Algorithmica (1997) 18: 3–20

Algorithmica
© 1997 Springer-Verlag New York Inc.

Primal–Dual Approximation Algorithms for Integral Flow and Multicut in Trees

N. Garg,[1] V. V. Vazirani,[1] and M. Yannakakis[2]

Abstract. We study the maximum integral multicommodity flow problem and the minimum multicut problem restricted to trees. This restriction is quite rich and contains as special cases classical optimization problems such as matching and vertex cover for general graphs. It is shown that both the maximum integral multicommodity flow and the minimum multicut problem are NP-hard and MAX SNP-hard on trees, although the maximum integral flow can be computed in polynomial time if the edges have unit capacity. We present an efficient algorithm that computes a multicut and integral flow such that the weight of the multicut is at most twice the value of the flow. This gives a 2-approximation algorithm for minimum multicut and a $\frac{1}{2}$-approximation algorithm for maximum integral multicommodity flow in trees.

Key Words. Integral multicommodity flow, Multicut, Approximation algorithm, MAX SNP-hard.

1. Introduction. The *Multicut* problem is defined as follows: Given a graph $G = (V, E)$ with a positive weight (or capacity) $c(e)$ on every edge $e \in E$, and a list of vertex pairs, (s_i, t_i), $1 \leq i \leq k$, find a minimum weight set of edges separating each pair of vertices in the list. We call such a set of edges a *multicut*. In this paper we deal with undirected graphs. The multicut problem was posed as early as 1969 by Hu [17]. For $k = 1$, the problem coincides with the ordinary s–t minimum cut problem. The problem is also polynomial-time solvable when $k = 2$, by using two applications of a minimum s–t cut algorithm [31].

The multicut problem includes as a special case the multiway (or multiterminal) cut problem [8], where instead of a list of pairs of vertices we are given a set of vertices (called terminals) and we wish to find a minimum weight set of edges whose removal separates every pair of terminals. Clearly, the multiway cut problem can be encoded as a multicut problem by including in the list of vertex pairs all distinct pairs of terminals. It was shown in [8] that the multiway cut problem is NP-hard and MAX SNP-hard for any fixed number $k \geq 3$ of terminals; hence the same is true of the multicut problem.

In this paper we address the special case of finding a minimum multicut when the input graph is a tree (the problem is dealt with in full generality in [12]). The minimum multiway cut in trees can be found in polynomial time using a straightforward dynamic programming approach [7]. However, for multicuts, NP-hardness sets in much earlier; we show that computing the minimum multicut is NP-hard and MAX SNP-hard even on unweighted trees of height 1.

[1] Department of Computer Science and Engineering, Indian Institute of Technology, New Delhi, India.
[2] AT&T Bell Laboratories, Murray Hill, NJ 07974, USA.

Received November 15, 1994; revised October 2, 1995. Communicated by M. X. Goemans.

We approach this intractability of the minimum multicut problem by considering a multicommodity flow problem; the formulation we deal with associates a commodity with each vertex pair and requires maximizing the sum of the flows routed subject to capacity and flow conservation requirements. Clearly, the maximum multicommodity flow is bounded from above by the minimum multicut; the question is whether equality holds. Consider a tree of height 1 with three leaves. Each pair of leaf vertices form the source–sink pair of a commodity. All edges have unit capacity. The maximum flow in the tree is $\frac{3}{2}$ whereas the minimum multicut has weight 2.

In this situation the best that can be hoped for is an approximate max-flow min-multicut theorem. Such an approximate theorem for general graphs was shown in [12], stating that the maximum flow and the minimum multicut are within a multiplicative factor $O(\log k)$ of each other (improving an earlier $O(\log^3 n)$ bound of [19]). Furthermore, the gap is tight up to a constant factor, i.e., there are instances in which the ratio between the minimum multicut and the maximum flow is $\Omega(\log k)$ [12]. (There is also another version of multicommodity flows and cuts that has been studied in the literature [22], [19], in which the commodities have associated demands and we wish to find a two-way cut that minimizes the ratio of the capacity over the demand across the cut. It is easy to see that this min-ratio cut problem is trivial on trees, so we do not discuss this version in this paper.)

We prove here that a tighter relationship between the maximum flow and the minimum multicut holds for trees. Furthermore, the relative simplicity of this setting (input graph a tree) allows us to consider a stronger version of flow, namely, integral flow; a commodity can now be routed only in integral units. Clearly, the maximum integral flow cannot exceed the maximum flow. Our main result is an approximate maximum-integral-flow minimum-multicut theorem.

