Approximating Minimum Feedback Sets and Multi-Cuts in Directed Graphs

(Extended Summary)

Guy Even[1] Joseph (Seffi) Naor[1] Baruch Schieber[2] Madhu Sudan[2]

[1] Computer Science Dept., Technion, Haifa 32000, Israel.
E-mail:{guy,naor}@cs.technion.ac.il.
[2] IBM Research Division, T.J. Watson Research Center, P.O. Box 218,
Yorktown Heights, NY 10598. E-mail:{sbar,madhu}@watson.ibm.com.

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 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. We present approximation algorithms for all four problems that achieve an approximation factor of $O(\min\{\log \tau^* \log\log \tau^*, \log n \log\log n)\}$, where τ^* denotes the value of the optimum fractional solution of the problem at hand. For the SUBSET-FVS and SUBSET-FES problems we also give an algorithm that achieves an approximation factor of $O(\log^2 |X|)$, where X is the subset of distinguished vertices and edges. This algorithm is based on an approximation algorithm for the multi-cut problem in a special type of directed networks. 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.

1 Introduction

This paper deals with approximating feedback sets in directed graphs, $G = (V, E)$. 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 which intersects every cycle in the graph. The unweighted versions of these problems are classical NP-hard problems, and appear in Karp's seminal paper [5]. 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 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 which connect gates [12]. Loosely speaking, finding a small feedback edge set in this graph helps reduce the hardware overhead required for testing the circuit using "scan registers" [3, 9]. 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 which intersects every interesting cycle. Similarly, a feedback vertex set with respect to the interesting cycles is a subset of vertices which intersects every interesting cycle. The interesting sets of cycles which we consider are characterized by a subset of vertices and edges. Namely, given a subset, X, (of vertices and of edges) the set of interesting cycles characterized by X is the set of all cycles which 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 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 first approximation algorithm for the FES problem was given by Leighton and Rao [11] (and followed by [8]). Their approximation factor is $O(\log^2 |V|)$ in the unweighted case. Leighton and Rao achieve this bound by using an $O(\log |V|)$ approximation algorithm for the bisection directed separator, which, in turn, is found by approximating special cuts which are called "quotient cuts".

It is easy to see that the FVS problem is a special case of a *covering* problem where the elements correspond to vertices and the subsets correspond to cycles. The (linear programming) dual of this problem is referred to as a *packing* problem, and in the case of the FVS problem, this means assigning a dual variable to each cycle in the graph, such that for each vertex, the sum of the variables corresponding to all the cycles passing through that vertex is at most the weight of the vertex. In a brilliant paper, Seymour [14] proved that the integral duality gap in the case of the unweighted FVS problem is at most $O(\log \tau^* \log\log \tau^*)$, where τ^* denotes the value of a maximum (fractional) packing. A careful examination of Seymour's proof reveals that all of his existential arguments can be made constructive, and thus, with certain other modifications, we obtain an algorithm for the unweighted FVS problem that achieves an approximation factor of $O(\log \tau^* \log\log \tau^*)$. (Notice that in the unweighted case $\tau^* < |V|$.)

Although Seymour explicitly restricts his discussion to the unweighted case, we are able to extend his proof to the weighted FVS problem. This results in an algorithm that achieves an approximation factor of $O(\min\{\log \tau^* \log\log \tau^*,$

$\log|V|\log\log|V|\}$), where, as before, τ^* denotes the cost of a maximum (fractional) packing, or equivalently the cost of a minimum (fractional) cover. Furthermore, we are able to extend this algorithm to the SUBSET-FVS and SUBSET-FES problems. This result is almost optimal within the primal-dual framework, since the duality gap is $\Omega(\log\tau^*)$ [14].

The first step in FVS and SUBSET-FVS 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 using the ellipsoid method. In fact, interior point algorithms can also achieve polynomial time since it is possible to decrease the number of constraints to be polynomial by reducing the FVS problem to multi-commodity flow [2]. (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 fractional solution, where the approximation bound is a constant. In Section 6 we present a combinatorial algorithm that finds a $(1+\varepsilon)$ approximation to the fractional FVS problem. This algorithm is both simple and is also more efficient than general linear programming methods. 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 [10]. We show how to adapt the approximate Set Cover algorithm to the FVS problem, in spite of the fact that the number of constraints in the corresponding Set Cover Problem is exponential. Plotkin, Shmoys and Tardos [13] 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 a approximate fractional solution significantly differs from the algorithm derived from [13]: They use a Lagrangian relaxation technique on the dual problem, whereas we deal with the primal problem.

