Approximation Algorithms for Steiner and Directed Multicuts

Philip N. Klein[*]
Brown University

Serge A. Plotkin[†]
Stanford University

Satish Rao
NEC Research

Éva Tardos[‡]
Cornell University

Abstract

In this paper we consider the *steiner multicut* problem. This is a generalization of the minimum multicut problem where instead of separating node *pairs*, the goal is to find a minimum weight set of edges that separates all given *sets* of nodes. A set is considered separated if it is not contained in a single connected component.

We show an $O(\log^3(kt))$ approximation algorithm for the steiner multicut problem, where k is the number of sets and t is the maximum cardinality of a set. This improves the $O(t \log k)$ bound that easily follows from the previously known multicut results.

We also consider an extension of multicuts to directed case, namely the problem of finding a minimum-weight set of edges whose removal ensures that none of the strongly connected components includes one of the prespecified k node pairs. In this paper we describe an $O(\log^2 k)$ approximation algorithm for this directed multicut problem. If $k \ll n$, this represents and an improvement over the $O(\log n \log\log n)$ approximation algorithm that is implied by the technique of Seymour.

[*]Research supported by NSF PYI award CCR-9157620, together with PYI matching funds from Honeywell Corporation, Thinking Machines Corporation, and Xerox Corporation. Additional support provided by ARPA contract N00014-91-J-4052 ARPA Order No. 8225.

[†]Research supported by U.S. Army Research Office Grant DAAL-03-91-G-0102 and by a grant from Mitsubishi Electric Laboratories.

[‡]Research supported in part by a Packard Fellowship, an NSF PYI award, by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research, through NSF grant DMS-8920550, and by NEC.

1 Introduction

Steiner Multicuts

Given an edge-weighted undirected graph, the minimum multicut problem is to find the minimum weight set of edges whose removal separates all given node pairs. The multicut problem is well studied [10, 9], and can be approximated to within a factor of $O(\log k)$, where k is the number of pairs [5].

In this paper we consider the *steiner multicut* problem. This is a natural generalization of the multicut problem, where instead of node *pairs*, we have *sets* of nodes. We say that a set is "separated" if it is not contained in a single connected component. The problem is to find a minimum weight set of edges whose removal separates all of the given sets.

Using techniques used to produce the $O(\log k)$ approximation to the multicut problem [10, 9, 5], it is relatively easy to get an $O(t \log k)$ approximation to the steiner multicut problem, where t is the cardinality of the largest set. In this paper we develop an $O(\log^3(kt))$ approximation algorithm.

The previous results about multicuts can be viewed as a consequence of a simple and very useful *network decomposition* lemma. This lemma states that that given a graph with n nodes and m edges, and a positive δ, one can remove at most $O(\frac{m}{\delta} \log n)$ edges such that the distance between any two nodes that stay in the same connected component is bounded by δ.[1] Other applications of network decomposition techniques of this type include routing with small size tables [11, 1], symmetry-breaking [2], and synchronization of asynchronous networks [3].

In contrast to the previous approaches, our multicut results do not follow from a generalized decomposition lemma. We derive a generalization of the above lemma as a corollary of our steiner multicut theorem. We show that by removing at most $O(\frac{m}{\delta} \log^3(kt))$ edges, one can separate all the sets whose minimum steiner trees have at least δ edges, where k is the number of sets considered, t is the cardinality of the largest set, and m is the total number of edges. We also prove a weighted version of this lemma.

Directed multicuts

Feedback arc set is the problem of finding a minimum set of edges whose removal makes the graph acyclic. Leighton and Rao showed an $O(\log^2 n)$ approximation algorithm [10], where n is the number of nodes. This was improved to $O(\log t \log \log t)$ by Seymour [14], where t is the size of the minimum size feedback arc set. Seymour's result trivially extends to the weighted case and leads to an $O(\log n \log \log n)$ approximation algorithm (see [4]).

A natural generalization of the feedback arc set problem is to consider a weighted directed graph and a set of node pairs. The *directed multicut* problem is to find a minimum-weight multicut that separates all the given pairs. In other words, we have to find a set of edges whose removal ensures that none of the prespecified node pairs are contained in a strongly connected component.

An example of a problem that reduces to finding minimum weight directed multicuts is the *2-CNF clause-deletion* problem, i.e. the problem of finding a minimum weight set of clauses in a 2-CNF formula whose deletion makes the formula satisfiable. Previously known undirected multicut

[1] Note that m/δ is an easy lower bound.

results have been used in approximation algorithm for the special case of 2-CNF$\equiv$ clause-deletion problem, as described in [9, 5].

As observed in [4], Seymour's result implies an $O(\log n \log\log n)$ approximation algorithm for the directed multicut problem. In this paper we show an $O(\log^2 k)$ approximation to the minimum weight directed multicut problem, where k is the number of given pairs. This is an improvement for the case where $k \ll n$. The same bound for a slightly more specialized case was later discovered by Even et. al. [4].

At the heart of our minimum-weight directed multicut algorithm lies the following *directed decomposition* lemma: By removing at most $O(\frac{m}{\delta}\log^2 n)$ edges, any directed graph can be decomposed into a set of *strongly connected components* where any two nodes in the same connected component lie on a directed cycle whose length is at most δ.

Our techniques are based on a fundamental relationship between multicuts and symmetric multicommodity flows in directed graphs. (A flow is symmetric if each unit of flow sent from a source s to the corresponding sink t also has to return from t to s, though not necessarily along the same path.) Multicuts can be viewed as integer solution to the linear programming dual of a multicommodity flow problem that naturally corresponds to the multicut problem.

As a byproduct we show that if the capacity of every multicut is at least $O(\log^3 k)$ times the demand separated by the multicut, then a feasible multicommodity flow exists. We also give an $O(\log^3 k)$-approximation algorithm for the *minimum-ratio directed multicut* problem, *i.e.*, the problem of finding a directed multicut minimizing the ratio of the capacity of the multicut to the sum of the demands that are separated by the multicut. The only previously known non-trivial results of this type for directed graphs are due to Leighton and Rao [10], who considered the special case where there is a *unit demand* between *every pair* of nodes.

Running Times

The computational bottleneck of the methods described in this paper is solving appropriately constructed linear programs. The LP needed to be solved for the directed cuts approximation can be solved by any linear programming algorithm. The corresponding LP in the steiner cuts case has an exponential number of constraints and can be solved by using convex programming (see *e.g.* [15]).

Although this leads to polynomial time algorithms, the resulting running time is pretty slow. Much faster algorithms can be obtained by using the techniques of [13], where the authors develop a general framework for constructing fast approximation algorithms for certain classes of linear programs. All the linear programs that arise in the context of the methods described in this paper fall into this general framework, and thus can be solved efficiently.

2 Steiner Multicuts

In this section we consider multicuts in undirected graphs where instead of separating *pairs* of terminals we need to separate *sets* of terminals. The main result in this section is an $O(\log^3(kt))$-approximation algorithm for finding a minimum-capacity multicut of this sort, where k is the number of sets and t is maximum cardinality of a set. The traditional multicut approximation algorithms that deal with separating pairs of nodes were derived using a result about graph decomposition. In contrast to this, the proof of the corresponding decomposition result for the Steiner case (presented at in the end of this section) follows as a corollary of our results about minimum Steiner multicuts.

2.1 Definitions

Let S_i denote the sets of terminals for commodity i where $1 \leq i \leq k$. A *multicut* $\mathcal{A}$ is a partition of V into sets $U_0, \ldots, U_p$; we say that a multicut *separates* terminal set S_i if there is no set U_j containing all of S_i. The capacity of the multicut $u(\mathcal{A})$ is the sum of the capacities of all edges crossing from one set in the partition to another, i.e. $u(U_1, \ldots, U_p) = \frac{1}{2}\sum_i u(U_i)$. The goal is to find a minimum-capacity multicut that separates all k sets of terminals.

In the related *minimum-ratio Steiner cut* problem there is a demand d_i associated with every set of terminals. The *demand* $d(U)$ across a cut U is the sum of the demands associated with the separated sets. The problem is to find a cut that minimizes the ratio of the capacity of the cut and the demand across the cut. In this minimum cut problem, demands express the desirability of separating the corresponding sets of terminals. For the minimum ratio problem we use cuts instead of multicuts. While multicuts would also give a natural way of separating sets of terminals, it turns out that the minimum ratio of a multicut is no less than the minimum of the cut ratios associated with the sets defining the multicut.

