RC 20074 (5/24/95)
Computer Science/Mathematics

IBM Research Report

Approximating Minimum Feedback Sets and Multicuts in Directed Graphs

G. Even, J. Naor

Computer Science Department
Technion
Haifa, Israel

B. Schieber, M. Sudan

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

Research Division
Almaden · T.J. Watson · Tokyo · Zurich

Approximating Minimum Feedback Sets and Multicuts in Directed Graphs*

Guy Even† Joseph (Seffi) Naor‡ Baruch Schieber§ Madhu Sudan§

Abstract

This paper deals with approximating feedback sets in directed graphs. We consider two related problems: the *weighted feedback vertex set* (FVS) problem, and the *weighted feedback edge set* problem (FES). In the FVS (resp. FES) problem, one is given a directed graph with weights (each of which at least 1) on the vertices (resp. edges), and is asked to find a subset of vertices (resp. edges) with minimum total weight that intersects every directed cycle in the graph. These problems are among the classical NP-Hard problems and have many applications. We also consider a generalization of these problems: SUBSET-FVS and SUBSET-FES, in which the feedback set has to intersect only a subset of the directed cycles in the graph. This subset contains all the cycles that go through a distinguished input subset of vertices and edges, denoted by X. This generalization is also NP-Hard even when $|X| = 2$. We present approximation algorithms for the SUBSET-FVS and SUBSET-FES problems. The first algorithm we present achieves an approximation factor of $O(\log^2 |X|)$. The second algorithm achieves an approximation factor of $O(\min\{\log \tau^* \log\log \tau^*, \log n \log\log n)\}$, where τ^* is the value of the optimum fractional solution of the problem at hand, and n is the number of vertices in the graph. We also define a multicut problem in a special type of directed networks which we call circular networks, and show that the SUBSET-FES and SUBSET-FVS problems are equivalent to this multicut problem. Another contribution of our paper is a *combinatorial* algorithm that computes a $(1+\varepsilon)$ approximation to the fractional optimal feedback vertex set. Computing the approximate solution is much simpler and more efficient than general linear programming methods. All of our algorithms use this approximate solution.

*A preliminary version of this paper appeared in the Proceedings of the 4th MPS Conference on Integer Programming and Combinatorial Optimization (IPCO), University of Copenhagen, Copenhagen, 1995.

†Computer Science Dept., Technion, Haifa 32000, Israel. E-mail:guy@cs.technion.ac.il. Research supported by the Miriam and Aaron Gutwirth Memorial Fellowship. Most of the research was done while visiting IBM T.J. Watson Research Center.

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

§IBM, T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598.
E-mail:{sbar,madhu}@watson.ibm.com.

1 Introduction

This paper deals with approximating feedback sets in directed graphs. We consider two related problems: the *weighted feedback vertex set* (FVS) problem, and the *weighted feedback edge set* problem (FES). In the FVS problem, one is given a directed graph with weights on the vertices, and is asked to find a subset of vertices with minimum total weight that intersects every directed cycle in the graph. In the FES problem, one is given a directed graph with weights on the edges, and is asked to find a subset of edges with minimum total weight that intersects every cycle in the graph. We assume that all weights are at least 1. The unweighted versions of these problems are classical NP-Hard problems, and appear in Karp's seminal paper [K72]. The FES and FVS problems are reducible to one another (§2). These reductions preserve approximations, namely, feedback vertex sets are mapped to feedback edge sets with the same weight and vice-versa. Hence, these problems are equally hard to approximate in polynomial time.

The problem of finding feedback sets arises in a variety of applications. One of the most interesting and important applications has to do with testing circuits. A circuit can be modeled by a directed graph where the vertices represent gates (which compute boolean functions) and by directed edges which represent wires that connect gates [LS91]. Loosely speaking, finding a small feedback edge set in this graph helps to reduce the hardware overhead required for testing the circuit using "scan registers" [GGB89, KW90]. Other applications have to do with efficient deadlock resolution, and analyzing manufacturing processes.

We also consider a generalization of the FES and FVS problems which we call the SUBSET-FES and SUBSET-FVS problems. In these problems, only a subset of the directed cycles in the graph is considered *interesting*. A feedback edge set with respect to the interesting cycles is a subset of edges that intersects every interesting cycle. Similarly, a feedback vertex set with respect to the interesting cycles is a subset of vertices that intersects every interesting cycle. The interesting sets of cycles that we consider are characterized by a subset of vertices and edges. Namely, given a subset X (of special vertices and edges), the set of interesting cycles characterized by X is the set of all cycles that intersect X. In the SUBSET-FES (resp. SUBSET-FVS) problem the goal is to find a minimum weight feedback edge (resp. vertex) set with respect to the interesting cycles. The SUBSET-FVS (SUBSET-FES) problem is NP-Hard even when the interesting cycles are characterized by a set X containing only two vertices, as proved in [YKCP83]. (See also a remark at the end of the paper.)

The motivation of this generalization is two-fold. First, in some of the applications we may only be interested in cycles that intersect a subset of the vertices and edges, e.g., when testing only part of a circuit. Second, from the theoretical standpoint, it is interesting whether the quality of the approximation depends on the size of the set that characterizes the interesting subset of cycles.

The relation between feedback set problems and multicut problems (first observed by Leighton and Rao [LR88]), motivated many researchers to focus their attention on multicut problems. This problem was first defined by Hu [H63] as follows. Given a capacitated network, and a set of k source-sink pairs, find a minimum capacity set of edges whose removal disconnects all the source-sink pairs. This problem is NP-Hard in general. We define a directed multicut problem in a special type of networks which we call *circular networks*. We show that the subset feedback set problem is equivalent to the directed multicut problem in such networks.

The first approximation algorithm for the FES problem was given by Leighton and Rao [LR88] (and followed by [KST90]). Their approximation factor is $O(\log^2 n)$ in the unweighted case, where n is the number of vertices in the graph. Leighton and Rao achieve this bound by using an $O(\log n)$ approximation algorithm for a directed separator that splits the graph into two (approximately) equal sized components. This separator is found by approximating special cuts which are called quotient cuts.

Recently, there has been much progress in approximating the multicut problem in the *undirected* case. Most notably, Garg, Vazirani and Yannakakis [GVY93] achieved a $O(\log k)$ approximation factor for this problem. They introduced a novel "sphere growing" technique, refining and simplifying previous works of [LR88, KARR90]. Actually, the $O(\log k)$ factor achieved in [GVY93] is with respect to the optimal fractional solution of the multicut problem. Garg, Vazirani and Yannakakis also show that this is the best one can hope for, by demonstrating a graph in which the gap between the fractional and integral solution is indeed $\Omega(\log k)$. To the best of our knowledge, no approximation is known for the multicut problem in general directed graphs.

The first result we present is an approximation algorithm that achieves an $O(\log^2 k)$ factor for the subset feedback set problem. Recall the equivalence established between the subset feedback set problem, and the directed minimum capacity multicut problem in circular networks. We are able to exploit the special structure of such networks to find an $O(\log^2 k)$ approximation factor of the minimum capacity multicut. Our algorithm is based on a novel decomposition of the graph, and an adaptation of the undirected "sphere growing" technique of [GVY93] to directed circular networks.

In a brilliant paper, Seymour [S92] proved that the integrality gap in the case of the unweighted FVS problem can be at most $O(\log \tau^* \log\log \tau^*)$, where τ^* denotes the optimal value of a fractional feedback set. A careful examination of Seymour's proof reveals that all of his existential arguments can be made constructive, and thus, with certain other modifications, an algorithm for the unweighted FVS problem that achieves an approximation factor of $O(\log \tau^* \log\log \tau^*)$ can be obtained. (Notice that in the unweighted case $\tau^* < n$.)

The second result we present is an extension of Seymour's proof to the weighted SUBSET-FVS problem. This results in an algorithm that achieves an approximation factor of $O(\min\{\log \tau^* \log\log \tau^*, \log n \log\log n\})$, where, as before, τ^* denotes the cost of a minimum fractional subset feedback set, and n is the number of vertices. This result is almost optimal within the primal-dual framework, since the integrality gap is $\Omega(\log \tau^*)$ [S92].

Both of our approximation algorithms recursively decompose a directed graph as follows. A fractional (optimal) solution to a directed feedback set problem induces a distance metric on the edge (vertex) set of the graph. The approximation algorithm picks (arbitrarily) a vertex $s \in X$, and conducts a single-source-shortest-paths algorithm from s with respect to the distance metric. The single-source-shortest-paths algorithm defines layers with respect to the source s, where each layer is a directed cut that partitions the graph into two parts. The approximation algorithm chooses a directed cut, adds the cut to the feedback set, and continues recursively in each part. The recursion ends when the graph does not contain any interesting cycles. Each approximation algorithm applies a different criterion for choosing the directed cut that partitions the graph. However, both algorithms relate the weight of the cut to the cost of the fractional solution.

The first step in all feedback set approximation algorithms requires the computation of an optimal fractional solution. Notice that the linear programming formulation of the FVS problem may contain an exponential number of constraints. Nevertheless, an optimal solution can be computed in this case in polynomial time by either using the Ellipsoid method or interior point methods. In fact, it is possible to decrease the number of constraints to be polynomial by reducing the FVS problem to a multi-commodity flow [GVY93]. (But, then the linear program is not positive anymore.) Using general linear programming methods is usually undesirable, and algorithms that exploit the combinatorial structure of the problem are sought when possible. Since we deal with approximation algorithms, we can actually settle for an approximate (initial) fractional solution, where the approximation bound is a constant.

