$.*

PROOF. The average distance of a node $u \in U - \{v\}$ from v is greater than $(1/4)(|U| - 1)$, 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: while the "diameter guarantee" requires that the diameter of a subset U be greater than $\mathsf{scaler}(H_U)$, the proven bound is only $\mathsf{scaler}(H_U)/4$. As stated in Section 2.5 in Remark 3, 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 11. *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 $h: V \to \{1, \ldots, |V|\}$ of G. Define $\ell(e) = |h(i) - h(j)|$ for $e = (i, j) \in E$. We show that $\ell(e)$ is a feasible solution. Clearly, the cost $\Sigma_{e \in E} c(e) \cdot \ell(e)$ equals the cost of the linear arrangement h. The feasibility of $\ell(\cdot)$ is proved as follows: Consider a subset $U \subseteq V$, and a vertex $v \in U$. We observe that the average distance from v of the vertices in U is minimized when U is "packed" around v. Now, 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,

$$\begin{aligned}
\sum_{u \in U} \mathsf{dist}_\ell(u, v) &= \sum_{u \in U_L} \mathsf{dist}_\ell(u, v) + \sum_{u \in U_R} \mathsf{dist}_\ell(u, v) \\
&\geq \sum_{i=1}^{|U_L|} i + \sum_{i=1}^{|U_R|} i \\
&\geq \sum_{i=1}^{\lfloor (|U|-1)/2 \rfloor} i + \sum_{i=1}^{\lceil (|U|-1)/2 \rceil} i \\
&= \begin{cases} \frac{1}{4}|U|^2 & \text{if} \quad |U| \text{ is even} \\ \frac{1}{4}(|U|^2 - 1) & \text{if} \quad |U| \text{ is odd} \end{cases}
\end{aligned}$$

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

Note that a binary decomposition tree specifies 2^{n-1} solutions to the linear arrangement problem, depending on the orientations of the internal tree nodes (i.e., determining for each internal node which child is the left child). Our approximation algorithm constructs a decomposition tree such that every solution specified by the decomposition tree approximates the optimal linear arrangement. In Bar-Yehuda et al. [1998], a polynomial time algorithm that finds one of the best solutions to the linear arrangement problem among the solutions that are specified by a given decomposition tree is presented.

6.3. 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 mesh. The value of the embedding is $\Sigma_{(u,v)\in E} d(h(u), h(v))$, 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, that is, $\Sigma_{e=(i,j)\in E} c(e) \cdot |h(i) - h(j)|^{1/d}$.

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 arrangement with d-dimensional cost.

Lemma 12. *Given a linear ordering of G of d-dimensional cost A, one can produce an embedding in the d-dimensional mesh of G of cost-value $O(d \cdot A)$.*

Proof Sketch. This follows directly from the existence of a curve through a d-dimensional mesh, in which any two vertices that differ by k along the curve are at distance $O(dk^{1/d})$ in the d-dimensional mesh. Given a linear ordering we transform it to an embedding in the d-dimensional mesh by mapping the linear order to such a curve and then by mapping each vertex on the curve to its corresponding point in the d-dimensional mesh. It is easy to see that the property of the curve implies that the cost of the embedding in the d-dimensional mesh is at most $O(d)$ times the d-dimensional cost of the linear ordering. A Peano curve is an example of such a curve. See Sagan [1994] for a description of space filling curves. □

Note that due to Lemma 12, the dimensionality of the mesh, namely d, increases the approximation factor by a linear factor in d. From now on we consider the problem of linear arrangement with d-dimensional cost.

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 $\mathsf{scaler}(H) \triangleq (|V| - 1)^{1/d}$. Clearly, if H contains no edges, then $|V| = 1$ and the problem (G, H) is trivial with $\mathsf{cost}(G, H) = 0$. As in the linear arrangement problem, to establish the divide-and-conquer applicability we consider any binary decomposition tree T that fully decomposes the problem. The solution to the linear arrangement problem with d dimensional cost that is represented by T is given by arranging the vertices of G in the order of their appearance as leaves of T. The cost of the tree T is $\mathsf{cost}(T) = \Sigma_{t \in T}(|V_t| - 1)^{1/d} c(F_t)$. To show that this cost bounds the cost of the solution represented by T, we prove that for every tree node t the cost of T_t bounds the cost of the solution represented by T_t. As before, we prove the claim by induction on the level of the tree nodes. The claim clearly holds for all leaves of T. Consider an internal tree node $t \in T$ and denote its two children by t_L and t_R. By induction the claim holds for both t_L and t_R. The solution represented by T_t is given by concatenating the solutions represented by T_{t_L} and T_{t_R}. Note that the additional cost is at most $(|V_t| - 1)^{1/d} = \mathsf{scaler}(H_t)$ times the capacity of the cut F_t. We get

$$\mathsf{cost}(T_t) \leq \mathsf{cost}(T_{t_L}) + \mathsf{cost}(T_{t_R}) + (|V_t| - 1)^{1/d} \cdot c(F_t).$$

The inductive claim follows.