The minimum-ratio Steiner cut problem is closely related to the following generalization of the multicommodity flow problems that we will refer to as the *concurrent Steiner flow* problem. In contrast to concurrent flow, where each commodity is associated with a single pair of terminals, here commodity i is associated with a set of terminals S_i, for $i = 1, \ldots, k$. Each commodity i also has an associated demand d_i. A *Steiner flow* f_i of commodity i in G connecting terminal set S_i is defined as a collection of Steiner trees spanning the terminal sets S_i; each tree has an associated real value. Let $\mathcal{T}_i$ denote a collection of Steiner trees that span terminals S_i, and let $f_i(\tau)$ be a nonnegative value for every tree $\tau \in \mathcal{T}_i$. The *value* of the Steiner flow thus defined is $\sum_{\tau \in \mathcal{T}_i} f_i(\tau)$. The amount of flow of commodity i through an edge vw is given by $f_i(vw) = \sum\{f_i(\tau) : \tau \in \mathcal{T}_i \text{ and } vw \in \tau\}$. A flow is *feasible* if it satisfies the capacity constraints, $\sum_i f_i(vw) \leq u(vw)$ for every edge $vw \in E$. The goal is to maximize a common percentage z such that there exists a feasible Steiner flow for which the flow of commodity i is zd_i.

One can view this concurrent Steiner flow problem as a fractional packing of Steiner trees into a capacitated graph, where for each i we pack at least zd_i trees associated with terminal set S_i. Notice that the integer version of this tree packing problem, namely integer packing of Steiner trees with given sets of terminals, arises in VLSI design, in particular in the problem of routing multiterminal nets. Another natural application of this problem arises in the context of optimal routing of multicast circuits in a communication network.

2.2 Finding approximately minimum-ratio Steiner cuts

The minimum-ratio Steiner cut provides a simple combinatorial bound on the maximum possible value of z. We prove that this bound is the best possible up to a factor of $O(\log^2(kt))$. We also give a polynomial time $O(\log^2(kt))$-approximation algorithm for finding the minimum-ratio Steiner cut.

Using this algorithm as a subroutine, we give an $O(\log^3(kt))$-approximation algorithm for finding the minimum-capacity Steiner multicut separating all sets of terminals.

The maximum concurrent Steiner flow problem can be formulated as a linear program, albeit one with an exponential number of variables, one for every possible Steiner tree. In order to find an optimal solution using the a convex programming algorithm *e.g.*, [15], or the more efficient approximation algorithm of [13], we would have to solve the minimum-cost Steiner tree problem, which is NP-hard. Instead of working with this linear program, we work with a closely related one, in which instead of Steiner trees we pack *restricted* Steiner trees. A restricted Steiner tree in G for a terminal set S is a connected multigraph H spanning S constructed in the following way: Let $G_S = (S, E_S)$ be a complete auxiliary graph on nodes in S and let T be a spanning tree in G_S. Multigraph H is constructed by replacing each edge vw of T with edges of some path from v to w in G. Observe that using restricted Steiner trees essentially corresponds to using the standard 2-approximation algorithm for the minimum-cost Steiner tree problem. Also, note that min-cost restricted Steiner tree can be found by assigning each vw edge in G_S weight equal to the distance from v to w in G and computing a minimum-weight spanning tree of G_S.

We will bound the capacity of the multicut we obtain in terms of the value of the optimum primal solution to the restricted problem. Clearly the value of the restricted problem $\tilde{z}^*$ is no more than the optimum value z^* for the unrestricted problem. It is not hard to show that $z^* \leq 2\tilde{z}^*$, but we will not need this fact.

Consider the linear-programming formulation of the concurrent restricted Steiner flow problem. The linear programming dual of the concurrent restricted Steiner flow problem is as follows. There is a nonnegative cost variable $\ell(e)$ for every edge e. Let $\tilde{\mathrm{w}}_\ell(S_i)$ denote the minimum cost of a restricted Steiner tree with terminal set S_i. The goal is to minimize $\sum_e \ell(e)u(e)$ subject to the constraint $\sum_i d_i \tilde{\mathrm{w}}_\ell(S_i) \geq 1$. Using the fact that min-weight restricted Steiner tree can be found in polynomial time, we can solve the separation problem for this dual linear program, and hence can find (approximately) optimum solutions to both the dual and the primal problem [15, 13].

We will use W to denote $\sum_e \ell(e)u(e)$, and $W(F) = \sum_{e\in F} \ell(e)u(e)$ for a subset of the edges $F \subseteq E$. For a subset of nodes S we use $e(S)$ to denote the set of edges having at least one endpoint in S. For Steiner flow and cut problems with sets of terminals S_i for $i = 1, \ldots, k$, let t_i denote $|S_i|$. Recall that $t = \max_i t_i$. The number of terminals $| \cup_i S_i|$ is denoted by T.

Theorem 2.1 Given an instance of the minimum ratio Steiner cut problem, let $\mathbb{C}$ denote the set of cuts that separate some the demand sets. Then one can find a cut U in polynomial time such that

$$\tilde{z}^* \leq z^* \leq \min_{U'\in\mathbb{C}} \frac{u(U')}{d(U')} \leq \frac{u(U)}{d(U)} \leq O(\log T \log(kt))\tilde{z}^*.$$

We will use the following lemma due to Garg, Vazirani and Yannakakis [5]:

Lemma 2.2 For any positive ϵ and p, and any node s, there exists a set U containing s such that $u(\Gamma(U)) \leq \epsilon\,(W/p + W(e(U)))$, and every node in U is at a distance less than $\epsilon^{-1}\ln(p+1)$ from s.

Proof: We include the proof of this lemma for completeness. Consider the set of nodes $V(x) = \{v \in V : \mathrm{dist}_\ell(s, v) \leq x\}$ for a parameter x. The function $W(x)$ defined below is in essence the volume reachable in distance x with an additional W/p extra volume at node s.

$$W(x) = W/p + \sum_{v,w \in V(x)} u(vw)\ell(vw) + \sum_{v \in V(x), w \notin V(x)} u(vw)(x - \mathrm{dist}_\ell(s,v)).$$

Note that W is a monotone function of x, it is differentiable almost everywhere, and satisfies $W'(x) \geq \sum_{v \in V(x), w \notin V(x)} u(vw)$. If the lemma is false, then $W'(x) > \epsilon W(x)$ for every $x \leq \epsilon^{-1} \ln(p+1)$. This implies that for $x = \epsilon^{-1} \ln(p+1)$

$$\ln W(x) > \epsilon x + \ln W(0) = \ln \left(\frac{p+1}{p} W \right).$$

In addition, we have that $W(0) = W/p$, and $W(x) \leq W/p + W = \frac{p+1}{p} W$ for every x. The resulting contradiction proves the lemma. ∎

First we prove a weaker version of Theorem 2.1 with the $\log(kt)$ in the bounds replaced by $\log D$ where $D = \sum_i (t_i - 1) d_i$, and we assume that the demands are integral. Then we show how to improve the weaker bound involving $\log D$ to the bound claimed in the theorem by using the technique of Plotkin and Tardos [12].

Theorem 2.3 Given an instance of the minimum ratio Steiner cut problem, let $\mathbb{C}$ denote the set of cuts that separate some the demand sets. Then one can find a cut U in polynomial time such that

$$\tilde{z}^* \leq z^* \leq \min_{U' \in \mathbb{C}} \frac{u(U')}{d(U')} \leq \frac{u(U)}{d(U)} \leq O(\log T \log D) \tilde{z}^*.$$

Proof: The only non-trivial inequality is the last one. We prove this inequality by induction on D. Let $\sigma^* = \min_{U' \in \mathbb{C}} \frac{u(U')}{d(U')}$ denote the minimum ratio of a cut. Let ℓ denote the optimum dual solution to the concurrent restricted Steiner flow problem. Then $\tilde{z}^*$ equals the dual optimum value $W = \sum_e \ell(e) u(e)$. The dual solution satisfies the constraint $\sum_i d_i \tilde{\mathrm{w}}_\ell(S_i) \geq 1$, therefore it suffices to prove the following inequality:

$$\sum_i d_i \tilde{\mathrm{w}}_\ell(S_i) \leq O(\log T \log D) \frac{W}{\sigma^*}. \tag{1}$$

We start along the lines of the proof of Klein, Rao, Agrawal and Ravi [9]. Apply Lemma 2.2 for a source s, $p = T$, and an appropriately chosen ϵ. Delete the resulting set U from the graph along with all the edges in $e(U)$, and apply the lemma to another terminal in the remaining graph. Repeat until no more terminals are left. Let $\mathcal{A} = (U_0, U_1, \ldots, U_p)$ denote the resulting multicut with U_0 denoting the set remaining when there are no more terminals left. By adding the bounds on capacities of the cuts we get that the total capacity of the multicut does not exceed $2\epsilon W$.

Consider a terminal set S_i. The multicut $\mathcal{A} = (U_0, \ldots, U_p)$ partitions the set S_i. For every j such that $S_i \cap U_j \neq \emptyset$, select one element in the intersection to represent the intersection, and let S_i' denote the set of these representatives.