In Section 5 we present a combinatorial algorithm that finds a $(1+\varepsilon)$ approximation to the fractional FVS and SUBSET-FVS problems. This algorithm is simple and also more efficient than general linear programming methods. The complexity of our algorithm is $O(n^2 M(n) \log^2 n)$ for any fixed ε, where $M(n)$ denotes the complexity of matrix multiplication. Our algorithm is based on a greedy $(1+\varepsilon)$ approximation algorithm for the (fractional) Set Cover Problem derived from a parallel algorithm for approximating positive linear programming by Luby and Nisan [LN93]. We show how to adapt the approximate Set Cover algorithm to the FVS and SUBSET-FVS problem, in spite of the fact that the number of constraints in the corresponding Set Cover Problem is exponential. Plotkin, Shmoys and Tardos [PST91] gave a general combinatorial approximation algorithm for fractional packing and covering problems. Their algorithm can also be applied to compute an approximate fractional solution in our case. Our algorithm for computing an approximate fractional solution differs significantly from the algorithm derived from [PST91]: They use a Lagrangian relaxation technique on the dual problem, whereas we deal with the primal problem. The complexity of the approximation algorithm of [PST91] is better than our algorithm by a factor of $O(n)$. However, we believe that our algorithm is simpler.

Independently, Klein *et al.* [KPRT93] present a network decomposition algorithm for directed graphs called *symmetric multicuts.* Using a weighted version of their decomposition they derive an $O(\log^2 k)$ approximation algorithm for finding a minimum capacity subset of edges that separates k given pairs of terminals. That is, for each pair (s_i, t_i) either the path from s_i to t_i or the path from t_i to s_i is separated. The NP-Hardness of finding such a minimum capacity decomposition follows from the reduction of the SUBSET-FES problem to this problem as shown below. The decomposition algorithm in [KPRT93] is essentially the same as our algorithm for approximating the minimum capacity multicut in circular networks. In fact, we show in the sequel how our approximation algorithms can be applied to obtain an $O(\min\{\log\tau^*\log\log\tau^*, \log n\log\log n\}, \log^2 k)$ approximation algorithm for finding a minimum capacity subset of edges that separates k given pairs of terminals, where τ^* is the optimal fractional solution.

To summarize, the results presented in this paper are:

1. Reductions between the various problems considered in the paper (§2).

2. A polynomial-time approximation algorithm for the multicut problem in circular networks that achieves an $O(\log^2 k)$ approximation factor, where k is the number of source-sink pairs. We apply this algorithm to approximate the weighted SUBSET-FVS and weighted SUBSET-FES problems and obtain an approximation factor of $O(\log^2 k)$, where k is the number of vertices and edges that characterize the interesting cycles. (§3).

3. A polynomial-time approximation algorithm for the weighted SUBSET-FVS and weighted SUBSET-FES problems that finds a feedback set with weight $O(\min\{\tau^*\log\tau^*\log\log\tau^*, \tau^*\log|V|\log\log|V|\})$, where τ^* is the cost of an optimum fractional feedback set (§4). We note that this algorithm achieves the same approximation factor for the multicut problem in circular networks since this problem is equivalent to the subset feedback set problem.

4. A $(1+\varepsilon)$ factor approximation (combinatorial) algorithm for the fractional FVS problem (§5).

2 The problems and reductions among them

In this section we define the problems that are considered in the paper. We describe various versions of these problems and discuss reductions among them. In all the reductions, a feasible solution is mapped to a feasible solution. Moreover, the cost of the feasible solutions is preserved by the reductions, and therefore, an approximate solution to one problem can be translated to an approximate solution to all other problems reducible to this problem. Most of the reductions can be performed in linear time, and therefore, these problems can be regarded as different representations of the same problem.

2.1 The problems

Let $G = (V, E)$ denote a directed graph, where $|V| = n$. There are two natural weight functions that are associated with G: either weights on the edges, denoted by $c(e)$ for $e \in E$, or weights on the vertices, denoted by $c(v)$ for $v \in V$. Let X denote a subset of the vertices (edges) that are called *special* vertices (edges), and let $|X| = k$.

The first problems we consider are the minimum weight feedback vertex set (FVS), the minimum weight feedback edge set (FES) and their generalization to the SUBSET-FVS and SUBSET-FES problems. All these problems are defined in the introduction. Another extension of the FVS problem that we consider is the BLACKOUT-FVS problem. In this problem an additional subset of "blackout" vertices, B, is given. We allow only feedback vertex sets that do not contain any vertex of B. In the BLACKOUT-FVS problem a minimum weight feedback vertex set that does not contain vertices of B is sought.

We consider the problem of finding minimum capacity multicuts in directed "circular" multi-commodity networks. A *network* $N = \left(\widetilde{V}, \widetilde{E}, c(\cdot), \{(s_i, t_i)\}_{i=1,\dots,k}\right)$ is a quadruple, where: $(\widetilde{V}, \widetilde{E})$ is a directed graph, $c(\cdot)$ is a capacity function defined on the edges, and $\{(s_i, t_i)\}_{i=1,\dots,k}$ are pairs of vertices called *source-sink pairs.* The *capacity* (or cost) of a subset of edges, $\widetilde{F} \subseteq \widetilde{E}$, is defined by $c(\widetilde{F}) = \sum_{\widetilde{e} \in \widetilde{F}} c(\widetilde{e})$. A *multicut* in a network is a subset of edges, $\widetilde{F} \subseteq \widetilde{E}$, such that for every source-sink pair, (s_i, t_i), every path in N from s_i to t_i contains at least one edge of the set $\widetilde{F}$. We consider special networks, which we call *circular networks.* In circular networks there is an infinite capacity edge $t_i \rightarrow s_i$ for every source-sink pair (s_i, t_i), and this edge is the only edge that emanates from t_i and the only one that enters s_i. We note that the last condition, namely, that (s_i, t_i) is the only edge that emanates from t_i and the only one that enters s_i, is only used to show the equivalence of this problem to the SUBSET-FES problem. However, our approximation algorithm is valid even if this condition is not satisfied.

2.2 Reductions among the problems

In this subsection we show reductions between the problems presented in Section 2.1. These reductions preserve feasible solutions and their cost, and therefore, these problems are equally hard to approximate in polynomial time.

FES $\preceq$ FVS

Construct the "directed line-graph", $G' = (V', E')$, of $G = (V, E)$ as follows: $V' = E$ and an edge in E' connects $v_1 \to v_2 \in V'$ with $v_3 \to v_4 \in V'$ if and only if $v_2 = v_3$. In the weighted version, the weight of an edge $v_1 \to v_2 \in E$ equals the weight of the "vertex" $v_1 \to v_2 \in V'$. A subset of edges, E, is a feedback vertex set of G if and only if it is a feedback vertex set of G'. Note that this reduction can be used to reduce SUBSET-FES problems to SUBSET-FVS problems as well.

FVS $\preceq$ FES

Split every vertex, $v \in V$, into two parts, v_1 and v_2; all the edges that enter v are connected to v_1, and all the edges that emanate from v emanate from v_2. Add an edge $v_1 \to v_2$ for every vertex $v \in V$. All edges are given infinite weight, except for the edges $\{v_1 \to v_2 : v \in V\}$ whose weight is $c(v)$. There is a 1-1 correspondence between the finite weighted feedback edge sets of the new graph and the feedback vertex sets of the original graph. Note that this reduction can be used to reduce SUBSET-FVS problems to SUBSET-FES problems as well.

Characterizing interesting cycles

The subset, $X \subset V \cup E$, defining the interesting cycles can be reduced to a subset of vertices as follows: Every edge $u \to v \in X$ is split by adding a new vertex a_{uv} and by replacing $u \to v$ with $u \to a_{uv}$ and $a_{uv} \to v$. Add a_{uv} to X instead of the edge $u \to v$. If a feedback vertex set is sought, then set $w(a_{uv}) = \infty$. If a feedback edge set is sought, then set $w(u \to a_{uv}) = w(a_{uv} \to v) = w(u \to v)$. There is a 1-1 correspondence between the interesting cycles before and after the reduction. Moreover, there is a weight preserving correspondence between the finite weighted feedback vertex sets before and after the reduction. (The same holds for feedback edge sets). Conversely, The subset, $X \subset V \cup E$, defining the interesting cycles can be reduced to a subset of edges by splitting the vertices in X. We omit the details.

BLACKOUT-FVS $\preceq$ FVS

Bypass the vertices of the blackout vertices, B, one by one as follows: For each blackout vertex, $v \in B$, connect edges between every two vertices that have a path of length two connecting them in which v is the middle vertex. Remove the blackout vertex from the graph, and continue with the next vertex in B. A subset of vertices, U, is a feedback vertex set that does not contain any vertex from B, if and only if it is a feedback vertex set of the graph obtained by the reduction. Note, that this is the only reduction that requires more than linear time.

This reduction can also be extended to the case in which there is a subset $X \subseteq E$ that defines the interesting cycles. Namely, a cycle is interesting if only if it intersects X. A problem with the reduction arises when an edge of X is incident to a blackout vertex, since bypassing the blackout vertex removes this edge. We overcome this problem by adding a new bypassing edge to X if the path of length two from which it originates contains an edge of X.

SUBSET-FES and directed multicuts

We reduce an instance of SUBSET-FES where $X \subset V$ to an instance of the minimum multicut problem in circular networks, as follows. Construct the network $N(\tilde{V}, \tilde{E})$ by breaking each vertex $u \in X$ into a source-sink pair (s_u, t_u). Modify all edges entering u so that they enter t_u, and modify all edges leaving u so that they leave s_u. The capacity of an edge is equal to its weight in the subset feedback set problem. Also, connect t_u to s_u by an infinite capacity edge. Notice that we defined circular networks in a way that makes this reduction invertible. A SUBSET-FES instance is obtained from a circular network by coalescing all source-sink pairs.