THEOREM 1.1 (Approximate Max-Integral-Flow Min-Multicut Theorem). *For trees,*

$$\textit{maximum integral flow} \leq \textit{minimum multicut} \leq 2 \cdot \textit{maximum integral flow}.$$

Our proof of this theorem is an efficient algorithm for computing a multicut and an integral flow such that the weight of the multicut is at most twice the value of the flow. This also gives us a 2-approximation algorithm for the minimum multicut problem in trees and a $\frac{1}{2}$-approximation algorithm for the maximum integral flow problem in trees (we show that this problem is NP-hard and MAX SNP-hard). The MAX SNP-hardness [25] of the multicut and integral flow problems implies that no polynomial-time approximation scheme exists unless $P = NP$ [1].

A specially interesting aspect of the algorithm is the methodology used in designing it—it is based on a primal–dual approach. The primal–dual method has been used extensively in the past for solving problems in P. The recent work of Goemans and Williamson [14] and Williamson *et al.* [30] demonstrates, conclusively, the effectiveness of this method in the context of approximation algorithms for NP-hard optimization problems.

The maximum integral flow problem has been extensively studied; see e.g., [10] and [27]. For a $k = 1$ commodity it coincides with the ordinary maximum flow problem, but for $k \geq 2$ commodities it becomes NP-hard [9]. In the case of unit edge capacities, the problem of finding the maximum integral flow is the same as that of finding a maximum cardinality set of edge-disjoint paths between the specified source–sink pairs; in this case the problem can be solved in polynomial time for fixed k by the results of Robertson and Seymour [26], and becomes NP-hard for general k. We do not know of approximation algorithms for any other NP-hard cases of the maximum integral flow problem besides ours. Variants of the problem where the commodities have specified demands that have to be (approximately) satisfied are studied in [20] and [28]. We show by means of an example that even for grid graphs, the ratio of the maximum (fractional) flow to the maximum integral flow (and also of the minimum multicut to the maximum integral flow) is $\Omega(k)$, thus indicating that these upper bounds cannot be of any help in obtaining good approximation algorithms for this problem.

Although the restricted setting of trees may seem very simple at first, we show that it captures a surprisingly rich collection of problems. When restricted to trees of height 1 and unit edge capacities, minimum multicut is the same as minimum vertex cover for general graphs; if the capacities are arbitrary (and the height 1), it is equivalent to minimum weight vertex cover for graphs with weights on the vertices. The best approximation factor known for vertex cover is 2 for both the unweighted and the weighted case (see e.g., [2]) and has not been improved in a long time, indicating that improving our result would be quite difficult. Regarding maximum integral flow, we note that in the case of trees of height 1 and unit edge capacities, it is equivalent to maximum matching in general graphs; if the trees are of height 1 and edge capacities are arbitrary, it corresponds to maximum b-matching in general graphs. On the other hand, if the edge capacities are unity and the trees are of arbitrary height, integral flow corresponds to a generalization of matching, which we call *cross-free-cut matching*. This generalization inherits many nice combinatorial properties of matching. We give a polynomial-time algorithm for maximum integral flow in trees with unit edge capacities and hence for finding a maximum cross-free-cut matching.

Finally, we also show that the multicut problem in trees is equivalent to the set cover problem for a special class of set systems, which we call *tree-representable set systems*. Interestingly enough, the problem of recognizing this class in polynomial time has been extensively studied in a different context [29] (it is the same as testing if a given binary matroid is graphic), and efficient algorithms have been discovered [5]. Hence, we also get a 2-approximation algorithm for the tree-representable set cover problem. The best aproximation factor known for the set cover problem is $O(\log n)$, where n is the number of elements being covered and this is the best possible (modulo constant factors) unless $NP \subseteq TIME(n^{\log\log n})$ [23], [3].

2. Preliminaries.

Given a tree $T = (V, E)$, a capacity function $c\colon E \to \mathbf{Z}^+$, and k pairs of vertices (s_i, t_i), $1 \le i \le k$, we associate a commodity i with the pair (s_i, t_i) and designate s_i as the source and t_i as the sink for this commodity.