The fact that the diameter of each one of the sets U_j for $j \geq 1$ is bounded by $2\epsilon^{-1} \ln(T+1)$ implies that

(2) $$\tilde{w}_\ell(S_i) \le \tilde{w}_\ell(S_i') + 2|S_i - S_i'|\epsilon^{-1}\ln(T+1).$$

We use the induction hypothesis on the problem with demands d_i and set of terminals S_i' for $i = 1, \ldots, k$. We set $\epsilon = \frac{\sigma^* D}{8W}$ to guarantee that $D' = \sum_i d_i(|S_i'|-1)$, the quantity corresponding to D in the new problem, is at most half of D. Indeed, by definition, for each j, we have $u(U_j)/d(U_j) \ge \sigma^*$. Hence we have

$$\sigma^* \le \frac{\sum_j u(U_j)}{\sum_j d(U_j)} = \frac{2u(\mathcal{A})}{\sum_{i:|S_i'|\neq 1} d_i|S_i'|} \le \frac{2u(\mathcal{A})}{\sum_i d_i(|S_i'|-1)} \le \frac{4\epsilon W}{D'},$$

which implies $D' \le D/2$ by the choice of ϵ. By the induction hypothesis:

$$\sum_i d_i \tilde{w}_\ell(S_i') \le O(\log T \log D')\frac{W}{\sigma^*}.$$

Together with inequalities (2) and the fact that $D' \le D/2$, this implies the desired inequality (1).

Since we do not know the value of σ^*, in order to turn this proof into an algorithm we make do with an estimate of it. The estimate is initialized by considering single-node cuts and it is updated each time the algorithm discovers that the estimate is more than a factor of two away from the minimum observed value. ∎

In order to complete the proof of Theorem 2.1, it remains to improve the $\log D$ in the above theorem to $O(\log(kt))$. We proceed along the lines of the proof of the similar improvement in the two-terminal case [12]. We group the demands into groups according to the magnitude of the demand. Group p consists of all the commodities with demand between $(2t^3k)^{p-1}\min_i d_i$ and $(2t^3k)^p \min_i d_i$, so that the range of demands in a single group is limited by $2t^3k$. Let $\sigma^* = \min_{\mathcal{A}} u(\mathcal{A})/d(\mathcal{A})$ denote the minimum capacity-to-separated demand ratio, and let σ_p^* denote the minimum capacity-to-demand ratio in the problem induced by the commodities in group p. Similarly, let $\tilde{z}^*$ denote the optimum value of the concurrent restricted Steiner flow problem, and z_p^* denote the value of the concurrent restricted Steiner flow problem induced by the commodities in group p. Using simple rounding and Theorem 2.3 we get the following lemma:

Lemma 2.4 For every group p we have that $\tilde{z}_p^* \le \sigma_p^* \le O(\log T \log(kt))\tilde{z}_p^*$.

Clearly, $\tilde{z}^* \le \min_p \tilde{z}_p^*$. It remains to prove that the minimum can not be much larger than $\tilde{z}^*$. The following lemma was proved in essence in [12].

Lemma 2.5 [12] Consider two sets X and X' of 2-terminal commodities, and let D_X denote the sum of the demands of the commodities in X. Assume that there exists a feasible flow that satisfies demands of commodities in X, and another feasible flow that satisfies demands of the commodities in X'. Then there exists a feasible flow that satisfies the demands of the commodities in X and also, for each commodity i in X', satisfies demand $d_i - 2D_X$.

The corresponding lemma for the Steiner flow case is as follows.

Lemma 2.6 Let Q and Q' denote two sets of Steiner commodities, with at most t terminals per commodity. Assume that every commodity in Q' has demand at least $2t^3$ times more than the total demand of commodities in Q', and assume that there is a feasible packing of restricted Steiner trees satisfying the demands in Q, and another feasible packing of restricted Steiner trees satisfying the demands in Q'. Then there exists a feasible packing of restricted Steiner trees satisfying all the demands in Q and at least half of each of the demands in Q'.

Proof: We reduce the proof to Lemma 2.5. A restricted Steiner tree for terminal set S_i consists of $|S_i| - 1$ paths connecting pairs of terminals. A packing of restricted Steiner trees for terminal set S_i corresponds this way to a solution to a problem on up to $|S_i|^2/2$ two terminal nets with total demand $(|S_i| - 1)d_i$.

Let X denote the set of terminal pairs (with associated demands) corresponding to the restricted Steiner tree packing for Q, and let X' denote the set of terminal pairs (with associated demands) corresponding to the restricted Steiner tree packing for Q'. The sum of demands D_X of commodities in X is at most $t \sum_{j \in Q} d_j$. By Lemma 2.5 there exists a feasible flow that satisfies the demands of the commodities in X and also, for each commodity i in X', satisfies demand $d_i - 2D_X$. A feasible flow that satisfies the demands of the commodities in X is a restricted Steiner flow satisfying the demands in Q.

Consider the restricted Steiner flow of a commodity i in Q'. The total amount of flow removed from commodity i due to decreased flow from s to t in X' for two nodes $s, t \in S_i$ is at most $2D_X \le 2t \sum_{j \in Q} d_j$. For a commodity in $i \in Q'$ we have to consider paths between up to $t^2/2$ pairs of its terminals. Therefore, there exists a feasible restricted Steiner flow that satisfies the demands of commodities in Q, and at least $d_i - t^3 \sum_{j \in Q'} d_j$ demand of commodity i. The assumption guarantees that this satisfies at least half of the demand of commodity i. The claim follows by repeating this procedure for all commodities in Q'. ▮

Applying the above lemma, we can now prove that $\tilde{z}^*$ is not much smaller than the smallest $\tilde{z}_p^*$:

Lemma 2.7 $\min_p \tilde{z}_p^* \le 4\tilde{z}^*$.

Proof: First consider all the commodities in Q_p for even p. Observe that for any p, the commodities in Q_p and Q_{p+2} satisfy the assumptions of Lemma 2.6. Thus, by induction on p it is easy to show that there exists a feasible restricted Steiner flow that satisfies $\min_p \tilde{z}_p^*/2$ fraction of demands in all groups Q_p with even p. The claim follows from repeating the same argument for odd-numbered commodity groups. ▮

Proof of Theorem 2.1. Applying Lemmas 2.4 and 2.7, we get:

$$\tilde{z}^* \le \sigma^* \le \min_p \sigma_p^* \le O(\log T \log(kt)) \min_p \tilde{z}_p^* \le O(\log T \log(kt)) \cdot (4\tilde{z}^*).$$

▮

Next we consider the problem of finding a minimum-capacity multicut separating all sets of terminals. This problem is naturally related to another Steiner flow problem, the *maximum sum Steiner flow problem*. In this problem we are seeking to find a feasible Steiner flow that maximizes $\sum_{i,\tau \in \mathcal{T}_i} f_i(\tau)$. The minimum capacity of a multicut that separates all of the terminal sets gives a combinatorial upper bound on the maximum Steiner flow value. For computability reasons we shall consider the analogous *maximum sum restricted Steiner flow problem*, in which flow of commodity i is carried by restricted Steiner trees for set S_i, rather than Steiner trees spanning set S_i. Using the above algorithm and Theorem 2.1, we show that this upper bound is within a factor of $O(\log T \log k \log(kt))$ of the maximum flow value.

Theorem 2.8 Given an instance of the problem of separating k sets of terminals with minimum capacity, one can find a multicut $\mathcal{A}$ in polynomial time such that

$$\tilde{\phi}^* \leq \phi^* \leq \min_{\mathcal{B} \in \mathbb{B}} u(\mathcal{B}) \leq u(\mathcal{A}) \leq O(\log T \log(kt) \log k)\tilde{\phi}^*$$

where ϕ^* is the value of the corresponding maximum sum Steiner flow, $\tilde{\phi}^*$ is the value of the maximum sum restricted Steiner flow, and $\mathbb{B}$ is the set of multicuts that separate all of the given sets of terminals.

Proof: The only non-trivial inequality is the last one. To prove this inequality, consider the linear programming dual of the max-sum restricted Steiner flow problem. In the dual there is a nonnegative length variable $\ell(vw)$ for each edge. The dual of the maximum sum restricted Steiner flow problem is to minimize $W = \sum_{vw} u(e)\ell(e)$ subject to the conditions $\ell \geq 0$ and $\tilde{\mathrm{w}}_\ell(S_i) \geq 1$ for all commodities i. Let ℓ be such optimal dual.

We use Theorem 2.3 with terminal sets S_i for $i = 1, \ldots, k$ and demands $d_i = 1$ for every i. Note that $D \leq kt$ for this problem and that ℓ/k is a feasible solution to the dual of this concurrent restricted Steiner flow problem with objective value W/k. Therefore, W/k is an upper bound to the concurrent restricted Steiner flow value. By applying Theorem 2.3 we obtain a cut U with capacity-to-demand ratio at most $O(\log T \log(kt))W/k$.