The relation between feedback set problems and multi-cut problems (first observed by Leighton and Rao), motivated many researchers to focus their attention on multi-cut problems. This problem was first defined by Hu [4]. Here, we consider one of the versions of this problem defined as follows. Given a network with edge capacities, and a set of k source-sink pairs, find a minimum (capacity or cardinality) set of edges whose removal disconnects all the source-sink pairs. This problem is also NP-Hard: as demonstrated later, we can reduce the feedback problems to the multi-cut problem in *directed* networks. Recently, there has been much progress in approximating the multi-cut problem in the *undirected* case. Most notably, Garg, Vazirani and Yannakakis [2] achieved a $O(\log k)$ approximation factor for this problem. They introduced a novel "sphere growing" technique, refining and simplifying previous works of [11, 7]. Actually, the $O(\log k)$ factor achieved in [2] is with respect to the optimal *fractional* solution of the multi-cut 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 solution and the integer solution is indeed $\Omega(\log k)$. To the best knowledge, no approximation is known for the multi-cut problem in general directed graphs.

We present a simple reduction from the FES problem to a minimum capacity multi-cut in directed networks. The networks obtained by the reduction have a special structure, which we call *circular networks*. We are able to exploit this special structure to find a $O(\log^2 k)$ approximation factor of the minimum capacity multi-cut. Our algorithm is based on an adaptation of the "sphere growing" technique to directed circular networks and on a novel decomposition procedure.

Independently, Klein *et al.* [6] present a network decomposition algorithm for directed graphs. 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. It is not difficult to see that when their algorithm is applied to circular networks it yields a $O(\log^2 k)$ approximation factor of the minimum capacity multi-cut, as our algorithm. We can show how our approximation algorithms can be applied to obtain an $O(\min\{\log \tau^* \log\log \tau^*, \log|V| \log\log|V|\}, \log^2 k)$ 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 weighted FVS and weighted 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 (§3).
3. An extension of the above algorithms for the weighted SUBSET-FVS and weighted SUBSET-FES problems (§4).
4. A polynomial-time approximation algorithm for the multi-cut problem in networks defined by the reduction from feedback set problems 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 $O(\log^2 |X|)$, where X is the subset of vertices and edges that characterizes the interesting cycles. These algorithms are superior to the algorithms above if $|X| = o(2^{\sqrt{\log|V| \log\log|V|}})$ (§5).
5. A $(1+\varepsilon)$ factor approximation (combinatorial) algorithm for the fractional FVS problem (§6).

Due to space limitations many details and proofs are omitted.

2 The problems and reductions among them

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 which 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 which do not contain any vertex of B. In the BLACKOUT-FVS problem a minimum weight feedback vertex set which does not contain vertices of B is sought.

We consider the problem of finding minimum capacity multi-cuts in directed "circular" multi-commodity networks.
A *network* $N = \left(\widetilde{V}, \widetilde{E}, c(.), \{(s_i, t_i)\}_{i=1,\ldots,k}\right)$ is a quadruple, where: $(\widetilde{V}, \widetilde{E})$ is a directed graph, $c(.)$ is a capacity function defined on the edges, and $\{(s_i, t_i)\}_{i=1,\ldots,k}$ 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 *multi-cut* 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 which emanates from t_i and which enters s_i.

The following reductions can be shown between the above problems: FES $\preceq$ FVS, FVS $\preceq$ FES, BLACKOUT-FVS $\preceq$ FVS, SUBSET-FES $\preceq$ the multi-cut problem in circular networks, and the multi-cut problem in circular networks $\preceq$ SUBSET-FES. It is also easy to see that the subset, $X \subset V \cup E$, defining the interesting cycles, can be reduced to a subset of vertices.

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. Therefore, these problems are equally hard to approximate in polynomial time.