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

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

The following lemma is analogous to Lemma 10, and proves that the diameter guarantee is satisfied up to a constant of 4.

LEMMA 13. *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 $\mathsf{dist}_\ell(u, v) \geq (1/4) \cdot (|U| - 1)^{1/d}$.*

The following lemma is analogous to Lemma 11, and proves that the lower bound property is satisfied.

LEMMA 14. *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 11. Consider a linear ordering $h(\cdot)$ of the vertices of G. For every edge $(i, j) \in E$, define $\ell(i, j)$ to be $|h(j) - h(i)|^{1/d}$. Clearly, the cost $\Sigma_{e \in E} c(e) \cdot \ell(e)$ equals the d-dimensional cost of the embedding $h(\cdot)$. The 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.

$$\sum_{u \in U} \mathsf{dist}_\ell(u, v) \geq \sum_{i=1}^{\lfloor (|U|-1)/2 \rfloor} i^{1/d} + \sum_{i=1}^{\lceil (|U|-1)/2 \rceil} i^{1/d}$$

$$\geq \int_0^{\lfloor (|U|-1)/2 \rfloor} x^{1/d} dx + \int_0^{\lceil (|U|-1)/2 \rceil} x^{1/d} dx$$

$$\geq 2 \frac{d}{1+d} \cdot \left(\frac{|U| - 1}{2} \right)^{1+1/d}$$

$$\geq \frac{1}{4} (|U| - 1)^{1+1/d} \text{ (assuming } d \geq 1). \qquad \square$$

Remark. An alternative method for finding a d-dimensional embedding is by following Hansen [1989] technique. 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.

6.4. Minimizing Storage-Time Product. Following Ravi et al. [1991], 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 $e = (u \to v)$ with capacity $c(e)$ corresponds to $c(e)$ units of storage generated by task u and consumed by task v.

Output: A linear ordering $h: V \to \{1, \ldots, |V|\}$ of the vertices such that if $(u \to v) \in E$, 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 storage-time product. Assuming that all tasks are of unit length, this translates to minimizing the cost of the ordering defined by $\mathsf{cost}(h) \triangleq \Sigma_{u \to v}(h(v) - h(u)) \cdot c(u \to v)$.

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. [1991], 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:

$$\min \quad \sum_{e \in E \cup E^R} c(e) \cdot \ell(e)$$

$$\text{s.t.} \quad \forall U \subseteq V, \quad \forall v \in U: \sum_{u \in U} (\mathsf{dist}_\ell(u, v) + \mathsf{dist}_\ell(v, u)) \geq \frac{1}{4} (|U|^2 - 1)$$

$$\forall e \in E \cup E^R: \ell(e) \geq 0$$

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 Lemma 10 and 11.

LEMMA 15. *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*

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

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

LEMMA 16. *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 11. Note that since the reversed edges have infinite capacity they must be given zero length in order that the cost would be bounded. Thus, when we follow the lines of Lemma 11 we ignore these edges and consider only the original edges. □

6.5. 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 Ramalingam and Pandu Rangan [1988]. 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 each such edge (v_i, v_{i+1}), $h(v_{i+1}) \geq h(v_i)$. The above characterization ensures that all vertices between v_i and v_{i+1} are connected to v_{i+1}. It follows that $\mathsf{deg}(v_{i+1}) \geq h(v_{i+1}) - h(v_i)$. 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 $\mathsf{scaler}(H = (V, E_H)) = |V| - 1$. To establish the divide-and-conquer applicability we consider any binary decomposition tree T that fully decomposes the problem. As before, each leaf of T corresponds to a distinct vertex of G. However, since we deal now with the "vertex version" each internal tree node $t \in T$ is associated with a vertex separator F_t rather than an edge cut. Namely, let G_t be the subgraph of G induced by V_t, and denote the two children of t by t_L and t_R. Then, F_t is the vertex separator that separates V_{t_L} from V_{t_R} in G_t. Clearly, the sets F_t for all $t \in T$ are disjoint and do not contain any vertex that corresponds to a leaf of T. The vertex ordering which defines a solution to the interval graph completion problem that is represented by T is defined recursively following the *nested dissection* construction introduced by Ravi et al. [1991]. Consider an internal tree node $t \in T$ and denote its two children by t_L and t_R. The ordering represented by T_t is given by the ordering represented by T_{t_L}, followed by the ordering represented by T_{t_R}, followed by the vertices in F_t in arbitrary order. The cost of the tree T is $\mathsf{cost}(T) = \Sigma_{t \in T}(|V_t| - 1)|F_t|$. To show that this cost bounds on the cost of the solution represented by T, we prove that for every tree node t the cost of T_t bounds the cost of the solution represented by T_t. Again, we prove the claim by induction on the level of the tree nodes. The claim clearly holds for all leaves of T. Consider an internal tree node $t \in T$ and denote its two children by t_L and t_R. By induction the claim holds for both t_L and t_R. The inductive claim follows since