2.3 Finding an optimal fractional solution

Feedback set problems belong to the class of *covering* problems, where the elements correspond to vertices or edges, and the subsets correspond to interesting cycles. A feedback set problem can be cast as a linear integer program, whose fractional relaxation can be solved efficiently. We describe the linear (integer) programming formulation for the case of a feedback edge problem. (The vertex case is similar.) There is a variable for each edge e denoted by $d(e)$. (The variable is $d(v)$ for $v \in V$ in the vertex case.)

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{e\in E} c(e)\cdot d(e) \\
\text{subject to} \quad & \sum_{e\in C} d(e) \geq 1 \text{ for every interesting cycle } C. \\
& d(e) \geq 0 \text{ for every edge } e \in E
\end{aligned}
$$

As can be seen from the formulation, in case we relax the integrality condition we get a fractional feedback edge set: a function $t : E \to [0,1]$ with the property that every interesting cycle is covered by t.

The (linear programming) dual of this covering problem is called a *packing* problem, and in the case of the feedback edge set problem, this means assigning a dual variable to all interesting cycles in the graph, such that for each edge, the sum of the variables corresponding to all the interesting cycles passing through that edge is at most the weight of the edge.

The linear program has an exponential number of constraints, but can be solved in polynomial time through either the Ellipsoid method, or interior point methods [NN94]. These methods can be applied here, since a violated constraint can be found in polynomial time.

As already mentioned, we can actually settle for an approximate solution of the above linear program. This can be achieved by either the algorithm described in Section 5, or through the methods of [PST91]. We note that Garg, Vazirani and Yannakakis [GVY93] provide an equivalent LP formulation with polynomially many constraints, however, their LP is not positive, and therefore, the techniques described in Section 5 are not applicable to this formulation.

2.4 The relation between directed multicut problems

Klein *et al.* [KPRT93] consider the following directed decomposition problem. Given a directed graph $G = (V, E)$ with edge capacities, and k terminal pairs (s_i, t_i), find a multicut that separates the pairs of terminals. Namely, after the removal of the edges of the multicut either s_i is not reachable from t_i, or vice versa. Klein *et al.* present an algorithm that finds an $O(\log^2 k)$ approximation of the minimum capacity of such a multicut.

The problem of finding minimum capacity multicuts in circular networks is easily reducible to the problem considered by [KPRT93], since there are infinite capacity edges from each sink to its corresponding source. Hence, the result presented in Section 3 can be obtained from the work of [KPRT93]. Similarly, our algorithm presented in Section 3 can be applied to the multicut problem considered by [KPRT93] to yield an alternative $O(\log^2 k)$ approximation algorithm. In fact, both algorithms can be viewed as extensions of the work of [GVY93] to directed graphs. In the sequel we show how to extend the algorithm presented in Section 4 to the multicut problem considered by [KPRT93]. This yields an $O(\min\{\log \tau^* \log\log \tau^*, \log n \log\log n\})$ approximation algorithm for this problem. In case $\min\{\log \tau^* \log\log \tau^*, \log n \log\log n\} = o(\log^2 k)$, this improves the $O(\log^2 k)$ approximation of [KPRT93].

3 The first SUBSET-FVS approximation algorithm

In this section we show how to find a subset feedback edge set of a weighted directed graph $G = (V, E)$, where the interesting cycles are defined by a set of special vertices $X \subseteq V$, such that $|X| = k$. The SUBSET-FES problem is considered for ease of presentation, but the reduction from the SUBSET-FVS problem to the SUBSET-FES problem described in Section 2.2 implies that the results presented in this section hold for the SUBSET-FVS problem as well. The weight of the feedback edge set found by the algorithm is $O(\tau^* \cdot \log^2 |X|)$, where τ^* is the weight of an optimal fractional feedback set. Therefore, we obtain an approximation factor which is independent of the weight of the optimum (fractional) feedback edge set.

We apply the reduction to the directed multicut problem in circular networks given in Section 2.2. Let $N = \left(\widetilde{V}, \widetilde{E}, c(\cdot), \{(s_i, t_i)\}_{i=1,\ldots,k}\right)$ be the directed network obtained by reducing the SUBSET-FES instance to a multicut problem instance. Let τ^* denote the cost of an optimal fractional multicut in N. We show how to find a multicut of cost $O(\tau^* \cdot \log^2 k)$.

3.1 Overview

Our algorithm uses the undirected sphere growing technique of Garg, Vazirani and Yannakakis [GVY93]. They used it to approximate the multicut problem in undirected networks. As we explain below, the application of this technique to the directed case does not seem to be trivial.

To gain some intuition, let us first describe the algorithm for the undirected case given by [GVY93]. The algorithm consists of two stages. In the first stage, the fractional multicut problem is solved (exactly) in polynomial time by any polynomial-time linear programming algorithm. We regard the output values, $d(e)$, as a distance function, where $d(e)$ denotes the distance between the tail and head of edge e. Note that the distance between s_i and t_i is at least one. In the second stage we grow radial spheres around the sources. We start from s_1 and grow a radial sphere around it. (A radial sphere of radius r includes the subgraph induced by all the vertices at distance at most r from s_1.) We look for the sphere with the smallest radius such that the cut separating it from the rest of the graph can be "charged" to the cut edges and the edges inside the sphere. (The exact notion of the charging scheme is omitted.) Garg, Vazirani and Yannakakis [GVY93] show that under an appropriate choice of parameters, the radius of the sphere is bounded by $1/2$. We add the cut corresponding to this sphere to the multicut. Note that this cut separates s_1 from t_1. Moreover, since no source-sink pair (s_i, t_i) can appear inside a sphere (because its radius is at most $1/2$, and the graph is undirected), in case there exist some index $1 \leq j \leq k$, such that either s_j or t_j is in the sphere, the respective source-sink pair is also separated. Thus, after taking this cut we can "throw away" all vertices in the sphere, and all edges that touch them. This ensures that we will not "charge" these edges in subsequent iterations, in which we grow spheres around other sources that are not yet separated.

Suppose that we wish to apply the same paradigm to the directed case. Consider a sphere of radius r around s_i. This sphere includes all vertices of distance r from s_i. In the application we face the following two problems: (1) The sphere may include a sink t_j whose source pair is outside the sphere. In order to separate s_i from t_i, and t_j from s_j, we have to take both incoming and outgoing cuts; i.e., the edges outgoing from, and the edges incoming to the sphere. However, we cannot "charge" the edges in the sphere for both cuts. (2) The sphere may include a source-sink pair (s_j, t_j), for $j \neq i$. In this case, even after separating s_i from t_i, we cannot "throw away" the vertices in the sphere. This may result in "charging" the same edges more than once.

We overcome these problems as follows. For the first problem, we note that in circular networks, all edges $e = t_i \rightarrow s_i$, for $i = 1, \ldots, k$, have infinite capacity edge. This implies that all these edges have *zero* length. Thus, while growing a radial sphere around a source we never encounter the situation where a sink is in the sphere while its source mate is outside. To overcome the second problem, we limit the number of times each edge is charged. This is done by growing disjoint spheres, one around s_i, and the other around t_i. (The latter is a "reversed" sphere.) We may choose either sphere for the cut, and we will choose the one that contains fewer source-sink pairs. This guarantees that an edge is charged only a logarithmic number of times.

Below, we describe the algorithm in detail. First, we describe how to grow the spheres. Then, we describe how to break the problem into small subproblems, such that each edge is "charged" at most a logarithmic number of times.

3.2 The sphere growing procedure

In this subsection, we present an adaptation of the region growing procedure of Garg, Vazirani and Yannakakis [GVY93]. The procedure of [GVY93] deals with undirected networks, whereas our procedure deals with directed circular networks (as defined in Section 2.1).

Let $\{d(e)\}_{e\in\widetilde{E}}$ denote an fractional multicut of the network N, and let τ^* denote the cost of this fractional multicut. We show how to grow a sphere from a source, s_{i_0}.

Regard the values, $d(e)$, as a distance function, where $d(e)$ denotes the distance between the tail and head of edge e. Define $dist(u,v)$ as the length of the shortest path from u to v. Order the vertices in ascending distance from s_{i_0}. Namely, $v_0 = s_{i_0}$, and $dist(v_0, v_{i+1}) \geq dist(v_0, v_i)$, for $1 \leq i < n$.

We use the following notation:

$$\begin{aligned} \ell_i &\triangleq dist(v_0, v_i) \\ A_i &\triangleq \{v_0, v_1, ..., v_i\} \end{aligned}$$

For a set A_i, $i \geq 0$, define the weight, $wt(A_i)$, as follows:

$$wt(A_i) \triangleq \frac{\tau^*}{k} + \sum_{e\in A_i\times A_i} c(e)\cdot d(e) + \sum_{e=(v_j\to v_k)\in A_i\times(\widetilde{V}-A_i)} c(e)\cdot(\ell_{i+1}-\ell_j)$$

The "credit" for the cut of the sphere A_i is given by $\varepsilon \cdot wt(A_i)$, for some parameter $\varepsilon > 0$ to be fixed in the analysis. Define a stopping index σ by:

$$\sigma \triangleq \min\left\{i : c(cut(A_i, \overline{A_i})) \leq \varepsilon \cdot wt(A_i)\right\}.$$

A_σ is the sphere found by the sphere growing procedure, since its cut can be charged to $\varepsilon \cdot wt(A_\sigma)$.