3 Approximating the weighted FVS problem

In this section we describe an algorithm for approximating the minimum weight FVS of a weighted directed graph $G = (V, E)$. This algorithm requires a fractional feedback vertex set (defined below). Such a fractional solution can be either found by using general linear programming techniques or by the approximation algorithm described in Section 6. The algorithm presented in this section is an algorithmic adaptation of Seymour's paper [14] which we extend to the weighted case.

A fractional feedback vertex set of G is a function $t : V \rightarrow [0, 1]$, such that every directed cycle, $\mathcal{C}$, is "covered" by t; that is, $\sum_{v \in \mathcal{C}} t(v) \geq 1$. The weight of a fractional feedback vertex set, t, is defined by $\sum_{v \in V} w(v) \cdot t(v)$.

Let $\widehat{\varphi}(v)$ denote the value of vertex v in the fractional solution found by the approximation algorithm described in Section 6. Let $\widehat{\Phi} = \sum_{v \in V} w(v) \cdot \widehat{\varphi}(v)$. Let τ^* denote the weight of the optimal fractional feedback vertex set. By the results of Section 6, $\widehat{\Phi} = O(\tau^*)$.

3.1 Modifying the graph

We first show how to transform $\widehat{\varphi}$ so that for each vertex v, if $\widehat{\varphi}(v) > 0$, then $\widehat{\varphi}(v) \geq 1/2n$. We define the following fractional FVS φ':

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

The following proposition is immediate.

Proposition 1. *The function φ' is a feasible* FVS *and its weight is at most $2\widehat{\Phi}$.*

We now round up each value of φ' to the nearest integral multiple of $1/2n$. This clearly preserves feasibility, and at most doubles the cost. Let us denote the new fractional FVS by φ. The cost of φ, denoted by Φ, is at most $4\widehat{\Phi}$.

We modify the graph as follows. All vertices $v \in V$ for which $\varphi(v) = 0$ are marked as blackout vertices, and we delete them from the graph by using the reduction BLACKOUT-FVS $\preceq$ FVS mentioned in Section 2.

We now "equalize" the graph G into graph H as follows. For each vertex $v \in V(G)$, we replace it by a directed chain of vertices $v_1 \rightarrow \ldots \rightarrow v_k$ where $k = \varphi(v) \cdot 2n$. By the above discussion, k is a positive integer. Set the weight of each vertex v_i, $1 \leq i \leq k$, to $w(v)$. For each edge $u \rightarrow v \in E(G)$, we add a directed edge from the last vertex of u's chain to the first vertex of v's chain. The next lemma summarizes the properties of the graph H.

Lemma 2. *[14] Graph H has the following properties:*
(i) There is a 1-1 correspondence between the 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, denoted by girth(H), satisfies:

$$girth(H) \geq \frac{w(V(H))}{\Phi}, \text{ where } w(V(H)) \text{ is the total weight of the vertices in } H.$$

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

3.2 The algorithm

We present the algorithm for approximating the FVS in graph H. Seymour [14] defines the function μ as follows:

$$\mu(x) = \begin{cases} 0 & \text{if } x < 1 \\ 4x \cdot \ln(4x) \cdot \ln\log_2(4x) & \text{otherwise.} \end{cases}$$

The algorithm for approximating the optimal weighted FVS proceeds recursively as follows: It first finds a directed vertex separator S of the graph H. A directed vertex separator of H is a partition of the vertices of the graph 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* $(H(V,E), w, g)$ depicted in Figure 1. We add the vertices of S to the FVS, and recursively compute an FVS of the subgraphs induced by A and B.

Theorem 3. *Suppose that the girth of graph H is not smaller than g. The algorithm finds an* FVS *of H whose weight is at most $\mu\left(\frac{w(V(H))}{g}\right)$.*

[hbt]

separator $(H(V,E), w, g)$
  Choose a vertex $v_0 \in V(H)$.
  Construct BFS "layers" starting from v_0 as follows:
    For $i = 1, \ldots, (g-1)$ define
      $X_i \triangleq \{u : dist(v_0, u) = i\}$.
    $X_g \triangleq V(H) - \cup_{j<g} X_j$.
  $i = 0$
  repeat
    $i = i + 1$
    $A_i \triangleq \cup_{j<i} X_j$
    $B_i \triangleq \cup_{j>i} X_j$
  until $w(X_i) \leq \mu(w(V(H))/g) - \mu(w(A_i)/g) - \mu(w(B_i)/g)$
  return(A_i, X_i, B_i)

Fig. 1. The directed separator algorithm.