A *multicommodity flow* is a way of simultaneously routing commodities from their sources to the respective sinks while ensuring that the flow of each commodity is conserved at each vertex (except the source and sink vertex for that commodity) and that the sum of the flows of all commodities through an edge does not exceed the capacity of the edge. A multicommodity flow in which the sum of the flows over all the commodities is maximized is called a *maximum (multicommodity) flow*. A flow is *integral* if each commodity has an integral flow through each edge. The *maximum integral flow* problem is to find an integral multicommodity flow that is maximum.

A *multicut* is defined as a set of edges whose removal disconnects each source–sink pair. The capacity (weight) of a multicut is the sum of the capacities of the edges in it. The *minimum multicut* problem is to find a multicut of minimum weight.

The maximum multicommodity flow problem can be solved in polynomial time using linear programming. In the case of trees the linear program (LP) has a particularly simple structure. Let p_i denote the unique path from s_i to t_i in the tree, and let f_i be a variable for the flow along this path. Since the flow along any path is nonnegative, $f_i \geq 0$. Further, the total flow through an edge cannot exceed the capacity of the edge, i.e.,

$$\sum_{i:e\in p_i} f_i \leq c_e, \qquad e \in E.$$

The flow would be maximum when $\sum_{i=1}^{k} f_i$ is maximized. Hence the linear program for a maximum multicommodity flow is

$$\begin{array}{lrll} \text{maximize} & \displaystyle\sum_{i=1}^{k} f_i & & \\ \text{subject to} & \displaystyle\sum_{i:e\in p_i} f_i & \leq c_e, & e \in E, \\ & f_i & \geq 0, & 1 \leq i \leq k. \end{array}$$

The additional constraint, $f_i \in \mathbf{Z}^+$, yields a program for the maximum integral flow problem.

The dual of this linear program is

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{e\in E} d_e c_e & \\ \text{subject to} & \displaystyle\sum_{e\in p_i} d_e \geq 1, & 1 \leq i \leq k, \\ & d_e \geq 0, & e \in E, \end{array}$$

and can be viewed as an assignment of nonnegative distance labels, d_e, to edges $e \in E$, so as to minimize $\sum_{e\in E} d_e c_e$, subject to the constraint that each pair, (s_i, t_i), be at least a unit distance apart.

The optimal integral solution to the dual program is a 0/1 assignment of distance labels to the edges such that, for every commodity, the path in the tree corresponding to the commodity contains an edge with distance label 1. Thus the edges with $d_e = 1$ form a multicut of weight equal to $\sum_{e\in E} d_e c_e$. Conversely, the minimum multicut corresponds to an integral solution to the dual program of value equal to the weight of the multicut. Thus the optimal integral solution to the dual LP is the minimum multicut and hence the value of the optimal (fractional) solution, which by the Duality Theorem is equal to the maximum multicommodity flow, is a lower bound on the weight of the minimum multicut.

A similar approach to formulating the multiway cut problem (in general graphs) leads to an integer program whose linear-programming relaxation has an an optimal solution

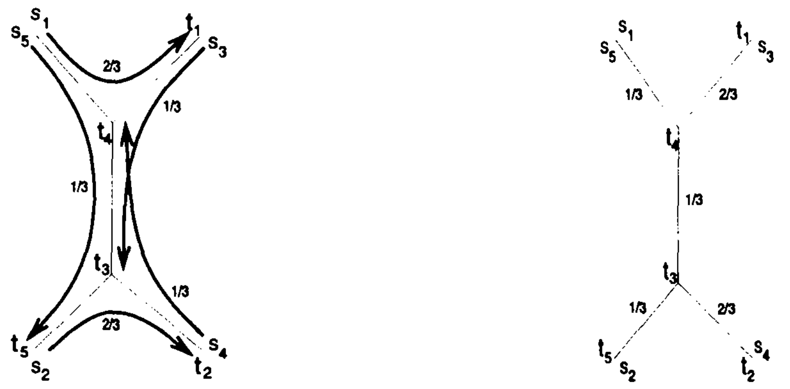

Fig. 1. Example to show that the minimum (fractional) multicut is not half-integral.

that is half-integral, i.e., every variable has value 0, $\frac{1}{2}$, or 1 [13]. This is not the case with the multicut problem, even for trees. All edges of the tree in Figure 1 have unit capacities. The figure shows a multicommodity flow of value $2\frac{1}{3}$ and a fractional multicut of the same weight. Therefore this is the pair of optimal solutions to the primal and dual linear programs.

3. Finding the Minimum Multicut.

The minimum multicut problem for trees can be solved in polynomial time for fixed k [31]. This is because the multicut contains at most k edges; one can in time $O(n^k)$ enumerate all subsets of edges of cardinality at most k and pick one that is a multicut and has the minimum weight. However, for arbitrary k the problem is NP-hard.