Let k' denote the number of terminal sets separated by the cut U. We have that $u(U) \leq O(\log T \log(kt))\frac{k'}{k}W$. Now, apply induction for the problem with the k' terminal sets separated by the cut U removed. The length function ℓ is a feasible dual solution with objective value W, therefore W is an upper bound on the maximum sum restricted Steiner flow value for the problem with $k - k'$ terminals. Adding the cut U to the multicut resulting from the induction we get a multicut with capacity at most

$$O(\log T \log(kt))\frac{k'}{k}W + O(\log T \log(kt)) \ln(k - k')W.$$

Solving the recursion we get that the resulting multicut has capacity at most $O(\log T \log(kt) \log k)W$, as desired. ∎

Finally, we derive a weighted version of the decomposition theorem mentioned in the introduction. Let ℓ be a nonnegative length function and u a capacity function on the edges, and let $W = \sum_e u(e)\ell(e)$. Assume that we are given k sets of terminals S_i, such that $|S_i| \leq t$, $|\cup_i S_i| \leq T$, and the minimum cost of a Steiner tree spanning the set of terminals S_i is at least δ for every i.

Theorem 2.9 Consider a graph G with edge-capacities u, edge-lengths ℓ, k sets of terminals. In polynomial time one can find a multicut of capacity at most $\delta^{-1}O(\log T \log(kt) \log k)W$ that separates every set of terminals, where W and δ is defined above.

Proof: Observe that $\ell'(vw) = \ell/\delta$ is a feasible dual for the maximum-sum Steiner flow problem with terminal sets specified by $\{S_i\}$. Hence the value of the maximum-sum flow is at most $\sum_{vw} u(vw)\ell'(vw) = W/\delta$. By Theorem 2.8 we have that there exists a multicut whose capacity is bounded by $O(\frac{W}{\delta} \log T \log k \log(kt))$ that separates each set of terminals, and such a multicut can be found in polynomial time. ∎

2.3 Computing the Dual Solutions

The most time-consuming stage in the approximation algorithms described in the previous section is computation of ℓ, the dual solution to the restricted steiner flow problem. Although this LP can be solved in polynomial time by any interior-point linear programming algorithm, the resulting running time is pretty slow. Much faster algorithms can be obtained by using the techniques of [13]. In what follows we sketch how to apply these techniques and state the running times.

Let $\mathcal{T}_i$ denote a collection of restricted Steiner trees that span terminals S_i, let $f_i(\tau)$ be a nonnegative value for every tree $\tau \in \mathcal{T}_i$, and let $f_i(vw) = \sum\{f_i(\tau) \; : \; \tau \in \mathcal{T}_i \text{ and } vw \in \tau\}$. The corresponding LP is as follows:

$$\begin{array}{llcll} & \text{minimize } \lambda & & \text{subject to:} & \\ (3) & \sum_i f_i(vw) & \leq & \lambda u(vw) & \forall vw \in E \\ (4) & \sum_{\tau \in \mathcal{T}_i} f_i(\tau) & = & d_i & 1 \leq i \leq k \\ & f_i(\tau) & \geq & 0 & \end{array}$$

Let λ^* denote the optimal value of this LP. Since λ^* can be quickly approximated to within a factor of k, we can scale the capacities $u(vw)$ such that λ^* is between 1 and 2. The packing algorithm in [13] approximately solves LPs of the form "minimize λ subject to $Ax \leq \lambda b, x \in \mathcal{P}$", where $Ax \geq 0 \forall x \in \mathcal{P}$ and $\mathcal{P}$ is a convex set. The solution involves repeated invocations of a minimization subroutine over $\mathcal{P}$, which is assumed to be given. In our case, $\mathcal{P}$ is defined by (4). Observe that it can be written as a Cartesian product of simplices $\mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_k$, where minimization over each such simplex corresponds to finding a minimum-cost restricted steiner tree and can be done in $O(\min\{tm \log n, \mathcal{M}(n) \log n\})$ time, where $\mathcal{M}(n)$ is the time to multiply two $n \times n$ matrices.

The expected number of iterations of the packing algorithm in [13] is proportional to parameter $k \max_i \rho_i$; in our case $\rho_i = \max_{f_i \in \mathcal{P}_i} f_i(vw)/u(vw)$. Observe that restricting each $\mathcal{T}_i$ to include only trees that use edges with capacity above $d_i/(2m)$ can change the value of the LP by at most a constant factor. Thus, we have that $k \max_i \rho_i = O(mk)$. Using Theorem 2.7 from [13], the expected number of iterations is $O(km \log n)$, resulting in an $O^*(km \min\{mt, \mathcal{M}(n)\})$ algorithm that solves the above LP to within a constant factor. The deterministic version of this algorithm requires the same number of iterations, where at each iteration we need to optimize over all of $\mathcal{P}$, which can be done in $O^*(\mathcal{M}(n) + kt^2)$.

In fact, in addition to the primal solution, this algorithm produces dual variables $\ell(vw)$, such that $\lambda \sum_{vw} u(vw)\ell(vw) \leq O(\sum_i d_i \tilde{w}'_\ell(S_i))$, where $1/\lambda \geq O(\tilde{z}^*)$ and where $\tilde{w}'_\ell(S_i)$ denotes the minimum-cost restricted steiner tree that spans nodes S_i and that does not use edges with capacity below $d_i/(2m)$. Setting $\ell'(vw) = \max\{\ell(vw), \sum_{vw} u(vw)\ell(vw)/(mu(vw)\}$ we get

$$\lambda \sum_{vw} u(vw)\ell'(vw) \leq O(\sum_i d_i \tilde{\mathrm{w}}_\ell(S_i)).$$

Normalization produces a feasible dual with value within a constant factor of optimum. Thus, we have the following theorem:

Theorem 2.10 The approximate length function ℓ, needed for the algorithm in Theorem 2.1, can be computed in expected $O^*(km \min\{mt, \mathcal{M}\})$ time by a randomized algorithm and in $O^*(mk(\mathcal{M}(n) + kt^2))$ time by a deterministic one.

3 Directed Multicuts

3.1 Directed Decomposition

In this section we prove a new directed decomposition theorem that is the basis of our directed multicut results described in the next subsection. In essence, the decomposition theorem states that given a graph where each edge e has capacity $u(e)$ and length $\ell(e)$, we can remove some edges with small total capacity such that any two nodes that stay in the same strongly connected component are close.

Let ℓ be a nonnegative length function on the edges, and let $W = \sum_{vw} \ell(vw)u(vw)$. Assume that we are given k pairs of nodes (s_i, t_i), such that the round trip distance between s_i to t_i is at least δ. We will refer to these nodes as *terminals*. A multicut *separates* a given pair of nodes if after removing the edges associated with this multicut, these nodes are left in different strongly connected components. Note that there are graphs where the capacity of a multicut separating all the given node pairs is at least $\Omega(W/\delta)$. The following theorem gives the complementary upper bound.

Theorem 3.1 Consider a directed graph G with edge-capacities, edge-lengths, k pairs of terminals, and δ and W as defined above. In polynomial time one can find a multicut of capacity at most $\delta^{-1}O(\log^2 k)W$ that separates every terminal pair.

We are going to prove the theorem through a sequence of lemmas. The first lemma is a straightforward adaptation of the related undirected graph lemma in [5]. For a node set U let $e(U)$ denote the set of edges with at least one endpoint in U, $\Gamma^{\text{out}}(U) \subseteq e(U)$ denote the set of edges leaving U, and $\Gamma^{\text{in}}(U) \subseteq e(U)$ denote the set of edges entering U. For a set of edges F let $u(F) = \sum_{vw \in F} u(vw)$, and $W(F) = \sum_{vw \in F} u(vw)\ell(vw)$.

Lemma 3.2 For any positive ϵ and p, and any node s, there exists a set U containing s such that $u(\Gamma^{\text{out}}(U)) \leq \epsilon\,(W/p + W(e(U)))$, and every node in U is at a distance less than $\epsilon^{-1}\ln(p+1)$ from s.

In the undirected case, using the set U of the above lemma we get a region around the node s such that the distance between any pair of nodes in this region is bounded by $2\epsilon^{-1}\ln(p+1)$, *i.e.* all the nodes in this region are close to each other. Unfortunately, we cannot make this type of claim for the directed case. This fact is the main reason that the previously known undirected decomposition theorems cannot be easily extended to the directed case.

Now set $\epsilon = 4\delta^{-1}\ln(k+1)$. Let s be a source, and let R^{out} denote the "outregion" constructed using this ϵ, $p = k$, and node s. Similarly, (by considering graph with reverse edges) we construct an "inregion" R^{in}. By the choice of ϵ, for any node x in the intersection $R^{\text{out}} \cap R^{\text{in}}$, there is an s-to-x path and an x-to-s path, each of length less than $\delta/4$. Using the fact that the distance from s_i to t_i plus the distance from t_i to s_i is at least δ (by the assumptions stated above Theorem 3.1), we obtain the following lemma.