Claim 1: The radius of the sphere A_σ, denoted ℓ_σ, satisfies $\ell_\sigma < \ln(k+1)/\varepsilon$.

Proof: The claim holds trivially for $\sigma = 0$. For the analysis define $wt(A_{-1})$ to be τ^*/k. We get that $wt(A_0) \geq wt(A_{-1}) + (\ell_1)\cdot c(cut(A_0, \overline{A_0}))$. For $\sigma > 1$, we also get that for all $1 \leq i < \sigma$:

$$\begin{aligned} wt(A_i) = &= \frac{\tau^*}{k} + \sum_{e\in A_i\times A_i} c(e)\cdot d(e) + \sum_{e=(v_j\to v_k)\in A_i\times(\widetilde{V}-A_i)} c(e)\cdot(\ell_{i+1}-\ell_j) \\ &= \frac{\tau^*}{k} + \sum_{e\in A_{i-1}\times A_{i-1}} c(e)\cdot d(e) + \sum_{e\in A_{i-1}\times\{v_i\}} c(e)\cdot d(e) + \sum_{e\in\{v_i\}\times A_{i-1}} c(e)\cdot d(e) \\ &\quad + \sum_{e\in A_{i-1}\times(\widetilde{V}-A_i)} c(e = v_j \to v_k)\cdot(\ell_{i+1}-\ell_j) + \sum_{e\in\{v_i\}\times(\widetilde{V}-A_i)} c(e)\cdot(\ell_{i+1}-\ell_i) \end{aligned}$$

$$\begin{aligned}
\geq\ & \frac{\tau^*}{k} + \sum_{e \in A_{i-1} \times A_{i-1}} c(e) \cdot d(e) + \sum_{e \in A_{i-1} \times \{v_i\}} c(e) \cdot d(e) \\
& + \sum_{e=(v_j \to v_k) \in A_{i-1} \times (\widetilde{V} - A_i)} c(e) \cdot (\ell_i - \ell_j) \\
& + \sum_{e \in \{v_i\} \times (\widetilde{V} - A_i)} c(e) \cdot (\ell_{i+1} - \ell_i) + \sum_{e=(v_j \to v_k) \in A_{i-1} \times (\widetilde{V} - A_i)} c(e) \cdot (\ell_{i+1} - \ell_j)
\end{aligned}$$

Since $\ell_i - \ell_j \leq d(v_j \to v_i)$, and $\ell_{i+1} - \ell_j \geq \ell_{i+1} - \ell_i$ if $i \geq j$, it follows that:

$$wt(A_i) \geq wt(A_{i-1}) + (\ell_{i+1} - \ell_i) \cdot c(cut(A_i, \overline{A_i})). \tag{1}$$

The above inequality and our definition of $wt(A_{-1})$ imply that for $0 \leq i < \sigma$,

$$wt(A_i) > wt(A_{i-1}) + (\ell_{i+1} - \ell_i) \cdot \varepsilon \cdot wt(A_i) \tag{2}$$

Rearranging Equation (2) yields:

$$\begin{aligned}
wt(A_i) \quad & > \quad \frac{wt(A_{i-1})}{1 - (\ell_{i+1} - \ell_i) \cdot \varepsilon} && (3) \\
& \geq \quad wt(A_{-1}) \cdot \prod_{j=0}^{i} (1 - (\ell_{j+1} - \ell_j) \cdot \varepsilon)^{-1} \\
& = \quad \frac{\tau^*}{k} \cdot \prod_{j=0}^{i} (1 - (\ell_{j+1} - \ell_j) \cdot \varepsilon)^{-1}
\end{aligned}$$

Substituting $\sigma - 1$ with i in Equation (3) yields,

$$wt(A_{\sigma-1}) > \frac{\tau^*}{k} \cdot \prod_{j=0}^{\sigma-1} (1 - (\ell_{j+1} - \ell_j) \cdot \varepsilon)^{-1} \tag{4}$$

Since $\tau^* \cdot (1 + 1/k) \geq wt(A_n)$, and since $wt(A_n) \geq wt(A_{\sigma-1})$

$$k + 1 > \prod_{j=0}^{\sigma-1} (1 - (\ell_{j+1} - \ell_j) \cdot \varepsilon)^{-1} \tag{5}$$

Take a natural logarithm from both sides of Equation (5):

$$\ln(k+1) > -\sum_{j=0}^{\sigma-1} \ln\left(1 - (\ell_{j+1} - \ell_j) \cdot \varepsilon\right)$$

From Equation (2) it follows that $(\ell_{j+1} - \ell_j) \cdot \varepsilon < 1$. Since $-\ln(1-x) \geq x$, for $0 \leq x < 1$,

$$\ln(k+1) > \sum_{j=0}^{\sigma-1} (\ell_{j+1} - \ell_j) \cdot \varepsilon \geq \varepsilon \cdot (\ell_\sigma - \ell_0) = \varepsilon \cdot \ell_\sigma. \tag{6}$$

The claim follows. □

3.3 Computing the multicut

We show how to use the sphere growing procedure to compute the multicut. This is done by breaking up the problems into subproblems, each of size at most $(k-1)/2$. We set the sphere growing parameter to $\varepsilon = 2\ln(k+1)$. Using this parameter, we grow a sphere around s_1. Recall that if this sphere includes a sink t_i, it also includes its mate s_i. We also grow another "reversed" sphere around t_1. (A "reversed" sphere is a sphere computed in the network given by flipping the directions of the edges.) Note that if this sphere includes a source s_i, it also includes its mate t_i. Since the radius of each such sphere is less than $1/2$, they are disjoint. Hence, one of them includes less than $(k-1)/2$ source-sink pairs. Suppose that this is the sphere around s_1. The other possibility is analogous. We add the edges outgoing from this sphere to the multicut. This separates all sources in the sphere whose mate is not in this sphere. To separate the pairs included in the sphere we have to solve the subproblem given by the network induced by the vertices inside the sphere. This subproblem has no more than $(k-1)/2$ source-sink pairs. Now, if there are more than $(k-1)/2$ source-sink pairs outside the sphere, we choose one such pair and repeat the process. When we end the process we are left with subproblems each of which consists of at most $(k-1)/2$ pairs and whose total size is bounded by the size of the original problem. For a detailed description of the process see Figure 1.

As explained above we associate a cut $\gamma(R_j)$ with each selected sphere R_j. Recall that if R_j is a sphere around a source, then $\gamma(R_j) = cut(R_j, \overline{R_j})$, and if R_j is a "reversed" sphere around a sink, then $\gamma(R_j) = cut(\overline{R_j}, R_j)$. Let $\{R_j\}_j$ denote all selected spheres computed recursively, until no sphere contains a source-sink pair. The set of edges $\cup_j \gamma(R_j)$ is a multicut.

We analyze the cost of the multicut, $\cup_j \gamma(R_j)$. Fix a network, N, with n vertices and k source-sink pairs. Let $cut(n,k)$ denote the maximum cost of the multicut found by our procedure. Let $R_1, \ldots, R_r$ denote the spheres selected in the top level of the process (i.e., in breaking into subproblems with no more than $(k-1)/2$ pairs). Let n_j denote the number of vertices in R_j, and k_j denote the number of source-sink pairs in R_j. Then, $cut(n,k)$

```
breaking-iteration (N, k, {d(e)}_{e∈N})
    N' = N
    k' = k
    i = 0
    while k' > (k − 1)/2 do
        begin
        i = i + 1
        Let (s_j, t_j) denote a source-sink pair in N'.
        Compute a sphere around s_j and a "reversed" sphere around t_j.
        Let R_i be the sphere with the least number of source-sink pairs.
        N' = the network induced by the vertices of N that do not belong to R_i.
        k' = the number of source-sink pairs in N'.
        end
    i = i + 1
    R_i = the vertices in N'.
    Return R_1, R_2, ..., R_i
```

Figure 1: Breaking into subproblems

satisfies the following recurrence inequality:

$$cut(n,k) \leq \begin{cases} 0 & \text{if } k = 0 \\ \sum_{j=1}^{r} c(\gamma(R_j)) + \sum_{j=1}^{r} cut(n_j, k_j) & \text{otherwise} \end{cases} \tag{7}$$

Where $\sum_{j=1}^{r} n_j = n$, $\sum_{j=1}^{r} k_j = k - r$, and $0 \leq k_j < k/2$, for every j.

Claim 2: If $\varepsilon = 2\ln(k+1)$, then $cut(n,k) \leq 2 \cdot \ln(2) \cdot \tau^* \cdot \log^2(k+1)$.

Proof: Let $\alpha = 2 \cdot \ln(2)$. It is sufficient to show that the following recurrence inequality holds:

$$\alpha \cdot \tau^* \cdot \log^2(k+1) \geq \begin{cases} 0 & \text{if } k = 0 \\ \sum_{j=1}^{r} c(\gamma(R_j)) + \sum_{j=1}^{r} \alpha \cdot \tau_j^* \cdot \log^2(k_j+1) & \text{otherwise} \end{cases} \tag{8}$$

where τ_j^* is the capacity optimum cost of the multicut in the subproblem defined by sphere R_j.