$$\mathsf{cost}(T_t) \leq \mathsf{cost}(T_{t_L}) + \mathsf{cost}(T_{t_R}) + (|V_t| - 1) \cdot |F_t|.$$

The above inequality holds because at most $(|V_t| - 1)$ edges need to be added per each vertex in F_t. Note that adding these edges is equivalent to making the intervals corresponding to F_t intersect all other intervals in V_t.

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 a fractional relaxation of the degree of v in an optimal solution. Let $\mathsf{dist}_\ell(u, v)$ denotes the shortest path connecting u and v with respect to the vertex lengths $\ell(v)$.

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

The following two lemmata are analogous to Lemmata 10 and 11 and are proved similarly.

LEMMA 18. *Let $\ell(v)$ 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 $\mathsf{dist}_\ell(u, v) \geq (1/4)|U|$.*

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 an interval graph $G^*(V, E^*)$. For every $v \in V$, let $\ell(v)$ be the degree of v in G^*. Clearly, the cost $(1/2)\Sigma_{v \in V} \ell(v)$ 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, $\mathsf{dist}_\ell(u, v) = \Sigma_{x \in P} \ell(x)$. Since $\ell(x)$ is the degree of x in G^*, by Lemma 17 we have $\Sigma_{x \in P} \ell(x) \geq |h(u) - h(v)|$. Summing over all vertices in U and using Lemma 11, we get

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

and the lemma follows. □

6.6. SUBSET 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 Evan et al. [1998]. 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, that is, a cycle containing a vertex from X.

In Evan et al. [1998], it is shown that this problem is equivalent to the problem of finding a multicut in a 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* $\{(s_i, t_i)\}_{i=1,\ldots,k}$. For every source-sink pair, (s_i, t_i), there is an infinite capacity edge $t_i \to s_i$.

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 $E_H = \{(s_i, t_i)\}_{i=1,\ldots,k}$. The scaler function is defined by $\mathsf{scaler}(H = (V, E_H)) = 1$ if E_H is nonempty, and 0 otherwise. Clearly, if $E_H = \emptyset$, then all the source-sink pairs are separated and the problem (G, H) is trivial. To establish the divide-and-conquer applicability, we consider any binary decomposition tree T that fully decomposes the problem. Consider an internal tree node $t \in T$ and denote its two children by t_L and t_R. Let F_t be the (directed) (V_{t_L}, V_{t_R}) cut in the subgraph induced by V_t. (Namely, the edges in F_t intersect any path from a vertex from V_{t_L} to a vertex in V_{t_R}.) The solution to the multicut problem that is represented by T is given by the union of all cuts F_t, for $t \in T$. Note that this union would not constitute a solution for arbitrary directed multicut problems, but only for circular networks which have infinite capacity edges from each sink to its corresponding source. The reason for this is that a region that is grown by the partition procedure around a terminal (either a source s_i or a sink t_i), may not contain a sink t_j without containing the source s_j. Therefore, if an additional source-sink pair s_j, t_j happens to be separated by a region, then the selected directed cut separates this pair in the right direction (i.e., covering every path from s_j to t_j). The cost of the tree T is $\mathsf{cost}(T) = \Sigma_{t \in T} c(F_t)$. To show that this cost bounds, the solution to the multicut problem that is represented by T, we prove that for every tree node t the cost of T_t bounds the cost of the solution represented by T_t. As before, we prove the claim by induction on the level of the tree nodes. For each leaf, the subgraph induced by the leaf corresponds to a trivial problem. For each internal tree node t, we have

$$\mathsf{cost}(T_t) \leq \mathsf{cost}(T_{t_L}) + \mathsf{cost}(T_{t_R}) + c(F_t).$$

The inductive claim follows.