Lemma 3.3 There are no pairs of terminals $\{s_i, t_i\}$ in the intersection $R^{\text{out}} \cap R^{\text{in}}$.

Lemma 3.4 Consider a graph G with edge-capacities, edge-lengths, k pairs of terminals, and δ and W as defined above Theorem 3.1. In polynomial time one can find a multicut $\mathcal{A} = (U_1, \ldots, U_p)$ of capacity at most $\delta^{-1}O(\log k)W$ such that each part U_j contains at most $k/2$ pairs of terminals.

Proof: We give a recursive algorithm to construct the multicut. Select a source s, and construct the regions R^{out} and R^{in} using Lemma 3.2 as used in Lemma 3.3. It follows from Lemma 3.3 that one of the two regions contains at most $k/2$ terminal pairs. Let U be this region.

We delete region U from the graph along with all adjacent edges $e(U)$, and recursively construct a multicut $\mathcal{A}' = (U_1, \ldots, U_r)$ in the remaining graph. In applying Lemma 3.2 we use $p = k$ in all levels of the recursion. We add the part U to multicut $\mathcal{A}'$. If U was an outregion, the new multicut is $\mathcal{A} = (U, U_1, \ldots, U_r)$. If U was an inregion, the new multicut is $\mathcal{A} = (U_1, \ldots, U_r, U)$. In either case, the capacity we added in going from $\mathcal{A}'$ to $\mathcal{A}$ is at most $4\delta^{-1}\ln(k+1)(W/k + W(e(U))$.

Now consider the capacity of the resulting multicut $\mathcal{A} = (U_1, \ldots, U_p)$. Lemma 3.2 bounds the contribution of each region U_j to the capacity of the multicut. The bound consist of two parts, a term independent of the region, $4\delta^{-1}\ln(k+1)W/k$; and a term depending on the edges adjacent to the region. In each recursion we reduce the number of pairs by at least one, so the number p of parts is at most k. Furthermore, since the edges adjacent to set U have been deleted from the graph before the recursive call, the sets of edges whose weight is used to bound the contribution are disjoint. Hence the capacity of $\mathcal{A}$ is at most $8\delta^{-1}\ln(k+1)W$. ∎

Proof of Theorem 3.1: The Theorem follows from Lemma 3.4 by induction on k. First apply Lemma 3.4, then apply the inductive hypothesis to the problems defined by the graphs spanned by each of the regions U_j, the pairs of terminals included in U_j, and the original length function ℓ. ∎

3.2 From Decomposition to Directed Multicuts

In this section we show how to use the directed decomposition technique presented above in order to design an algorithms that find approximately optimal solutions for two multicut problems. In the *minimum multicut problem* we are given k pairs of terminals (s_i, t_i), and the problem is to find the minimum capacity multicut that separates all terminal pairs. In the *minimum ratio multicut problem* each pair of terminals (s_i, t_i) has an associated demand d_i, and the problem is to find a multicut whose capacity-to-separated demand ratio is minimal.

The multicut problems are closely related to two flow problems. A *multicommodity flow problem* (or *multiflow* problem for short) is defined by a directed graph G with nonnegative edge-capacities $u(vw)$, and k commodities. A *commodity* is specified by a source-sink pair (s_i, t_i), the terminals of commodity i. In a *symmetric st-flow* f each unit of flow sent from a source s to the corresponding sink t also has to be returned from t to s, though not necessarily along the same path. The amount of flow through an edge vw is defined to be the sum of the values of all paths in the flow that include the edge vw. A multiflow is *feasible* if, for each edge vw, the amount of flow through vw is at most its capacity $u(vw)$.

We consider two kinds of optimization problems concerning symmetric multiflows. The minimum capacity multicut problem is related to the *maximum sum symmetric multiflow* problem, in which the goal is to find a feasible symmetric multiflow maximizing the sum of the values of the flows. The minimum ratio multicut problem is related to the concurrent multiflow problem, in which we are given a nonnegative demand d_i for each commodity i. A multiflow f has *throughput* z if the value of commodity i's flow f_i is zd_i. The *concurrent multiflow problem* is to find a feasible multiflow with maximum throughput z^*. We will use D to denote the sum $\sum_i d_i$, and in bounds containing D we will assume that the demands are integral.

The dual linear programs for these multiflow problems can be viewed as linear programming relaxations of the corresponding multicut problems. The dual linear programs are similar. In each, there is a nonnegative length variable $\ell(vw)$ for each edge. Let $\mathrm{dist}_\ell(x, y)$ denote the distance from x to y in G with respect to the length function ℓ. The dual of the maximum sum symmetric multiflow problem is to minimize $W = \sum_{vw} u(e)\ell(e)$ subject to the conditions $\ell \geq 0$ and $\mathrm{dist}_\ell(s_i, t_i) + \mathrm{dist}_\ell(t_i, s_i) \geq 1$ for all commodities i. The dual of the concurrent flow problem is to minimize $W = \sum_{vw} u(e)\ell(e)$ subject to the conditions $\ell \geq 0$ and $\sum_i d_i[\mathrm{dist}_\ell(s_i, t_i) + \mathrm{dist}_\ell(t_i, s_i)] \geq 1$.

Now we show how to use the directed decomposition Theorem 3.1 to derive an algorithm that, given a feasible solution to the linear programming dual of the multiflow problem, finds a multicut with value at most a logarithmic factor above the value of the dual solution. Since an optimal dual solution value is equal to the optimal value of the multiflow problem, this proves both the approximate max-flow min-cut theorem, and that the multicut produced by the algorithm is close to optimal. The basic outline of our proof and algorithm is analogous to the outline of the algorithms for the undirected case [10, 9, 5]. The main difference is the use of the more involved directed decomposition given by Theorem 3.1.

Let ℓ be an optimal dual solution of the maximum multicommodity flow problem. The dual objective value is $W = \sum_{vw} u(vw)\ell(vw)$. The dual constraints ensure that $\mathrm{dist}_\ell(s_i, t_i) + \mathrm{dist}_\ell(t_i, s_i) \geq 1$, for all commodities i. Hence by Theorem 3.1 there is a multicut of capacity at most $O(\log^2 k)W$, that is, at most $O(\log^2 k)$ times the dual objective value. Since the dual objective value equals the primal objective value, the maximum multiflow value, we obtain the following theorem.

Theorem 3.5 Given an instance of directed multicut problem with k terminal pairs, let $\mathbb{B}$ denote the set of the multicuts that separate all the demand pairs. Then one can find a multicut $\mathcal{A}$ in polynomial time such that

$$\phi^* \leq \min_{\mathcal{B} \in \mathbb{B}} u(\mathcal{B}) \leq u(\mathcal{A}) \leq O(\log^2 k)\phi^*$$

where ϕ^* is the value of the maximum sum multiflow in the corresponding directed symmetric multiflow problem.

For the minimum ratio multicut problem we have the following theorem:

Theorem 3.6 Given an instance of directed minimum ratio multicut problem with k terminal pairs and symmetric demands, let $\mathbb{C}$ denote the set of all multicuts that separate some demand pairs. Then one can find a multicut $\mathcal{A}$ in polynomial time such that

$$z^* \leq \min_{\mathcal{C}\in\mathbb{C}} \frac{u(\mathcal{C})}{d(\mathcal{C})} \leq \frac{u(\mathcal{A})}{d(\mathcal{A})} \leq O(\log^3 k) z^*,$$

where for a multicut $\mathcal{C}$ we use $d(\mathcal{C})$ to denote the sum of the demands separated by $\mathcal{C}$, and z^* denotes the maximum value of the corresponding directed concurrent multiflow problem.

We start with proving a weaker version of the theorem with one $\log k$ replaced by a $\log D$ in the upper bound. Similarly to the case of undirected graphs, the proofs of Theorems 3.1, and 3.5 can be modified along the lines of [9] to prove this version. Kahale [8] has shown that in the undirected case this weaker form of the min-max theorem for concurrent multiflows can be derived directly from the claim of maximum sum multiflow theorem, instead of modifying the proof of this theorem. Kahale's derivation, which can be easily adapted to the directed case, is based on the following lemma.