Combining Theorem 3 with Lemma 2 yields the following corollary.

Corollary 4. *Let H denote the directed graph obtained by the modifications described in subsection 3.1 of the graph G given as input. Let $U \subseteq V(H)$ denote the* FVS *of H found by the algorithm. Let $\bar{U} \subseteq V(G)$ denote the* FVS *of G that corresponds to U. Then,*

$$w(\bar{U}) \leq \mu(\Phi) \leq 16\widehat{\Phi} \cdot \ln(16\widehat{\Phi}) \cdot \ln\log_2(16\widehat{\Phi}).$$

Since $\widehat{\Phi} = O(\tau^*)$, where τ^* denotes the cost of an optimal fractional feedback vertex set, $w(\bar{U}) \leq \mu(O(\tau^*))$.

3.3 Strongly polynomial approximation bounds and running times

The approximation bounds obtained in Corollary 4 may be very weak in cases where the weight of an optimal fractional feedback vertex set, τ^*, is super-polynomial in $n = |V(G)|$. To overcome this problem we "truncate" the vertex weights to obtain an FVS of weight $O(\tau^* \cdot \ln n \cdot \log_2 \log_2 n)$.

We truncate the vertex weights as follows. Let Φ denote the weight of a fractional feedback vertex set for which $\Phi = O(\tau^*)$. Let IN $\triangleq \{v \in V : w(v) \leq \Phi/n\}$ and let OUT $\triangleq \{v \in V : w(v) > \Phi \cdot n\}$. We add all vertices of IN to the feedback vertex set, which increases the weight of the approximate FVS by at most Φ. All 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 FVS. The latter follows by observing that the ratio between the values of an optimal integral FVS and an optimal fractional FVS is at most n. (Given a fractional FVS, if one rounds up to 1 all values which are bigger than or equal to $1/n$, and rounds down to 0 all other values, then a feasible integral 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 FVS is a polynomial in n, and the modified (equalized) graph, H, is also of polynomial size. Therefore, the algorithm finds a FVS of weight $O(\tau^* \ln n \log\log n)$.

4 Approximating weighted SUBSET-FVS

In this section we discuss approximating solutions of SUBSET-FVS. The reductions from SUBSET-FES to SUBSET-FVS mentioned in Section 2 imply that the results of this section hold for the SUBSET-FES problem as well. We consider instances when one is interested in disconnecting cycles which intersect a given set of vertices, $X \subset V$. The algorithm and its analysis is a modification of the algorithm presented in Subsections 3.1 and 3.2. Therefore, we only discuss the required modifications.
The $(1+\varepsilon)$ approximation algorithm is adapted to the SUBSET-FVS problem.
A modified definition of girth is required: Define the girth of the graph to be the minimum number of non-special vertices in an interesting cycle. Note that Lemma 2 holds with respect to the new definition of girth.
Before searching for a separator, the graph is partitioned into strongly connected components. If a strongly connected component lacks special vertices then the empty set is a valid feedback vertex set with respect to X. A separator for components which contain special vertices is found separately for each strongly connected component. In Procedure *separator* the layers are constructed starting from a special vertex. This ensures that the graph contains g layers, where g is a lower bound on the modified girth.

5 An $O(\log^2 |X|)$-approximation for SUBSET-FES

In this section we show how to find a feedback edge set of a weighted directed graph $G(V,E)$ with respect to a set of interesting cycles which is characterized by a subset of vertices $X \subseteq V$, where $|X| = k$. The SUBSET-FES problem is considered for ease of presentation, but the reduction from the SUBSET-FVS problem to the SUBSET-FES problem mentioned in Section 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 $O(\tau^* \log^2 k)$, 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 use the reduction to the multi-cut problem mentioned in Section 2. Let $N = \left(\widetilde{V}, \widetilde{E}, c(.), \{(s_i, t_i)\}_{i=1,\ldots,k}\right)$ be the directed network given by reducing the SUBSET-FES instance to a multi-cut problem instance. Let C^* denote the cost of an optimal fractional multi-cut in N. We show how to find a multi-cut whose cost is $O(C^* \cdot \log^2 k)$. Note that a fractional multi-cut may be found either by solving a general linear program or by using the approximation algorithm described in Section 6.