The spreading metric is defined by the following linear program:

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

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

LEMMA 20. *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 21. *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. □

6.7. SYMMETRIC MULTICUTS IN DIRECTED GRAPHS. Klein et al. [1997] 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 in circular networks. 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.

$$\min \quad \sum_{e \in E} c(e) \cdot \ell(e)$$

$$\text{s.t.} \quad \text{For } 1 \le i \le k, \text{ for every cycle } C \text{ that intersects } s_i \text{ and } t_i\text{: } \sum_{e \in C} \ell(e) \ge 1$$

$$\forall e \in E : \ell(e) \ge 0$$

The following two lemmata are similar to Lemmata 20 and 21.

LEMMA 22. *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 23. *The cost of an optimal solution of the linear program is a lower bound on the cost of an optimal symmetric multicut of G.*

6.8. 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 Even et al. [1999]. We consider here only directed graphs since the approximation factor given in Even et al. [1999] 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 Even et al. [1999].

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 $0 < \rho \leq 1$.

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 "bicriteria 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 more restricted problem. We need the difference between ρ and ν in order to guarantee a diameter, as described in Lemma 24, and the approximation factor depends also on $1/(\rho - \nu)$. 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 Even et al. [1999], we relate the ρ-separator problem to the problem of partitioning a graph into q roughly equal parts, which is called the q-balanced partitioning 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 than $2/\rho$ groups. Hence, we can compute a q-balanced partition by finding a $2/q$-separator and grouping the resulting components.

The q-balanced partitioning problem was considered by Leighton et al. [1990] and by Simon and Teng [1997]. They proposed applying recursively the approximate separator algorithm in Leighton and Rao [1999] until all strongly connected components are small enough. This approach yields a q-balanced partitioning in which the number of vertices in each part is bounded by $2n/q$. The capacity of the separator is $O(\log n \log q \cdot \tau')$, where τ' denotes the minimum cost of a $1/q$-separator. Our algorithm is more flexible since for any fixed $\varepsilon > 0$ it yields a partitioning in which each part contains at most $(1 + \varepsilon)n/q$ vertices, where 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 $q > \min\{\log n, \tau'^{\log\log \tau'/\log n}\}$.

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 subhypergraph 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.

We now show that an optimal solution to the following linear program is a spreading metric.

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

Note that the constraints for small subsets of vertices (namely, $|U| \leq \nu \cdot n$) are trivial and may be omitted. Moreover, constraints for subsets $|U| \leq \rho \cdot n$ may be omitted since such subsets need not be partitioned. (This reduces the complexity of solving the linear program in some algorithms [Even et al. 1999].)

LEMMA 24. *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*

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

The proof of this lemma is similar to the proof of Lemma 10. Note however, that our relaxation of the diameter guarantee adds a factor of $2\rho/(\rho - \nu)$ to the approximation factor. Since we cannot guarantee a diameter of at least $(\rho - \nu)/2\rho$ unless there are at least $\rho \cdot n$ vertices, we obtain a ρ-separator rather than a ν-separator.

LEMMA 25. *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 F, or every path from v to u contains an edge of F. Hence, $\mathsf{dist}_\ell(v, u) + \mathsf{dist}_\ell(u, v) \geq 1$, and

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

□

7. *Further Results*

Since the preliminary version of this paper appeared in Even et al. [1995], several new applications of spreading metrics were discovered.

(1) Even et al. [1999] used spreading metrics to obtain improved approximation factors for balanced partitioning problems.
(2) Kuo and Cheng [1997] used spreading metrics for partitioning problems that arise in VLSI design.
(3) Rao and Richa [1998] improved the approximation factor for the linear arrangement problem and for the interval graph completion problem to $O(\log n)$.
(4) Blum et al. [1998] used spreading constraints in a semi-definite program for approximating the bandwidth problem and obtained an approximation factor of $O(\sqrt{n})$. It is interesting to note that this is the best approximation factor that can be achieved for this problem using spreading constraints. To obtain polylogarithmic approximation factors, more constraints need to be introduced, as shown by Feige [1998]. Based on Feige's volume preserving embeddings, Vempala [1998] used a spreading metric similar to the spreading metric used in this paper for graph embeddings in d-dimensional meshes to compute edge lengths which are used by the volume preserving embedding. Vempala presented a polylogarithmic approximation algorithm for the bandwidth problem in embeddings in d-dimensional meshes as well as a bicriteria polylogarithmic approximation algorithm that minimizes the bandwidth and the sum of the edge lengths in embeddings in d-dimensional meshes.