Lemma 3.7 [8] For given positive $\delta_1, \ldots, \delta_k$, and integers $d_1, \ldots, d_k$ there exists a set $Q \subseteq \{1, \ldots, k\}$ such that

$$\frac{\sum_{i=1}^k d_i\delta_i}{\ln(1+\sum_{i=1}^k d_i)} \leq (\sum_{i\in Q} d_i)(\min_{i\in Q} \delta_i). \tag{5}$$

Theorem 3.8 Given a directed concurrent flow problem with symmetric demands, let $\mathbb{C}$ denote the set of all multicuts that separate some demand pairs. Then one can find a multicut $\mathcal{A}$ such that:

$$z^* \leq \min_{\mathcal{C}\in\mathbb{C}} \frac{u(\mathcal{C})}{d(\mathcal{C})} \leq \frac{u(\mathcal{A})}{d(\mathcal{A})} \leq O(\log D \log^2 k) z^*,$$

where for a multicut $\mathcal{C}$ we use $d(\mathcal{C})$ to denote the sum of the demands separated by $\mathcal{C}$.

Proof: The only non-trivial inequality is the last one. To prove the last inequality let ℓ be an optimal dual solution for the concurrent multiflow problem. The dual objective value is $W = \sum_{vw} u(vw)\ell(vw)$. For commodity i let $\delta_i = \mathrm{dist}_\ell(s_i,t_i) + \mathrm{dist}_\ell(t_i,s_i)$. The constraint of the dual linear program ensures that $\sum_i d_i\delta_i \geq 1$. Apply Lemma 3.7 to obtain a set Q, and let $\delta = \min_{i\in Q}\delta_i$. The inequalities (5) and $\sum_i d_i\delta_i \geq 1$ imply that

$$\delta \sum_{i\in Q} d_i \geq \frac{1}{\ln(1+D)} \tag{6}$$

We apply Theorem 3.1 to the length function ℓ, and the commodities in Q. The theorem shows that there is a multicut $\mathcal{A}$ separating all commodities in Q of capacity $\frac{1}{\delta}O(\log^2 k)W$. The demand separated by $\mathcal{A}$ is at least $\sum_{i \in Q} d_i$. Using (6) we see that the capacity-to-demand ratio of $\mathcal{A}$ is

$$\frac{u(\mathcal{A})}{d(\mathcal{A})} = O(\log D \log^2 k)W.$$

Since $W = \sum_{vw} u(vw)\ell(vw)$ is the dual objective value and is therefore equal to the primal objective value z^*, we obtain the theorem. ∎

Next we show how to get the stronger bound of Theorem 3.6, independent of the size of the demands, by extending the technique of Plotkin and Tardos [12] to the case of directed graphs. In essence we prove that up to small constant factors the worst min-cut/max-flow ratios occur in problems with integer demands that are bounded by a small-degree polynomial in k.

First, observe that rounding the demands up to multiples of $\min_i d_i$ can change the value of z^* by at most a factor of 2. Moreover, this rounding cannot change the demand across any multicut by more than a factor of 2. This implies that the value of D in Theorem 3.8 can be replaced by $\hat{D} = \sum_i d_i / \min_i d_i$ without assuming the integrality of the demands.

Next we group the demands into groups according to the magnitude of the demand. Group p consists of all the commodities with demand between $(10k^2)^{p-1} \min_i d_i$ and $(10k^2)^p \min_i d_i$, so that the range of demands in a single group is limited by $10k^2$. Let $\sigma^* = \min_{\mathcal{A}} u(\mathcal{A})/d(\mathcal{A})$ denote the minimum capacity-to-separated demand ratio, and let σ_p^* denote the minimum capacity-to-demand ratio in the problem induced by the commodities in group p. Similarly, let z^* denote the optimum value of the concurrent flow problem, and z_p^* denote the value of the concurrent flow problem induced by the commodities in group p. Theorem 3.8 together with the fact that the range of the demands in each group is bounded by $O(k^2)$ and the above observation about demand rounding, gives the following lemma.

Lemma 3.9 For every group p we have that $z_p^* \leq \sigma_p \leq O(\log^3 k) z_p^*$.

Next we prove that the optimum value of the concurrent flow z^* is within a constant factor of the $\min_p z_p^*$. Clearly, $z^* \leq z_p^*$ for all p. The next lemma encapsulates the main ideas needed to prove the opposite inequality.

Lemma 3.10 Let Q and Q' denote two sets of demands. Assume that every commodity in Q' has demand at least $(8k+4)$ times more than the total demand of commodities in Q, and assume that there is both a feasible flow f satisfying the demands in Q, and a feasible flow f' satisfying the demands in Q'. Then there exists a flow satisfying all the demands in Q and at least half of each of the demands in Q' such that the flow through an edge vw is limited by $(1 + \frac{1}{k})u(vw)$.

Proof: It is no loss of generality to assume that both of the flows f and f', and the capacities u are rational. Multiplying up with the common denominator, we can further assume without loss of generality that f, f', and u are integral. We will regard an edge e with capacity $u(e)$ as a collection of $u(e)$ parallel edges, and flows f, f' as collections of edge-disjoint paths in the network. Notice that nothing prevents a flow path of a commodity in Q from using the same edge as some flow path of a commodity in Q'. We will call such situation a *collision*.

The idea of the proof is to delete a small number of flow paths of commodities in Q' and reroute the flow paths of the commodities in Q, in order to eliminate all collisions between flow paths of the commodities in Q and Q'. The rerouting procedure, as described below, creates both unit-flow paths that carry up to one unit of flow each, and $(1/k)$-flow paths that carry $1/k$ units of flow. The goal is to ensure that a single unit-capacity edge will participate in at most one unit-flow path and one $(1/k)$-flow paths. We will eliminate collisions one at a time, giving a pseudopolynomial algorithm for rerouting. Note, however, that the flow, whose existence is proved by this lemma, can be constructed without referring to this proof, by running a multicommodity flow algorithm.

First, we will concentrate on eliminating collisions with a single commodity $j \in Q'$. We remove some flow paths of commodity j, and reroute some of the flow paths of a commodities in Q'. We will show that this procedure will not create further collisions, except that unit-capacity edges used by the flow paths of commodity j might participate in an $(1/k)$-flow paths in addition to a unit-flow paths. Repeating this procedure for each commodity in Q' we obtain the lemma.

To eliminate collisions with commodity j, we first create a set of *beginning segments* $\mathcal{P}_1$ of the flow paths of commodities Q. Initially, $\mathcal{P}_1$ consists of all flow paths of commodities in Q. On each flow path of commodity j, we will note the *last collision* with a path in $\mathcal{P}_1$, and denote the set of these "last collision edges" by L_j. Consider a path $P \in \mathcal{P}_1$ that uses more than k edges of L_j, *i.e.* $|P \cap L_j| > k$. Let P_1 denote the part of this path from its source until the kth collision, and delete the rest of this path. This operation changes L_j, by exposing new collisions as the last ones on some flow paths of commodity j. However, the number of collisions is decreasing, and hence after a finite number of such changes the set L_j will have of at most k collision edges on each of the paths in $\mathcal{P}_1$.

In a symmetric way create a set of *end segments* $\mathcal{P}_2$ of the flow paths of commodities Q. Again, we start with letting $\mathcal{P}_2$ consist of all flow paths of commodity Q. We consider the "first collision" edges F_j. In a way analogous to the above we delete the beginning segments of paths in $\mathcal{P}_2$ until each path in $\mathcal{P}_2$ has at most k edges in F_j.

An example of the construction of the start and end segments is shown in Figure 1. The flow paths of the jth commodity are going from top to bottom, and the flow paths of the three commodities (a, b, and c) in Q are going from left to right. Collisions are represented by black and grey circles; black circles crossed out with "x" represent collisions in L_j and plain black circles represent collisions in F_j. To simplify the figure we constructed beginning and end segments with keeping only 2 collisions on every flow path in Q. The "end segments" are shown as dashed lines, the "beginning segments" as solid lines.

Now delete all the flow paths of commodity j that contain edges of L_j and F_j (that is, those that cause collisions). This ensures that there are no collisions between remaining flow paths of commodity j and paths in $\mathcal{P}_1 \cup \mathcal{P}_2$.