5.1 Overview

Our algorithm uses the sphere growing technique of Garg, Vazirani and Yannakakis [2]. They used it to approximate the multi-cut 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 in [2]. The algorithm consists of two stages. In the first stage the fractional multi-cut problem is found (exactly) using a polynomial-time linear programming algorithm. (Note that our $(1+\varepsilon)$-factor approximation algorithm described in Section 6 can also be used.) We regard the output values, $d(e)$, as a distance function, where $d(e)$ describes the distance between the tail of e and the head of 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 all 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 [2] show that under an appropriate choice of parameters the radius of the sphere is bounded by 1/2. We add the cut of this sphere to the multi-cut. 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" all the vertices in the sphere and the 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 the 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 the 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 cause us to "charge" the same edges more than once.

We overcome these problems as follows. For the first problem, we note that in the *special* circular network defined by the SUBSET-FES problem 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 have the situation that 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 one 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.

5.2 The sphere growing procedure

In this subsection, we present an adaptation of the region growing procedure of Garg, Vazirani and Yannakakis [2]. The procedure in [2] deals with undirected networks, whereas our procedure deals with directed circular networks.

Let $\{d(e)\}_{e \in \widetilde{E}}$ be an optimal (within a constant factor) fractional multi-cut of the network N, and let C denote its cost. We show how to grow a sphere from a source s_{i_0}.

Regard the values, $d(e)$, as a distance function, where $d(e)$ describes the distance between the tail of e and the head of e. Define $dist(u, v)$ as 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:

$$\ell_i \triangleq dist(v_0, v_i) \qquad A_i \triangleq \{v_0, v_1, ..., v_i\}$$

For a set A_i, $i \geq 0$, define the weight, $wt(A_i)$, as follows. (Do not confuse this weight with the edge weights, $w(e)$, in the SUBSET-FES problem.)

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

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 time, σ

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

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

Lemma 5. *The radius of the sphere A_σ, denoted ℓ_σ, is bounded by* $\ln(k+1)/\varepsilon$.

5.3 Computing the multi-cut

We show how to use the sphere growing procedure to compute the multi-cut. We set the sphere growing parameter to $\varepsilon = (2 + \delta) \cdot \ln(k + 1)$, for some $\delta > 0$. 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 multi-cut. 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 sub-problem given by the network induced by the vertices inside the sphere. This sub-problem 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 sub-problems

each of which consists of at most $(k-1)/2$ pairs and whose total size is bounded by the size of the original problem.

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 multi-cut.

We analyze the cost of the multi-cut, $\cup_j \gamma(R_j)$. Fix a network, N, with n nodes and k source-sink pairs. Let $cut(n,k)$ denote the maximum cost of the multi-cut 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 sub-problems with no more than $(k-1)/2$ pairs). Let $c(\gamma(R_j))$ denote the cost of $\gamma(R_j)$, n_j denote the number of vertices in R_j, and k_j denote the number of source-sink pairs in R_j. Then, $cut(n,k)$ satisfies the following recurrence inequality:

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

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

Lemma 6. *If $\varepsilon = (2+\delta)\ln(k+1)$, then $cut(n,k) \leq 2(2+\delta)\ln(2) \cdot C \cdot \log^2(k+1)$.*

We summarize the approximation factor obtainable by our procedure.