The *demand graph* H corresponding to a multicommodity flow or multicut instance is the graph whose vertices are the sources and sinks, and which contains an edge for each source–sink pair (s_i, t_i).

PROPOSITION 3.1. *For trees of height* 1 *and unit edge capacities, the minimum multicut problem is equivalent to the minimum vertex cover problem on general graphs. For trees of height* 1 *and arbitrary edge capacities, the minimum multicut problem is equivalent to the minimum weight vertex cover problem on general graphs.*

PROOF. Consider an instance of the multicut problem on a tree T of height 1 with root v and leaves $v_1, v_2, \ldots, v_d$. If the root v and a leaf v_i form a source–sink pair in the multicut instance, then clearly we must remove the edge (v, v_i). By removing all these forced edges we can assume without loss of generality that the given list of pairs in the multicut instance contains only leaves. Let the edge $e_i = (v, v_i)$ have capacity c_i. Consider the (weighted) vertex cover problem on the demand graph H, where the weight of a vertex v_i is c_i. If the edges $e_{i_1}, e_{i_2}, \ldots, e_{i_p}$ form a multicut in T, then the vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_p}$ form a vertex cover in the demand graph, H, and vice versa.

Thus, finding the minimum multicut in T is equivalent to finding a minimum weight vertex cover in H.

Conversely, given an instance of the (weighted) vertex cover problem on a graph H, we can construct a tree T of height 1 that has one leaf v_i for every vertex of H and one commodity for every every edge of H. $\square$

Since the vertex cover problem is NP-hard and MAX SNP-hard [25] we have:

THEOREM 3.1. *The minimum multicut problem is* NP-*hard and MAX* SNP-*hard even for trees of height* 1 *and unit capacities.*

4. Finding the Maximum Integral Flow.

The problem of finding the maximum integral flow is the same as that of finding a maximum cardinality set of edge-disjoint paths between the specified source–sink pairs. For the case when the demand graph is a complete subgraph, there exists under certain conditions (e.g., for Eulerian graphs) a min-max theorem relating the maximum number of edge-disjoint paths to the weight of the minimum multiway cut [21], [6], [24]. In this section we relate the problem of finding the maximum integral flow in a tree to other combinatorial optimization problems and establish its complexity.

4.1. *Unit Height Trees.* Let T be a unit height tree with root v and leaves $v_1, v_2, \ldots, v_d$. Let the edge $e_i = (v, v_i)$ have capacity c_i.

PROPOSITION 4.1. *For trees of height* 1 *and unit edge capacities, finding the maximum integral flow is equivalent to the maximum matching problem on general graphs.*

PROOF. Consider an instance of the maximum integral flow problem on a tree T of height 1 and unit edge capacities. If the root v and a leaf v_i form a source–sink pair, then we can route flow along the edge (v, v_i) without losing optimality. Thus, after routing these root–leaf flows we may assume, without any loss of generality, that the sources and sinks of all commodities are leaves of T. Consider the maximum matching problem on the demand graph H. Routing a unit of flow from v_i to v_j corresponds to picking edge (v_i, v_j) in H. Since all edges of T have unit capacity, an integral flow in T is a matching in H of the same size. The converse is also true; a matching in H of size f corresponds to an integral flow of f units in T. Thus computing the maximum integral flow in T is equivalent to finding a maximum matching in H.

Conversely, given an instance of the maximum matching problem on a graph H, we can construct a tree T of height 1 and unit edge capacities that has one leaf v_i for every vertex of H and one commodity for every edge of H. $\square$

A well-studied generalization of matching is the *b-matching* problem. Given a graph $G = (V, E)$ and a function $b\colon V \to \mathbf{Z}^+$, a b-matching is a set of edges, $E' \subseteq E$ with associated multiplicities $m\colon E' \to \mathbf{Z}^+$, such that each vertex, $v \in V$, has at most $b(v)$ edges incident at it (i.e., the sum of the multiplicities of the edges incident to v is at most $b(v)$).