For each flow path P of a commodity in Q with source s and sink t, we have two path segments $P_1 = (s, v) \in \mathcal{P}_1$ and $P_2 = (w, t) \in \mathcal{P}_2$, where v and w are some intermediate nodes on the path P. If $P = P_1 \cup P_2$, then there are no collisions on P, and we leave it as it was originally routed. Otherwise, there are k paths associated with collisions $L_j \cap P_1$ of the jth commodity, and another k paths of the jth commodity associated with collisions $F_j \cap P_2$. Split the unit flow associated with P_1 into k pieces, each carrying $(1/k)$ units of flow, and route each piece from s to the sink t_j of the jth commodity using part of the segment P_1, and one of the removed flow paths of the

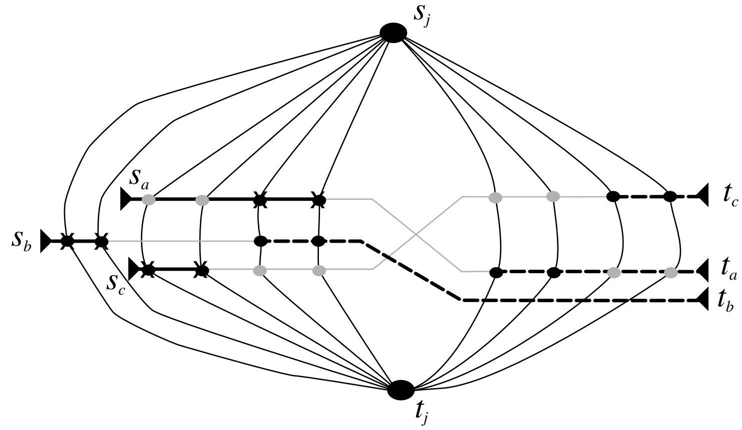

Figure 1: Construction of the start and end segments.

jth commodity associated with one of the collisions in $L_j \cap P_1$. Similarly, split the unit of flow associated with P_2, and route from the source of the jth commodity to t, using the removed flow paths of the jth commodity associated with $F_j \cap P_2$.

Observe that the above procedure adds only $(1/k)$-flow paths that use edges freed by the commodity $j \in Q'$. The added paths can not collide with flow paths of commodities in Q'. Moreover, the procedure does not create any collisions between unit flow paths of commodities in Q. New collisions can be created only on edges freed by removing flow paths of commodity j. Note that $(1/k)$-flow paths that connect start segments to t_j are edge-disjoint and do not have edges in common with the start segments. Similarly, $(1/k)$-flow paths that connect s_j to the end segments are edge-disjoint and do not have edges in common with the end segments. Thus, the worst that can happen is that some of the edges freed by removing flow paths of commodity j are used by a single unit-flow path and a single $(1/k)$-flow path at the same time.

Now we repeat the rerouting procedure in order to get rid of collisions with the flow of the jth commodity from t_j to s_j. After this rerouting, each flow path of a commodity i in Q is either routed from the corresponding source s_i to the corresponding sink t_i, or is split into two parts, where the first part routes a unit of flow from s_i to either t_j or s_j, and the second routes a unit from either s_j or t_j to t_i, respectively. By removing additional d_i flow paths of the jth commodity from s_j to t_j and another d_i flow paths from t_j to s_j, we create sufficient number of free paths that allow us to complete the routing of the ith commodity.

The above procedure removed at most $2k+1$ paths from of commodity j from s_j to t_j for every flow path of a commodity in Q; same bound on the removed flow from t_j to s_j. By symmetry, there are two flow path for every unit of demand, and amount of unsatisfied demand of commodity j is bounded by $4k+2$ times the total demand of the commodities in Q. The conditions of the lemma imply that this is below $d_j/2$. Notice that after rerouting that eliminates collisions with j, an edge can be assigned at most $1+1/k$ units of flow: one unit due to unit-flow path of a commodity in Q, and the rest due to a $(1/k)$-flow path path of a commodity in Q. Recall that this can happen only on the edges originally used by commodity j. Hence, after eliminating collisions with all the commodities in Q, it is sufficient to increase the capacity by an $(1+1/k)$ factor to be able to route all of the commodities in Q and half of the demand of the commodities in Q' simultaneously. ∎

Theorem 3.11 $z^* \leq \min_p z_p^* \leq 8z^*$

Proof: Clearly, $z^* \leq z_p^*$ for all p. To prove the other inequality, let $z = \min_p z_p^*$. First we consider all of the commodities in Q_p with even p. For every such group Q_p, we have a feasible multicommodity flow f_p satisfying demands zd_i, where $i \in Q_p$. Let k' denote the number of non-empty even indexed groups. We claim that there exists a feasible multicommodity flow f_{even} that satisfies demands $\frac{z}{2}d_i$ for each commodity i in group Q_p with even p, where the flow though an edge vw is limited by $(1+1/k)^{k'}u(vw)$.

We prove the above claim by induction on the number of non-empty even-indexed groups. Let k_p denote the number of non-empty even indexed groups among the first p groups. The claim is obviously true for p such that $k_p = 1$. To prove that the required flow exists for some even p such that Q_p is non-empty, apply Lemma 3.10 for commodity groups $Q = Q_2 \cup \ldots \cup Q_{p-2}$, and $Q' = Q_p$. Inductively, assume that there exists a flow f that satisfies at least $\frac{z}{2}$ fraction of the demand of each commodity in Q, and the flow through an edge vw is limited by $(1+1/k)^{k_{p-2}}u(vw)$. Lemma 3.10 applied to f and f_p implies that there exists a flow that satisfies the demands that were satisfied by f and at least $1/2$ of the demands satisfied by flow f_p and the flow through an edge vw is limited by $(1+1/k)^{k_p}u(vw)$.

Applying the same argument for the sets Q_p for odd p, we conclude that there exist a feasible flow f_{odd} that satisfies at least $\frac{z}{2}$ fraction of each demand in odd indexed groups such that the flow though an edge vw is limited by $(1+1/k)^{k''}u(vw)$, where k'' denotes the number of non-empty odd indexed groups. Therefore, there exists a feasible flow f that satisfies $z/2$ fraction of each one of the demands such that the flow though an edge vw is limited by $[(1+1/k)^{k'} + (1+1/k)^{k''}]u(vw)$.

Observe that $k' + k''$, the number of non-empty groups, is at most k, and $(1+1/k)^{k'} + (1+1/k)^{k''} \leq (1+1/k)^k + 1 \leq 4$. Diving the flow by 4 we get that there exists a feasible flow f that satisfies demands $\frac{z}{8}d_i$ for every commodity i. ∎

Proof of Theorem 3.6. By combining Lemma 3.9 and Theorem 3.11, we get:

$$z^* \leq \sigma^* \leq \min_p \sigma_p^* \leq O(\log^3 n) \min_p z_p^* \leq O(\log^3 n) z^*.$$

∎

3.3 Application: 2-CNF Clause Deletion Problem

One direct application of our minimum multicut algorithm yields an approximation algorithm for the *2-CNF clause-deletion* problem, i.e. the problem of finding the minimum-weight set of clauses in a 2-CNF formula whose deletion makes the formula satisfiable, where k is the number of literals in the formula. The exact variant of this problem is equivalent to the MAX 2-SAT problem, in which one seeks the maximum subset of clauses comprising a satisfiable formula. However, in the context of approximation algorithms, these problems seem to differ in difficulty. It is easy to approximate MAX 2-SAT, and even MAX SAT to within a small constant factor [7, 16, 6]. However, the number of clauses discarded by these algorithms cannot be expected to be within a polylogarithmic bound of the minimum.

The basis for the reduction is the following well-known graph construction. Given a 2-SAT formula F, we can assume its clauses have the form $p \to q$, where p and q are literals (variables or negations of variables). To reduce this problem to the directed multicut problem, we construct an auxiliary graph $G(F)$ with a node for each literal. For each clause $p \to q$ of weight w, there is an edge $p \to q$ and an edge $\bar{q} \to \bar{p}$, each of capacity w.

Lemma 3.12 The formula F is satisfiable if and only if no strongly connected component of $G(F)$ contains both a variable and its negation.

Proof: Note that for any satisfying assignment, the literals in a directed cycle must all receive the same truth value. The same therefore holds for a strongly connected component. The "only-if" direction is then immediate.

Conversely, suppose that no strongly connected component contains both a variable and its negation. Note that by the construction of the graph, for each cycle $p_1 \to \cdots \to p_n \to p_1$, there is a negated cycle $\bar{p}_1 \to \bar{p}_n \to \cdots \to \bar{p}_1$. Hence the strongly connected components come in pairs: for a strongly connected component containing some set of literals, there is a strongly connected component containing exactly the negations of these literals.

Consider the graph G' obtained from $G(F)$ by treating each strongly connected component as a single supernode. Then G' is a directed acyclic graph. Let $\mathcal{V}$ be a strongly connected component that is a sink in G'; i.e., $\mathcal{V}$ has no outgoing edges. Let $\bar{\mathcal{V}}$ denote the strongly connected component consisting of the negations of the literals in $\mathcal{V}$. Observe that $\bar{\mathcal{V}}$ has no incoming edges in G'. We assign true to all the literals in $\mathcal{V}$, and false to all the literals in $\bar{\mathcal{V}}$. We claim that this ensures the truth of every clause involving a literal in $\mathcal{V}$. Such a clause must be of the form $p \to q$, where q is in $\mathcal{V}$, because $\mathcal{V}$ has no outgoing edges. Since q is in $\mathcal{V}$, we have assigned it true, so the clause is satisfied. Since $\bar{\mathcal{V}}$ has no incoming edges, all clauses involving a literal in $\bar{\mathcal{V}}$ are of the form $q \to p$, where q is in $\bar{\mathcal{V}}$, and hence are automatically satisfied. Thus, we can delete $\mathcal{V}$, $\bar{\mathcal{V}}$ and all the associated clauses, and recurse. ∎

The 2-CNF clause-deletion problem is related to the minimum multicut problem by the following lemma.