Equation (8) holds trivially for $k = 0$. Suppose that $k > 0$. Each R_j is constructed by the sphere growing procedure, and it follows that $c(\gamma(R_j)) \leq \varepsilon \cdot wt(R_j)$. Since the spheres $R_1, \ldots, R_r$ are disjoint, $\sum_{j=1}^{r} \tau_j^* \leq \tau^*$. Since $k_j \leq (k-1)/2$, it follows that $\log^2(k_j +$

$1) \leq \log^2((k+1)/2)$. These inequalities imply that

$$\sum_{j=1}^{r} c(\gamma(R_j)) + \sum_{j=1}^{r} \alpha \cdot \tau_j^* \cdot \log^2(k_j+1) \leq \varepsilon \cdot \sum_{j=1}^{r} wt(R_j) + \alpha \cdot \tau^* \cdot \log^2((k+1)/2) \quad (9)$$

It is sufficient to prove that $\alpha \cdot \tau^* \cdot \log^2(k+1)$ is not less than the right hand side of Equation (9). Note that $\sum_{j=1}^{r} wt(R_j) \leq \tau^* + r \cdot \tau^*/k \leq 2 \cdot \tau^*$. Therefore, it is sufficient to show that

$$\alpha \cdot \tau^* \cdot \log^2(k+1) \geq 2 \cdot \varepsilon \cdot \tau^* + \alpha \cdot \tau^* \cdot \log^2((k+1)/2) \quad (10)$$

Equation (10) holds if $\alpha \geq 2 \cdot \varepsilon / (\log^2(k+1) - \log^2((k+1)/2))$. Substituting $\varepsilon = 2\ln(k+1)$, and $\alpha = 2 \cdot \ln(2)$, the claim follows. □

We summarize the approximation factor obtainable by our procedure.

Theorem 3: If $\varepsilon = 2 \cdot \ln(k+1)$, then $cut(n,k) \leq 1.39 \cdot \tau^* \cdot \log^2(k+1)$.

4 The second SUBSET-FVS approximation algorithm

In this section we describe an algorithm for approximating the minimum weight SUBSET-FVS of a weighted directed graph $G = (V, E)$. The reductions from SUBSET-FES to SUBSET-FVS shown in Section 2.2 imply that the results of this section hold for the SUBSET-FES problem as well. The algorithm presented in this section is an algorithmic adaptation of Seymour's paper [S92], which we extend to the weighted subset feedback set problem. The presented algorithm is not the most efficient implementation of Seymour's ideas, however, it is simpler than the more efficient implementation presented in [ENS95].

Let $d(v)$ denote the fractional value of vertex $v \in V$ in the solution produced by the linear programming formulation of the SUBSET-FVS problem, and let τ^* denote the value of the optimal fractional SUBSET-FVS. Without loss of generality we can assume that $d(v) = 0$, for all special vertices $v \in X$. (This can be obtained by splitting special vertices into two vertices.)

4.1 Modifying the graph

We first show how to modify the fractional solution so that all values attached to vertices are integral multiples of $1/2n$. This modification increases the cost of the fractional solution by at most a factor of 4. We define the following fractional SUBSET-FVS d':

$$d'(v) = \begin{cases} 0 & \text{if } d(v) < 1/2n \\ 2 \cdot d(v) & \text{otherwise.} \end{cases}$$

The following proposition is immediate.

Proposition 4: The function d' is a feasible SUBSET-FVS and its weight is at most $2\tau^*$.

We now round up each value of d' to the nearest integral multiple of $1/2n$. This clearly preserves feasibility, and doubles the cost by at most a factor of two. Let us denote the new fractional SUBSET-FVS by d''. Its cost is at most $4\tau^*$. Since these modifications in the fractional solution changed the value of the fractional feedback set by only a constant multiplicative factor, we can henceforth rename d'' and call it d, and also let τ^* be the changed value of the fractional feedback set.

We now modify the graph as follows. All non-special vertices v for which $d(v) = 0$ are marked as blackout vertices, and we delete them from the graph by using the reduction BLACKOUT-FVS $\preceq$ FVS described in section 2.2.

Next, we "equalize" the graph G into graph H as follows. For each non-special vertex v, we replace it by a directed chain of vertices $v_1 \to \ldots \to v_\ell$ where $\ell = d(v) \cdot 2n$. By the above discussion, ℓ is a positive integer. Set the weight of each vertex v_i, $1 \leq i \leq \ell$, to $c(v)$. For each edge $u \to v \in E(G)$, we add a directed edge from the last vertex of u's chain to the first vertex of v's chain. The *girth* of a graph is defined to be the length of a shortest cycle in the graph. Here, we modify the definition of the girth with respect to the graph H as follows: the girth of H, denoted by $\mathrm{girth}(H)$, equals the minimum, over all interesting cycles, of the number of non-special vertices in the cycle. The next lemma summarizes the properties of the graph H.

Lemma 5: [S92] Graph H has the following properties:
(i) There is a 1-1 correspondence between the interesting cycles of G and H. There is also a weight preserving correspondence between the (fractional) feedback vertex sets of G and H.
(ii) The girth of H satisfies:

$$\mathsf{girth}(H) \geq \frac{wt(V(H))}{\tau^*}, \text{ where } wt(V(H)) \text{ is the total weight of the vertices in } H.$$

Proof: Property (i) follows from the "equalizing" transformation. Any SUBSET-FVS of G can be transformed into an SUBSET-FVS of H by assigning the value of each vertex $v \in V(G)$ to the first vertex in v's chain. All other vertices are assigned value zero. Conversely, given an SUBSET-FVS of H, construct an SUBSET-FVS of G by assigning each vertex $v \in V(G)$ the sum of the values of the vertices in its corresponding chain in H.

Property (ii) follows by observing that

$$wt(V(H)) = \sum_{v \in V(G)} c(v) \cdot d(v) \cdot 2n = 2n \cdot \tau^*$$

and that each interesting cycle $C \in H$ contains at least

$$\sum_{v \in \bar{C}} d(v) \cdot 2n \geq 2n$$

non-special vertices, where $\bar{C}$ is the cycle in G corresponding to C. Since C is an arbitrary interesting cycle in H

$$\text{girth}(H) \geq \frac{wt(V(H))}{\tau^*}$$

□ The above lemma implies that the problem of finding an SUBSET-FVS in an arbitrary graph G is equivalent to finding an SUBSET-FVS in a graph H with the property that its girth, and the weight of its optimal (fractional) SUBSET-FVS are inversely related. Notice that by assigning each non-special vertex in H the value $1/\text{girth}(H)$, we obtain a near-optimal fractional SUBSET-FVS of H. Therefore, we consider H to be "equalized".

4.2 The algorithm

We present the algorithm for approximating the SUBSET-FVS problem in the graph H. Seymour [S92] defines the function μ as follows:

$$\mu(x) = \begin{cases} 0 & \text{if } x < 1 \\ 4x \cdot \ln(4x) \cdot \ln\log(4x) & \text{otherwise.} \end{cases}$$

The algorithm for approximating the optimal weighted SUBSET-FVS proceeds recursively as follows. The graph is first partitioned into strongly connected components. If a strongly connected component lacks vertices of X, then an empty set is a valid feedback vertex set with respect to this component. Otherwise, a directed separator S is computed (separately) for all strongly connected components containing at least one vertex of X. We henceforth assume that H us strongly connected. A directed separator partitions the vertices of H into three disjoint subsets: A, S, and B, such that there are no directed edges from A to B. The directed separator is computed by the procedure *separator* $(H = (V, E), c(\cdot), g)$ depicted in Figure 2. (The parameters to this procedure are the graph H, the weight function $c(\cdot)$, and a lower bound on the girth of H). We add the vertices of S to the SUBSET-FVS, and recursively compute a SUBSET-FVS of the subgraphs induced by A and B. In Procedure *separator*, we perform a Breadth-First-Search (BFS) starting from an special vertex v_0. The distance $dist(v_0, u)$ used in the BFS is the minimum, over all paths from v_0 to u, of the number of non-special vertices along a path. It is important that in Procedure *separator*, the BFS starts from a special vertex. This ensures that the BFS contains at least g layers.

```
separator (H = (V, E), c(·), g)
    Choose a vertex v_0 ∈ X.
    Construct "layers" starting from v_0 as follows:
        For i = 1, ..., (g − 1) define
            L_i ≜ {u ∈ V − X : dist(v_0, u) = i}.
        L_g ≜ V(H) − ∪_{j<g} L_j.
    i = 0
    repeat
        i = i + 1
        A_i ≜ {u ∈ V : dist(v_0, u) < i}.
        B_i ≜ V − A_i − L_i.
    until wt(L_i) ≤ μ(wt(V(H))/g) − μ(wt(A_i)/g) − μ(wt(B_i)/g).
    return(A_i, L_i, B_i)
```

Figure 2: The directed separator algorithm.

Theorem 6: Suppose that the girth of graph H is at least g. The algorithm finds a SUBSET-FVS of H whose weight is at most $\mu\left(\frac{wt(V(H))}{g}\right)$.

Combining Theorem 6 with Lemma 5 and setting $g = girth(H)$ yield the following corollary.

Corollary 7: Let H denote the directed graph obtained by modifying the graph G as described in subsection 4.1. Let $U \subseteq V(H)$ denote the SUBSET-FVS of H, found by the algorithm. Let $\bar{U} \subseteq V(G)$ denote the SUBSET-FVS of G that corresponds to U. Then,

$$wt(\bar{U}) \leq \mu(\tau^*) \leq 4\tau^* \cdot \ln(4\tau^*) \cdot \ln\log(4\tau^*).$$

Hence, an $O(\log \tau^* \log\log \tau^*)$-approximation algorithm is obtained.

4.3 Proof of Theorem 6

We begin with the following lemma of Seymour [S92, Lemma 2.3].