Theorem 7. *If $\varepsilon = 2.005\ln(k+1)$, then $cut(n,k) \leq 2.78C \cdot \log^2(k+1)$.*

6 Approximating the fractional optimal solution

In this section we show that an approximation by a $(1+\varepsilon)$ factor to the fractional FVS problem can be computed efficiently by a combinatorial algorithm. In Section 6.1 we present a $(1+\varepsilon)$-approximation algorithm for an arbitrary Set Cover problem, that is derived from a parallel algorithm for approximating positive linear programming by Luby and Nisan [10]. An important feature of this algorithm is that the complexity weakly depends (logarithmically) on the the number of subsets in the set system. In Section 6.2 we show that this algorithm can be implemented in the case of the FVS problem in spite of the fact that the number of constraints in the Set Cover problem is exponential.

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

The Set Cover Problem is defined as follows. Let $(V, \mathcal{S})$ be a set system, where V denotes the elements and $\mathcal{S}$ denotes the collection of subsets. There is a non-negative cost function $c(v)$ associated with each element $v \in V$. 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 the optimal fractional cover.

The algorithm we present associates an attraction function $p(.)$ with each subset. 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 $\sum_{S:v\in S} p(S)$. Let $n = |V|$, and $m = |\mathcal{S}|$.

The algorithm *approximate_SC* depicted in Figure 2 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.

[hbt]

```
approximate_SC (V, S, c(.), ε)
    Define: λ ≜ (1 + (1/3)ε) and a ≜ 6/ε.
    ∀v ∈ V : ℓ(v) = 0.
    ∀S ∈ S : p(S) = m^a.
    repeat
        Choose an element v ∈ V that minimizes the ratio c(v)/p(v).
        ℓ(v) ← ℓ(v) + 1
        ∀S such that v ∈ S:
            if p(S) > 1 then p(S) ← p(S)/λ.
            else p(S) ← 0.
    until p(S) ≤ 1
    return({ℓ(v) : v ∈ V})
```

Fig. 2. The integral multi cover approximation algorithm.

For a vertex $v \in V$, The fractional cover $\varphi(v)$ is obtained from the multi cover as follows

$$\varphi(v) \triangleq \frac{\ell(v)}{a \cdot \log_\lambda m}, \text{ where } \lambda \triangleq (1 + \frac{1}{3}\varepsilon) \text{ and } a \triangleq \frac{6}{\varepsilon}.$$

Theorem 8. *The following claims hold: (i) The function φ is a feasible cover of $\mathcal{S}$. (ii) $\sum_{v\in V} \varphi(v) \cdot c(v) \leq (1+\varepsilon) \cdot \tau^*$. (iii) The number of iterations of the algorithm is $O\left(n \ln m/\varepsilon^2\right)$.*

6.2 Implementing the fractional FVS algorithm

The FVS problem reduces naturally to a Set Cover problem by defining the vertices of the graph as elements and the simple directed cycles as the subsets of set system. However, a naive implementation of algorithm *approximate_SC* to solve the problem would take exponential time, because the size of the set system is 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. Below, we present a more involved implementation that runs in polynomial time. The detailed implementation is given in Figure 3.

First, we modify the input graph $G = (V, E)$ by splitting each vertex $v \in V$ into two new vertices, v' and 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 $G' = (V', E')$. The natural

correspondence between the vertices of G and G' induces a surjection of the directed cycles (both simple and non-simple) of G' which pass through v' onto the cycles of G which pass through v. Let A denote the adjacency matrix of graph G' and let I denote the identity matrix.

Next, we redefine the Set Cover problem that corresponds to the FVS problem. As before, the elements correspond to the vertices in the graph. The subsets now correspond to the complete set of directed cycles in the graph, simple and non-simple, whose length is at most $|V'|$. To facilitate efficient implementation, some of the cycles appear with multiplicity in the set system. Clearly, this does not increase the value of the optimal solution τ^*. Specifically, the multiplicity of each cycle is set in such a way that for each vertex $v \in V'$, the number of cycles of length at most $|V'|$ that contain it, is given by the entry $B(v,v)$ of the matrix $B = (A+I)^{|V'|}$. (It is well known that this entry counts the number of cycles of length at most $|V'|$ that contain v, but with some multiplicity greater than one.) The size of the set system, m, is upper bounded by the sum of all the diagonal entries of B.