REFERENCES

AUMANN, Y., AND RABANI, Y. 1998. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput. 27*, 291–301.

BAR-YEHUDA, R., EVEN, G., AND NAOR, J. 1998. Computing an optimal orientation of a balanced decomposition tree for linear arrangement problems. Manuscript.

BLUM, A., KONJEVOD, G., RAVI, R., AND VEMPALA, S. 1998. Semi-definite relaxations for minimum bandwidth and other vertex ordering problems. In *Proceedings of the 30th ACM Symposium on Theory of Computing*. ACM, New York, pp. 100–105.

BHATT, S. N., AND LEIGHTON, F. T. 1984. A framework for solving VLSI graph layout problems. *J. Comput. Syst. Sci. 28*, 300–343.

EVEN, G., NAOR, J., RAO, S., AND SCHIEBER, B. 1998. Divide-and-conquer approximation algorithms via spreading metrics. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science.* IEEE Computer Science Press, Los Alamitos, Calif., pp. 62–71.

EVEN, G., NAOR, J., RAO, S., AND SCHIEBER, B. 1999. Fast approximate graph partitioning algorithms. *SIAM J. Comput. 28*, 6, 2187–2214.

EVEN, G., NAOR, J., SCHIEBER, B., AND SUDAN, M. 1998. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica 20*, 151–174.

FEIGE, U. 1998. Approximating the bandwidth via volume respecting embeddings. In *Proceedings of the 30th ACM Symposium on Theory of Computing.* ACM, New York, pp. 90–99.

GAREY, M. R., AND JOHNSON, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, Calif.

GRÖTSCHEL, M., LOVASZ, L., AND SCHRIJVER, A. 1988. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, New York.

GARG, N., VAZIRANI, V. V., AND YANNAKAKIS, M. 1996. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM J. Comput. 25*, 235–251.

HANSEN, M. 1989. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science.* IEEE Computer Society Press, Los Alamitos, Calif., pp. 604–609.

KUO, M.-T., AND CHENG, C.-K. 1997. A network flow approach for hierarchical tree partitioning. In *Proceedings of the 34th ACM-IEEE Design Automation Conference* (Anaheim, Calif., June 9–11). ACM, New York, pp. 512–517.

KLEIN, P. N., PLOTKIN, S. A., RAO, S., AND TARDOS, E. 1997. Approximation algorithms for Steiner and directed multicuts. *J. Algorithms 22*, 241–269.

LEIGHTON, F. T., MAKEDON, F., AND TRAGOUDAS, S. 1990. Approximation algorithms for VLSI partition problems. In *Proceedings of the IEEE International Symposium on Circuits and Systems.* IEEE Computer Society Press, Los Alamitos, Calif. pp. 2866–2868.

LEIGHTON, F. T., AND RAO, S. 1999. An approximate max-flow min-cut theorem for uniform multi-commodity flow problems with applications to approximation algorithms. *J. ACM, 46*, 6 (Nov.), 787–832.

LINIAL, N., LONDON, E., AND RABINOVICH, Y. 1995. The geometry of graphs and some of its algorithmic applications. *Combinatorica 15*, 215–245.

NESTEROV, Y., AND NEMIROVSKII, A. 1994. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, Philadelphia.

RAVI, R., AGRAWAL, A., AND KLEIN, P. 1991. Ordering problems approximated: Single processor scheduling and interval graph completion. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming.* pp. 751–762.

RAMALINGAM, G., AND PANDU RANGAN, C. 1988. A unified approach to domination problems in interval graphs. *Inf. Proc. Lett. 27*, 271–274.

RAO, S., AND RICHA, A. 1998. New approximation techniques for some ordering problems. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms.* ACM, New York, pp. 211–219.

SAGAN, H. 1994. *Space-Filling Curves.* Springer-Verlag, New York.

SEYMOUR, P. D. 1995. Packing directed circuits fractionally. *Combinatorica 15*, 281–288.

SHMOYS, D. B. 1997. Cut problems and their application to divide-and-conquer. In *Approximation algorithms for NP-hard problems.* D. S. Hochbaum, Ed. PWS Publishing Co.

SIMON, H. D., AND TENG, S-H. 1997. How good is recursive bisection. *SIAM J. Sci. Comput. 18*, pp. 1436–1445.

VEMPALA, S. 1998. Random projection: A new approach to VLSI layout. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science.* IEEE Computer Society Press, Los Alamitos, Calif., pp. 389–395.

RECEIVED FEBRUARY 1996; REVISED SEPTEMBER 1999; ACCEPTED SEPTEMBER 1999