Lemma 8: Let $\ell > 0$ be a real number. For $0 \leq x \leq 1$, let $y(\cdot)$ be a real-valued continuous function of x, such that $y(0) \geq 0$, $y(1) \leq 1$, and for all $h \in [0,1] - I$, where $I \subseteq [0,1]$ is some finite subset of $[0,1]$, $y(\cdot)$ is differentiable, and $\left.\frac{dy}{dx}\right|_{x=h} \geq \frac{1}{\ell}$. Then, there exists h with $1/4 < h < 3/4$, $h \notin I$, such that

$$\ell \left.\frac{dy}{dx}\right|_{x=h} \leq \mu(\ell) - \mu(\ell y) - \mu(\ell(1-y)).$$

The following claim establishes that procedure *separator* $(H = (V, E), c(\cdot), g)$ is successful in finding a directed separator. We use the same notation used in the procedure.

Claim 9: There exists a layer, L_i, that satisfies:

$$wt(L_i) \leq \mu\left(\frac{wt(V(H))}{g}\right) - \mu\left(\frac{wt(A_i)}{g}\right) - \mu\left(\frac{wt(B_i)}{g}\right).$$

Proof: Define the function $y(x)$ in the interval $[0,1]$ as follows:

$$y(x) \triangleq \frac{1}{wt(V(H))} \cdot (wt(A_i) + (gx - i) \cdot wt(L_i))$$

where $i = \lfloor gx \rfloor + 1$.

The function $y(x)$ satisfies the requirements of Lemma 8, for $\ell = wt(V(H))/g$. (Recall that all weights are greater than 1.) Therefore, there exists an $h \in (1/4, 3/4)$, $h \notin \{1/g, 2/g, \ldots, 1\}$ for which

$$\ell \left.\frac{dy}{dx}\right|_{x=h} \leq \mu(\ell) - \mu(\ell y) - \mu(\ell(1-y)).$$

Define $i = \lfloor gh \rfloor + 1$. We claim that the following three equations hold:

$$wt(L_i) = \ell \left.\frac{dy}{dx}\right|_{x=h} \tag{11}$$

$$\frac{wt(A_i)}{g} \leq \ell \cdot y(h) \tag{12}$$

$$\frac{wt(B_i)}{g} \leq \ell \cdot (1 - y(h)) \tag{13}$$

Proof of (11) follows from the fact that $\frac{dy}{dx}|_{x=h} = \frac{g \cdot wt(L_i)}{wt(V(H))}$, and from the definition of ℓ.

Proofs of (12) and (13) follow from the observation that

$$wt(A_i) \leq y(h) \cdot wt(V(H)) \leq wt(V(H)) - wt(B_i).$$

Since the function $\mu(\cdot)$ is monotone non-decreasing, the claim follows. $\square$

We have established that procedure *separator* $(H = (V, E), c(\cdot), g)$ is successful in finding a directed separator.

The proof of the theorem follows by induction on $|V(H)|$. The induction basis is immediate. Assume that it holds for graphs with less than $|V(H)|$ vertices. Since the separator S decomposes the graph into two subgraphs A and B, we obtain the following recurrence:

$$wt(\text{SUBSET-FVS}(H)) \leq wt(\text{SUBSET-FVS}(A)) + wt(\text{SUBSET-FVS}(B)) + wt(S),$$

where SUBSET-FVS(G) denotes the subset feedback vertex set found by the algorithm on graph G. Since $\text{girth}(A) \geq \text{girth}(H)$, and $|V(A|) < |V(H)|$, it follows by the induction hypothesis that $wt(\text{SUBSET-FVS}(A)) \leq \mu\left(wt(A)/g\right)$. Similarly, $wt(\text{SUBSET-FVS}(B)) \leq \mu\left(wt(B)/g\right)$. Therefore,

$$wt(\text{SUBSET-FVS}(A)) + wt(\text{SUBSET-FVS}(B)) + wt(S) \leq \mu\left(wt(A)/g\right) + \mu\left(wt(B)/g\right) + wt(S).$$

By the definition of S, it follows that the right hand side of the previous equation is bounded by $\mu\left(\frac{wt(V(H))}{g}\right)$. Consequently,

$$wt(\text{SUBSET-FVS}(H)) \leq \mu\left(\frac{wt(V(H))}{g}\right)$$

and the theorem follows. $\square$

The role of the function $y(x)$ is very similar to the region growing procedures in the works of [LR88, KARR90, GVY93]. Seymour's refined analysis improves the stopping condition, and the recurrence equation obtained by the decomposition scheme has a smaller upper bound.

4.4 Strongly polynomial approximation bounds and running times

The approximation bounds obtained in Corollary 7 may be very weak, in cases where the cost of an optimal fractional subset feedback vertex set, τ^*, is super-polynomial in $n = |V(G)|$. To overcome this problem we "truncate" the vertex weights to obtain a SUBSET-FVS of weight $O(\tau^* \cdot \ln n \cdot \log\log n)$.

We truncate the vertex weights as follows. Let $\text{IN} \triangleq \{v \in V : c(v) \leq \tau^*/n\}$ and let $\text{OUT} \triangleq \{v \in V : c(v) > \tau^* \cdot n\}$. We add all vertices of IN to the SUBSET-FVS. This increases the total weight of the approximate SUBSET-FVS by at most τ^*. All non-special vertices of OUT are marked as blackout vertices. This can be done since it is guaranteed that none of the vertices in OUT participates in an integral optimal SUBSET-FVS. The latter follows by observing that the ratio between the values of an optimal integral SUBSET-FVS and an optimal fractional SUBSET-FVS is at most n. (Given a fractional SUBSET-FVS, if one rounds up to 1 all values that are bigger than or equal to $1/n$, and rounds down to 0 all other values, then a feasible integral SUBSET-FVS is obtained).

In the remaining graph the ratio between the maximum and minimum weights of vertices is bounded by n^2. Hence, after normalizing the vertex weights, the cost of an optimal fractional SUBSET-FVS is a polynomial in n, and the modified (equalized) graph, H, is also of polynomial size. Therefore, the algorithm finds an SUBSET-FVS of weight at most $O(\tau^* \ln n \log\log n)$.

4.5 Finding multicuts that separate pairs of terminals

We now turn to the decomposition problem considered in Klein *et al.* [KPRT93] defined as follows. Given a directed graph $G = (V, E)$ with edge capacities, and k terminal pairs (s_i, t_i), find a multicut that separates the pairs of terminals. Namely, after the removal of the edges of the multicut either s_i is not reachable from t_i, or vice versa. Klein *et al.* present an algorithm that finds an $O(\log^2 k)$ approximation of the minimum capacity of such a multicut. The algorithm given in this section can be extended to get an $O(\min\{\log \tau^* \log\log \tau^*, \log n \log\log n\})$ approximation algorithm for this decomposition problem as follows.

First, reduce the multicut problem to a version of the SUBSET-FVS problem in which the interesting cycles are cycles that pass through a pair of terminals. (Note that our algorithm does not consider this version of the SUBSET-FVS problem.) Let τ^* denote the optimal fractional solution of this SUBSET-FVS problem. Equalize the graph as described in Subsection 4.1 to obtain an inverse relation between the length of the interesting cycles and the cost τ^*. Let g denote the length of the shortest interesting cycle after equalizing the graph. The decomposition of the graph proceeds as follows:

Repeat as long as there exists a terminal pair such that both its terminals are in the same strongly connected component of the graph. Pick such a terminal pair (s_i, t_i).

Assume that the shortest path from s_i to t_i contains $g/2$ non-terminal vertices. (Either this holds or the shortest path from t_i to s_i contains $g/2$ vertices.) Run the separator algorithm starting from s_i with $g/2$ layers to find a subset of vertices L_j whose removal disconnects all paths from s_i to t_i.

It is not difficult to see that the same analysis holds for this version of the algorithm also, and hence, an $O(\min\{\log \tau^* \log\log \tau^*, \log n \log\log n\})$ approximation algorithm for this problem is obtained.

5 Approximating the fractional optimal solution

In this section we show that an approximation by a $(1+\varepsilon)$ factor to the fractional FVS and SUBSET-FVS problems can be computed efficiently by a combinatorial algorithm. Our algorithm is based on a $(1+\varepsilon)$-approximation algorithm for an arbitrary Set Cover problem presented in Section 5.1 which is derived from a parallel algorithm for approximating positive linear programming by Luby and Nisan [LN93]. An important feature of the Set Cover algorithm is that the dependence of its complexity on the number of sets in the set system is only logarithmic.

In the Set Cover problem the input consists of a universal set of elements and a collection of subsets of elements. A non-negative cost is associated with each element, and the goal is to find a minimum cost set of elements that intersects (or covers) all subsets. We represent Feedback Set problems as Set Cover problems by viewing the cycles as the subsets in the set system, and the vertices as elements.

In Section 5.2, we present an implementation of the Set Cover algorithm for Feedback Set problems that uses a succinct representation of the set of cycles in the graph, which might be exponentially large. The algorithm is presented for the fractional FVS problem. However, it can be easily modified to handle the covering problem associated with a SUBSET-FVS problem.

5.1 A $(1+\varepsilon)$-approximation Set Cover algorithm

The Set Cover Problem is defined as follows. Let $(V, \mathcal{S}, c(\cdot))$ be a set system, where V is a universal set, $\mathcal{S}$ is a collection of subsets of elements from V, and $c(\cdot)$ is a non-negative cost function associated with each element of V. Let $|V| = n$ and $|\mathcal{S}| = m$. The goal is to find a *fractional* minimum cost subset of V that covers all the subsets in $\mathcal{S}$. Let τ^* denote the cost of an optimal fractional cover.