Finally, we scale down the parameters used by the algorithm. Instead of initializing $p(C)$, for each cycle C, to m^a, we initialize it to 1. Similarly, we scale down the thresholds for stopping the loop and nullifying $p(C)$ to m^{-a}. Notice that after the scaling, the initial value of $p(v')$ is the number of cycles of length at most $|V'|$ that contain vertex v', and thus is given by $B(v',v')$. It is not difficult to see that 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 $A(v',v'')$ (that corresponds to edge $v' \rightarrow v''$) by dividing it by λ (or nullifying it, if $A(v',v'') \leq m^{-a}$). Then, the updated entry $B(v',v')$ is the new value of $p(v')$.

[hbt]

*fractional_*FVS $(V', A, c(.), \varepsilon)$
  Define: $\lambda \triangleq (1 + \frac{1}{3}\varepsilon)$ and $a \triangleq 6/\varepsilon$.
  $\forall v' \in V' : \ell(v') = 0$.
  for $i = 1, \ldots, (18n(1+1/\varepsilon)\ln m)/\varepsilon^2$
    Compute $B = (A+I)^{|V'|}$.
    Choose an element $v' \in V'$ that minimizes the ratio $\frac{c(v')}{B(v',v')}$.
    $\ell(v') \leftarrow \ell(v') + 1$
    $\forall (v', v'') \in V' \times V''$
      if $A(v',v'') > m^{-a}$ then $A(v',v'') \leftarrow A(v',v'')/\lambda$.
      else $A(v',v'') \leftarrow 0$.
  return$(\{\ell(v') : v' \in V'\})$

Fig. 3. The fractional FVS algorithm.

Lemma 9. *Let $\ell_i(v)$ denote the multiplicity of vertex v' in the multi-cover up to*

iteration i. For every cycle C of G', the attraction satisfies

$$p(C) \leq \left(\frac{1}{\lambda}\right)^{\sum_{v \in C} \ell_i(v)}$$

Note, that equality holds for simple cycles. For non-simple cycles, the attraction may decrease faster since some vertices appear several times in such cycles.

We get that for simple cycles the attraction is updated as required by Algorithm *fractional*_FVS. For a non-simple cycle, the attraction may decrease faster than required. However, a faster decrease in the attraction (of a cycle) has the following effect. On one hand, it may decrease the total number of iterations – but this can only be of help to the algorithm. On the other hand, the coverage of such a cycle may not be sufficient – but since this applies only to non-simple cycles, their level of coverage is not of interest to us.

There exists a constant α such that the number of cycles considered by the above algorithm is at most $2^{\alpha n \log n}$. Hence, the number of iterations of the algorithm is at most $O(n^2 \log n)$. 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 $O(n^2 M(n) \log^2 n)$.

Remark: The algorithm presented in this section can be modified to handle the covering problem associated with a SUBSET-FVS problem.

Acknowledgement

We would like to thank Noam Nisan for useful discussions. The research of the first author was supported in part by the Miriam and Aaron Gutwirth Memorial Fellowship. The research of the second author was supported in part by Grant No. 92-00225 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel.

References

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

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

[3] 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.

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

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

[6] P.N. Klein, S.A. Plotkin, S. Rap and É. Tardos, "New network decompositions theorems with applications," unpublished manuscript, 1993.

[7] P. Klein, A. Agrawal, R. Ravi, and S. Rao, "Approximation through multi-commodity flow," *31st FOCS,* pp. 726-737, 1990.

[8] P. Klein, C. Stein, and É. Tardos, "Leighton-Rao might be practical: faster approximation algorithms for concurrent flow with uniform capacities," *22nd STOC,* pp. 310-321, 1990.

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

[10] M. Luby and N. Nisan, "A parallel approximation algorithm for positive linear programming," *25th STOC,* pp. 448-457, 1993.

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

[12] C.E. Leiserson and J.B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica,* Vol. 6, No. 1, pp. 5-35. 1991.

[13] S. Plotkin, É. Tardos and D. Shmoys, "Fast approximation algorithms for fractional packing and covering problems", *32nd FOCS,* pp. 495-504, 1991.

[14] P.D. Seymour, "Packing Directed Circuits Fractionally," Manuscript, (1992). To appear in *Combinatorica.*

This article was processed using the LaTeX macro package with LLNCS style