Lemma 3.13 The minimum capacity of a multicut in $G(F)$ separating every variable from its negation is at most twice the minimum weight of a set of clauses whose deletion makes F satisfiable. Conversely, given any multicut $\mathcal{A}$ that separates every variable from its negation, one can find a set of clauses having weight at most the capacity of $\mathcal{A}$ whose deletion makes the formula satisfiable.

Proof: Let S be a set of clauses whose deletion from F yields a satisfiable formula F'. Let C be the set of edges corresponding to the clauses in S. For each clause there are two edges, each having capacity equal to the cost of the clause. Hence the capacity of C is twice the cost of S. Then $G(F) - C$ is the auxiliary graph $G(F')$ for the satisfiable formula F'. By Lemma 3.12, $G(F')$ has no variable and its negation in the same strongly connected component, and thus a subset of C is a multicut separating each variable from its negation.

Conversely, given a multicut $\mathcal{A}$, delete from F the clauses corresponding to edges in $\mathcal{A}$. For the resulting formula F', the auxiliary graph $G(F')$ is a subgraph of $G(F) - \mathcal{A}$. Since the latter graph has no variable and its negation in the same strongly connected component, neither does the former. Hence by Lemma 3.12, F' is satisfiable. ∎

We can use the algorithm of Theorem 3.5 to find a multicut separating each variable from its negation. The multicut found has capacity at most $O(\log^2 k)$ times the minimum total capacity of an edge set whose deletion separates each variable from its negation. We obtain the following theorem.

Theorem 3.14 There exists a polynomial-time algorithm to approximate the minimum-weight deletion problem for 2-CNF formulae. The solution output has weight $O(\log^2 k)$ times optimal, where k is the number of variables in the formula.

Recently, Even et al [4] observed that the fractional directed cycle packing result of Seymour [14] directly implies an $O(\log n \log\log n)$ approximation algorithm to the directed multicuts problem. Since our reduction of 2-CNF deletion problem results in a graph with $n = O(k^2)$ nodes, this implies that the 2-CNF minimum deletion problem can be approximated to within a factor of $O(\log k \log\log k)$.

3.4 Computing the dual solution

As in the Steiner cuts case, the most time-consuming part of our directed multicut algorithms is computing the solution to the corresponding LP, which in this case corresponds to the dual of maximum sum directed multiflow problem.

Instead of solving the directed multiflow problem, we will solve the following LP: Let Γ_i denote a collection of directed path pairs, where in each pair one path is from s_i to t_i and the other one is from t_i to s_i. Let $f_i(\gamma)$ be a nonnegative value for every path-pair $\gamma \in \Gamma_i$, and let $f_i(vw) = \sum\{f_i(\gamma) : \gamma \in \Gamma_i \text{ and } vw \in \gamma\}$. The corresponding LP is as follows:

$$\text{minimize } \lambda \qquad \text{subject to:}$$

$$(7) \qquad \sum_i f_i(vw) \leq \lambda u(vw) \qquad \forall vw \in E$$

$$(8) \qquad \sum_i \sum_{\gamma\in\Gamma_i} f_i(\gamma) = 1 \qquad 1 \leq i \leq k$$

$$f_i(\gamma) \geq 0 \qquad \forall \gamma \in \Gamma_i, 1 \leq i \leq k$$

(9)

Observe that any feasible primal solution to this LP with value λ can be directly transformed into a primal feasible solution of symmetric maximum multiflow with value $\phi = 1/(2\lambda)$. Also, any primal feasible solution to symmetric max multiflow with value ϕ can be transformed to a primal feasible solution to the above problem with $\lambda = 1/\phi$. Thus, we have:

$$(10) \qquad \frac{1}{\lambda^*} \leq 2\phi^* \leq \frac{2}{\lambda^*}$$

where λ^* denotes the optimum solution to the above LP. Moreover, since we can easily approximate λ to within a factor of k, we can use binary search to scale capacities such that the problem reduces to the case where the optimum solution λ^* is between 1 and 2.

Observe that restricting each Γ_i to include only path pairs that do not use edges with capacity below $1/(2m)$ can change the value of the linear program by at most a factor of 2. Let Γ_i' denote the set of restricted path pairs. Thus, by Theorem 2.7 of [13], we can find a constant factor approximation to the above LP in $O(m \log n)$ iterations, where each iteration involves finding a shortest $\gamma \in \cup_i \Gamma_i'$. This can be done in $O(\mathcal{M}(n) \log n)$ time by first computing all-pairs shortest paths, where $\mathcal{M}(n)$ denotes the time to multiply two $n \times n$ matrices.

In addition to the primal solution $\{f_i\}$, the algorithm produces a vector ℓ', such that $\lambda \leq O(\lambda^*)$ where λ^* is the optimum value and such that

$$\lambda \sum_{vw} \ell'(vw) u(vw) \leq \beta \min_{\gamma \in \cup_i \Gamma_i} \sum_{vw \in \gamma} \ell'(vw)$$

for some constant β. Note if we set $\ell''(vw) = \max\{\ell'(vw), \sum_{vw} \ell'(vw)u(vw)/(mu(vw)\}$, it will satisfy the same inequality (with a different constant β') with Γ_i' replaced by Γ_i.

Let $\ell = \beta' \ell'' / (\lambda \sum_{vw} \ell''(vw) u(vw))$. The length of the shortest directed cycle with respect to ℓ is above 1. Moreover, we have

$$\sum_{vw} u(vw)\ell(vw) = \beta/\lambda \leq \beta/\lambda^* \leq 2\beta\phi^*.$$

The above discussion implies the following claim.

Theorem 3.15 The length function ℓ needed to compute the multicut in Theorem 3.5 can be computed in $O^*(m\mathcal{M}(n))$ time.

Remark. In [4] the authors develop an algorithm similar to the packing algorithm but not based on [13] to find the length function ℓ. Using our notation their running time is $O^*(m^2\mathcal{M}(n))$.

Acknowledgments

We are grateful to Jon Kleinberg for many helpful discussions. In particular, we would like to thank him for pointing out a crucial flaw in an earlier approach to making the min-multicut/max-flow bound for the directed symmetric concurrent multiflow problem independent of D, the sum of the demands. We would like to thank the author of [4] for pointing out reference [14].

References

[1] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Improved routing strategies with succinct tables. *J. Alg.*, 11:307–341, 1990.

[2] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network Decomposition and Locality in Distributed Computation. In *Proc. 30th IEEE Annual Symposium on Foundations of Computer Science*, pages 364–369, 1989.

[3] B. Awerbuch and D. Peleg. Network synchronization with polylogarithmic overhead. In *Proc. 31st IEEE Annual Symposium on Foundations of Computer Science*, pages 514–522, 1990.

[4] G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multi-cuts in directed graphs. Unpublished manuscript, May 1994.

[5] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, May 1993.

[6] M. X. Goemans and D. P. Williamson. A new $\frac{3}{4}$-approximation algorithm for MAX SAT. In *Proc. Third Conference on Integer Programming and Combinatorial Optimization*, pages 313–322, 1993.

[7] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comp. and Syst. Sci.*, 9:256–278.

[8] N. Kahale. On reducing the cut ratio to the multicut problem. Technical Report TR-93-78, DIMACS, 1993.

[9] P. N. Klein, S. Rao, A. Agrawal, and R. Ravi. An approximate max-flow min-cut relation for multicommodity flow, with applications. *Combinatorica*. to appear. Preliminary version appeared as "Approximation through multicommodity flow," In *Proc. 31th IEEE Annual Symposium on Foundations of Computer Science*, pages 726–737, 1990.

[10] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proc. 29th IEEE Annual Symposium on Foundations of Computer Science*, pages 422–431, 1988.

[11] D. Peleg and E. Upfal. A tradeoff between size and efficiency for routing tables. *J. ACM*, 36:510–530, 1989.

[12] S. Plotkin and É. Tardos. Improved bounds on the max-flow min-cut ratio for multicommodity flows. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, May 1993.

[13] S. A. Plotkin, D. Shmoys, and É. Tardos. Fast Approximation Algorithms for Fractional Packing and Covering. In *Proc. 32nd IEEE Annual Symposium on Foundations of Computer Science*, 1991.

[14] P.D. Seymour. Packing directed circuits fractionally. Unpublished manuscript. Revised November 1993, June 1992.

[15] P.M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *Proc. 30th IEEE Annual Symposium on Foundations of Computer Science*, pages 338–343, 1989.

[16] M. Yannakakis. On the approximation of maximum satisfiability. In *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1–9, 1992.