The algorithm we present associates an *attraction function* $p(\cdot)$ with each subset, where initially, $p(S) = 1$ for each $S \in \mathcal{S}$. The attraction of the set system is denoted by $p(\mathcal{S})$ and is equal to $\sum_{S\in\mathcal{S}} p(S)$. Define the attraction $p(v)$ of an element $v \in V$ to be $\sum_{S:v\in S} p(S)$.

The *approximate_SC* algorithm depicted in Figure 3 computes a multi-cover of $\mathcal{S}$, i.e., a cover that may contain an element several times. Let $\ell(v)$ denote the multiplicity of element $v \in V$ in the multi-cover computed by the algorithm. The fractional cover is derived by normalizing the values of $\ell(v)$ as follows:

$$\varphi(v) \triangleq \ell(v) \cdot \frac{\ln \lambda_1}{\lambda_2 \ln m}$$

approximate_SC $(V, \mathcal{S}, c(\cdot), \varepsilon)$
If $\varepsilon > 1$ then $\varepsilon = 1$.
Define: $\lambda_1 \triangleq 1 + \varepsilon/4$ and $\lambda_2 = 4/\varepsilon$
$\forall v \in V : \ell(v) = 0$.
$\forall S \in \mathcal{S} : p(S) = 1$.
repeat
Choose an element in $\{v \in V : \ p(v) > m^{-\lambda_2}\}$ that minimizes the ratio $\frac{c(v)}{p(v)}$.
$\ell(v) \leftarrow \ell(v) + 1$
$\forall S$ such that $v \in S$: $p(S) \leftarrow p(S)/\lambda_1$.
until $p(v) \leq m^{-\lambda_2}$ for every $v \in V$.
return($\{\ell(v) : v \in V\}$)

Figure 3: The integral multi cover approximation algorithm.

Note that the stopping condition of the repeat loop can be replaced by $p(S) \leq m^{-\lambda_2}$, for every $S \in \mathcal{S}$. Due to efficient implementation considerations, as described in Section 5.2, we choose the stricter stopping condition.

Theorem 10: The following claims hold:
(i) The function φ is a feasible cover of $\mathcal{S}$.
(ii) The cost of the function φ is at most $(1 + \epsilon)$ times the cost of an optimal fractional cover, namely, $\sum_{v \in V} \varphi(v) \cdot c(v) \leq (1 + \varepsilon) \cdot \tau^*$.
(iii) The number of iterations of the algorithm is $O\left(\varepsilon^{-2} \cdot n \ln m\right)$.

Proof: We first show that φ is indeed a feasible cover. Consider a subset S and an element $v \in S$. When Algorithm *approximate_SC* terminates, $p(S) \leq p(v) \leq m^{-\lambda_2}$. Thus, each subset $S \in \mathcal{S}$ is covered at least $\frac{\lambda_2 \ln m}{\ln \lambda_1}$ times, i.e., $\sum_{v : v \in S} \ell(v) \geq \frac{\lambda_2 \ln m}{\ln \lambda_1}$. Consequently, φ defines a feasible cover for $\mathcal{S}$.

To prove (ii) and (iii), we first show that

$$\sum_{v \in V} c(v) \cdot \ell(v) \leq \frac{\lambda_1}{\lambda_1 - 1} \left[(\lambda_2 + 1) \ln m + \ln \lambda_1\right] \tau^* \tag{14}$$

Let $p_i(S)$ and $p_i(v)$ denote the attraction of subset S and element v in the beginning of the ith iteration of the algorithm. Suppose v_i is the element chosen in the ith iteration. We claim that

$$\frac{c(v_i)}{p_i(v_i)} \leq \frac{\tau^*}{p_i(\mathcal{S})} \tag{15}$$

First, consider the sum $\sum_v \tau^*(v) p_i(v)$,

$$\begin{aligned}\sum_v \tau^*(v) p_i(v) &= \sum_v \left(\tau^*(v) \sum_{S: v\in S} p_i(S)\right) \\ &= \sum_S \left(p_i(S) \sum_{v\in S} \tau^*(v)\right) \\ &\geq p_i(\mathcal{S})\end{aligned}$$

Now, Equation (15) is proved as follows:

$$\begin{aligned}\frac{\tau^*}{p_i(\mathcal{S})} &\geq \frac{\sum_v \tau^*(v) c(v)}{\sum_v \tau^*(v) p_i(v)} \\ &\geq \min_v \frac{\tau^*(v) c(v)}{\tau^*(v) p_i(v)} \\ &= \frac{c(v_i)}{p_i(v_i)}\end{aligned}$$

We now bound the ratio between $p_i(\mathcal{S})$ and $p_{i+1}(\mathcal{S})$ at each iteration i. The decrease in $p_i(\mathcal{S})$ is precisely $\left(1 - \frac{1}{\lambda_1}\right) \cdot p_i(v_i)$, and hence,

$$\frac{p_{i+1}(\mathcal{S})}{p_i(\mathcal{S})} = \frac{p_i(\mathcal{S}) - \left(1 - \frac{1}{\lambda_1}\right) \cdot p_i(v_i)}{p_i(\mathcal{S})} = 1 - \frac{(\lambda_1 - 1) \cdot p_i(v_i)}{\lambda_1 \cdot p_i(\mathcal{S})} \leq 1 - \frac{(\lambda_1 - 1) \cdot c(v_i)}{\lambda_1 \cdot \tau^*}, \qquad (16)$$

where the last inequality follows from (15).

Let I denote the number of iterations of the algorithm, and let $\bar{c} = \sum_{i=1}^{I} c(v_i)/I$. We claim that, for every $\rho > 0$,

$$\prod_{i=1}^{I} \left(1 - \frac{c(v_i)}{\rho \cdot \tau^*}\right) \leq \left(1 - \frac{\bar{c}}{\rho \cdot \tau^*}\right)^I \qquad (17)$$

To prove Equation (17) we consider the logarithms of both sides, and divide both sides by I. The left hand side equals then,

$$\frac{1}{I} \cdot \log\left(\prod_{i=1}^{I}\left(1 - \frac{c(v_i)}{\rho \cdot \tau^*}\right)\right) = \frac{1}{I} \cdot \sum_{i=1}^{I} \log\left(1 - \frac{c(v_i)}{\rho \cdot \tau^*}\right)$$

and by Jensen's Inequality,

$$\frac{1}{I} \cdot \sum_{i=1}^{I} \log\left(1 - \frac{c(v_i)}{\rho \cdot \tau^*}\right) \leq \log\left(\frac{\sum_{i=1}^{I}\left(1 - \frac{c(v_i)}{\rho \cdot \tau^*}\right)}{I}\right) = \log\left(1 - \frac{\sum_{i=1}^{I} c(v_i)}{\rho \cdot \tau^* \cdot I}\right) = \log\left(1 - \frac{\bar{c}}{\rho \cdot \tau^*}\right),$$

which finishes the proof of Equation (17).

By the initial assignment to $p(S)$, it follows that $p_1(\mathcal{S}) = m$. Upon termination, $p_{I+1}(S) > m^{-\lambda_2}/\lambda_1$, for every set S. Hence,

$$\frac{1}{\lambda_1 m^{\lambda_2+1}} \leq \frac{p_{I+1}(\mathcal{S})}{p_1(\mathcal{S})}$$

We upper bound the same ratio using Equations (16) and (17), where $\rho = \frac{\lambda_1}{\lambda_1-1}$:

$$\frac{p_{I+1}(\mathcal{S})}{p_1(\mathcal{S})} = \prod_{i=1}^{I} \frac{p_{i+1}(\mathcal{S})}{p_i(\mathcal{S})} \leq \prod_{i=1}^{I} \left(1 - \frac{(\lambda_1 - 1)c(v_i)}{\lambda_1 \tau^*}\right) \leq \left(1 - \frac{(\lambda_1 - 1)\bar{c}}{\lambda_1 \tau^*}\right)^I.$$

We get

$$\frac{1}{\lambda_1 m^{\lambda_2+1}} \leq \left(1 - \frac{(\lambda_1 - 1)\bar{c}}{\lambda_1 \tau^*}\right)^I.$$

Take the natural logarithm of both hands. Because of Equation (15) $\frac{c(v_i)}{\tau^*} \leq 1$, for all $1 \leq i \leq n$. Hence, $0 < \frac{(\lambda_1-1)\bar{c}}{\lambda_1 \tau^*} < 1$, and we can apply the inequality $x \leq -\ln(1-x)$, for $0 \leq x < 1$, to get that

$$(\lambda_2 + 1)\ln m + \ln \lambda_1 \geq I \cdot \frac{(\lambda_1 - 1)\bar{c}}{\lambda_1 \tau^*}$$

Rearranging the terms we get

$$\sum_{i=1}^{I} c(v_i) = I \cdot \bar{c} \leq \frac{\lambda_1}{\lambda_1 - 1} \left[(\lambda_2 + 1)\ln m + \ln \lambda_1\right] \tau^*$$

which proves Equation (14). The fractional cover, φ, is obtained by dividing the multiplicity, $\ell(v)$, by $\frac{\lambda_2 \ln m}{\ln \lambda_1}$, for every element v. The resulting bound on the cost of the fractional cover is

$$\varphi \leq \sum_{i=1}^{I} c(v_i) \cdot \frac{\ln \lambda_1}{\lambda_2 \ln m} \leq \frac{\ln \lambda_1}{\lambda_2 \ln m} \cdot \frac{\lambda_1}{\lambda_1 - 1} \left[(\lambda_2 + 1)\ln m + \ln \lambda_1\right] \tau^*$$

Since $(\ln \lambda_1) \leq (\lambda_1 - 1)$, it follows that

$$\frac{\varphi}{\tau^*} \leq \lambda_1 \left(1 + \frac{1}{\lambda_2} + \frac{\ln \lambda_1}{\lambda_2 \ln m}\right)$$

Substituting $\lambda_1 = 1 + \varepsilon/4$ and $\lambda_2 = 4/\varepsilon$, we get the bound of (ii) for $\varepsilon \leq 1$, i.e., $\varphi \leq (1+\varepsilon)\tau^*$.

Each element is chosen at most $\frac{\lambda_2 \ln m}{\ln \lambda_1} + 1$ times. Since $\ln \lambda_1 \geq \varepsilon/8$, if $\varepsilon \leq 1$, the total number of iterations is $O\left(\varepsilon^{-2} \cdot n \ln m\right)$, as required.

5.2 Implementing the fractional FVS algorithm

The FVS problem naturally reduces to a Set Cover problem by defining the vertices of the graph as elements and the simple directed cycles as the subsets of the set system. However, a naive implementation of algorithm *approximate_SC* would take exponential time, since the size of the set system might be exponential, and the algorithm requires attaching an "attraction" $p(S)$ to every subset S, updating these attractions, and calculating $p(v) = \sum\{p(S)|v \in S\}$, for every vertex v. In this section, we present a more involved implementation that runs in polynomial time.

First, we modify the input graph $G = (V, E)$ by splitting each vertex $v \in V$ into a "left" vertex v' and a "right" vertex v'', where the incoming edges into v now enter v', and the outgoing edges from v now leave v''. In addition there is a directed edge from v' to v'', and a self loop on v''. The new graph is denoted by $G' = (V', E')$.

We redefine the Set Cover problem that corresponds to the FVS problem on G. Consider the set of all simple and non-simple directed cycles in G' of length $|V'|$ that contain at least one "left" vertex. Every such cycle in G' corresponds to a simple directed cycle in G by eliminating the self loops and merging all adjacent "left" and "right" vertices. (In case the resulting cycle is non-simple, pick any simple sub-cycle in it.) Conversely, every simple directed cycle in G has at least one corresponding cycle of length $|V'|$ in G' that contains a "left" vertex. Clearly, this correspondence is not one-to-one, and cycles in G may have several corresponding closed paths in G'. Nevertheless, the problem of covering with "left" vertices all cycles of length $|V'|$ in G' that contain a "left" vertex is equivalent to the original FVS problem. The "redundancy" in G' facilitates efficient implementation as described below.

Let A denote the adjacency matrix of graph G', and let $B = A^{|V'|}$. The entry $B(v', v')$ equals the number of cycles of length $|V'|$ that contain v'. Thus, $B(v', v')$ equals $p(v')$ after initialization in the *approximate_SC* algorithm. Intermediate values of $p(v')$ can be computed as follows. Suppose that vertex $v' \in V'$ is chosen at some iteration. Update the value of the entry $A(v', v'')$, which corresponds to edge $v' \to v''$, by dividing it by λ_1. Consequently, the attraction of every cycle that contains v is divided by λ_1^k, where k denotes the number of times v is contained in the cycle. Note that this deviation from the algorithm, namely, dividing the attraction of a cycle by λ_1^k (for $k \geq 1$) instead of λ_1, is immaterial, since it suffices to cover cycles of length $|V'|$ in G' in which each "left" vertex appears at most once. Hence, the updated entry $B(v', v')$ equals the new value of $p(v')$. Note that the value m denoting the total number of subsets in the set system can be replaced by any upper bound on it, e.g., $m \leq (2n)^{2n}$. This completes the description of an efficient implementation of the approximate Set Cover algorithm for the FVS problem.

The number of cycles considered by the above algorithm is at most $(2n)^{2n}$. Hence, the number of iterations of the algorithm is at most $O(\varepsilon^{-2} \cdot n^2 \log n)$. The complexity of each iteration is dominated by $O(M(n) \log n)$, where $M(n)$ denotes the complexity of matrix multiplication. Hence, the complexity of the algorithm is $O(\varepsilon^{-2} n^2 M(n) \log^2 n)$.

6 Remark: the NP-hardness of the problems

In [YKCP83] Yannakakis *et al.* show that the following DCUT problem is NP-Hard.
Input: A directed graph $H = (N, F)$, a directed graph $G = (V, E)$ with positive weights $w(e)$ associated with each edge $e \in E$, a 1-1 mapping $f : N \to V$, and an integer k.
Question: Is there a subset $S \subseteq E$ such that: $\sum_{e \in S} w(e) \leq k$, and if $(u, v) \in F$, then there is no directed path from $f(u)$ to $f(v)$ in $(V, E - S)$.

It is easy to see that the problem of finding minimum capacity multicuts in circular networks is a special case of the DCUT problem. In this special case the vertices of the "pattern graph" $H = (N, F)$ correspond to the terminals in the multicut problem, and the set of edges F consists of a directed edge from each source to its corresponding sink.

Since the multicut problem in circular networks is easily reducible to the symmetric multicut problem considered by [KPRT93], the symmetric multicut problem is also NP-Hard.

In the NP-Hardness proof of [YKCP83], the problem is first reduced to the case where the "pattern graph" H has no vertex with both incoming and outgoing edges. This is done by splitting each vertex u in H with both incoming and outgoing edges into two vertices u_{in} and u_{out} such that all the edges incoming to u are now incoming to u_{in}, and all the edges outgoing from u are now outgoing from u_{out}. Accordingly, the vertex v in G that corresponds to u is split into v_{in} and v_{out}, all edges originally incoming u are set to enter v_{in}, and all edges originally outgoing v are set leave v_{out}. Also, v_{out} is connected to v_{in} by an infinite capacity edge.

Then, it is proved that for such modified "pattern graphs", if H is not a complete bipartite graph, then the DCUT problem is NP-Hard even if H consists of only four vertices and all the weights are one. This implies that the multicut problem in circular networks is NP-Hard even if there are only two pairs of terminals and all the weights are one. Consequently, the SUBSET-FES problem is NP-Hard even if there are only two special vertices and all the weights are one.

We note that in order for the NP-Hardness proof in [YKCP83] to apply for the case where the original "pattern graph" H consists of vertices with both incoming and outgoing edges (and in particular to the multicut problem in circular networks and to the SUBSET-FES problem), we have to consider the special case in which the graph G contains infinite capacity edges between pairs of vertices. (These are the pairs of vertices that are constructed by splitting vertices in the original graph G.) Although the construction in [YKCP83] does not consider this special case, it is easy to see that by a minor change in their construction, this case is covered as well.

Acknowledgement

We would like to thank Noam Nisan for useful discussions.

References

[ABF90] M. Abramovici, M.A. Breuer and A.D. Friedman, "Digital Systems Testing and Testable Design," New York, Computer Science Press, 1990.

[ENS95] G. Even, J. Naor and B. Schieber, Unpublished manuscript, 1995.

[GVY93] N. Garg, V.V. Vazirani and M. Yannakakis, "Approximate max-flow min-(multi) cut theorems and their applications," *Proc. of the 25th ACM Symp. on Theory of Computing,* pp. 698–707, 1993.

[GGB89] R. Gupta, R. Gupta and M.A. Breuer, "BALLAST: A Methodology for Partial Scan Design," *Proc. 19th Int'l. Symp. on Fault-Tolerant Computing,* pp. 118–125, June, 1989.

[H63] T.C. Hu, "Multi-commodity network flows," *Operations Research,* 11:344–360, 1963.

[K72] R.M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations,* pp. 85–104, Plenum Press, N.Y., 1972.

[KPRT93] P.N. Klein, S.A. Plotkin, S. Rao and É. Tardos, New network decompositions theorems with applications, Unpublished manuscript, 1993.

[KARR90] P. Klein, A. Agrawal, R. Ravi, and S. Rao, "Approximation through multicommodity flow," *Proc. of the 31st IEEE Symp. on Foundations of Computer Science,* pp. 726–737, 1990.

[KST90] P. Klein, C. Stein, and É. Tardos, "Leighton-Rao might be practical: faster approximation algorithms for concurrent flow with uniform capacities," *Proc. of the 22nd ACM Symp. on Theory of Computing,* pp. 310–321, 1990.

[KW90] A. Kunzmann and H.J. Wunderlich, "An Analytical Approach to the Partial Scan Problem," *Journal of Elec. Testing: Theory and Applications,* 1:163–174, 1990.

[LN93] M. Luby and N. Nisan, "A parallel approximation algorithm for positive linear programming," *Proc. of the 25th ACM Symp. on Theory of Computing,* pp. 448–457, 1993.

[LR88] T. Leighton and S. Rao, "An approximate max-flow min-cut theorem for uniform multi-commodity flow problems with applications to approximation algorithms," *Proc. of the 29th IEEE Symp. on Foundations of Computer Science,* pp. 422–431, 1988. Directed graphs are dealt with in manuscript, Feb., 1992.

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

[LS91] C.E. Leiserson and J.B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica,* 6(1):5–35, 1991.

[PST91] S. Plotkin, É. Tardos and D. Shmoys, "Fast approximation algorithms for fractional packing and covering problems", *Proc. of the 32nd IEEE Symp. on Foundations of Computer Science,* pp. 495–504, 1991.

[S92] P. D. Seymour, "Packing Directed Circuits Fractionally", Manuscript, 1992. To appear in Combinatorica.

[YKCP83] M. Yannakakis, P.C. Kanellakis, S.S. Cosmadakis and C.H. Papadimitriou, "Cutting and partitioning a graph after a fixed pattern", *Proc. of the 10th Colloquium on Automata, Languages and Programming,* Lecture Notes in Computer Science No. 154, Springer-Verlag, pp. 712–722, 1983.