PROPOSITION 4.2. *For trees of height 1 and unit edge capacities, finding the maximum integral flow is equivalent to the maximum b-matching problem on general graphs.*

PROOF. Given an instance of the multicut problem on a tree T of height 1, we can first saturate all edges (v, v_i) such that the root v and leaf v_i are the source and sink of a commodity, and hence we can assume, without any loss of generality, that the source–sink of each commodity is a leaf of T. Consider the b-matching problem on the demand graph H, where we let $b(v_i)$ be the capacity of the edge $e_i = (v, v_i)$. Sending f_{ij} units of flow between the source–sink pair (v_i, v_j) corresponds to picking the edge (v_i, v_j) with multiplicity f_{ij} in the b-matching of H. Thus, computing a maximum integral flow in T now corresponds to finding a maximum b-matching in H.

Conversely, given an instance (H, b) of the b-matching problem we can define an instance of the maximum integral flow problem on a tree of height one as in the proof of Proposition 4.1. The edge $e_i = (v, v_i)$ is now assigned a capacity $b(v_i)$. □

We remark that the variant of the b-matching problem, where edges can be picked only with multiplicity one (see e.g., [4]), or more generally there are upper bounds on the allowed multiplicities, can be translated to the b-matching problem as defined above (unbounded multiplicities); see e.g., p. 258 of [16].

Since the b-matching problem can be solved in polynomial time (see, for example, [11]), it follows that the maximum integral flow problem on trees of height 1 can be also solved in polynomial time.

4.2. *Trees with Unit Capacity Edges.* If S is a proper subset of the set of V of vertices, we use $\bar{S}$ to denote its complement $V - S$, and denote the partition of V into the two sets S and $\bar{S}$ and the corresponding cut by $(S, \bar{S})$. Following standard terminology, two cuts $(S, \bar{S})$ and $(Q, \bar{Q})$ are said to be *crossing* iff $S \cap Q$, $S \cap \bar{Q}$, $\bar{S} \cap Q$, and $\bar{S} \cap \bar{Q}$ are all nonempty. A family of cuts is *noncrossing* if no two cuts in the family are crossing.

Given a graph $G = (V, E)$ and a family, $\mathcal{F}$, of noncrossing cuts, define an $\mathcal{F}$-matching (a *cross-free-cut matching*) as a set of edges, $E' \subseteq E$, such that E' contains at most one edge from each cut in $\mathcal{F}$. If $\mathcal{F}$ is the set of all singleton cuts, $(v, V - v)$, $v \in V$, then a cross-free-cut matching is simply a matching in G (note that this family of cuts is noncrossing). Thus a cross-free-cut matching generalizes the notion of a matching. In general, a family $\mathcal{F}$ may not include some of the singleton cuts, in which case an $\mathcal{F}$-matching may have two or more edges incident to some nodes; that is, a cross-free-cut matching is not necessarily a matching, depending on the given family $\mathcal{F}$. The *maximum cross-free-cut matching* problem is to find a cross-free-cut matching of maximum cardinality for a given graph G and noncrossing family $\mathcal{F}$ of cuts.

We now extend Proposition 4.1 to trees of arbitrary height.

PROPOSITION 4.3. *The maximum integral flow problem on trees with unit capacity edges is equivalent to the maximum cross-free-cut matching problem.*

PROOF. Consider an instance of the maximum integral flow problem consisting of a tree T with unit capacities and a list of pairs of vertices (s_i, t_i). Define an instance of the maximum cross-free-cut matching problem on the demand graph H with a family of cuts $\mathcal{F}_T$ that includes one cut $(S_e, \bar{S}_e)$ for every edge e of T. Note that removing an edge e from the tree T separates T into two components; this induces a partition $(S_e, \bar{S}_e)$ of the vertices of H according to which component of $T - e$ they belong to. A feasible integral flow in T is a set of edge-disjoint source–sink paths, which by construction corresponds to a cross-free-cut matching in H, and vice versa.

Conversely, let H be a graph and let $\mathcal{F}$ be a family of noncrossing cuts. Clearly, if an edge of H does not belong to any of the cuts in $\mathcal{F}$ we can include it in the cross-free-cut matching. After removing these edges from H we may assume without loss of generality that every edge of the graph H belongs to some cut of $\mathcal{F}$.

We define an instance of the maximum integral flow problem on a tree T, so that there is a correspondence between integral flows in T and cross-free-cut matchings in H. Every vertex of H is assigned to a unique vertex of T (not necessarily one-to-one), and for every edge (u, v) of H there is one commodity whose source and sink are the vertices of T that are assigned u and v. There is a one-to-one correspondence between edges of T and cuts in $\mathcal{F}$. We define T by induction on the size of $\mathcal{F}$. If the family $\mathcal{F}$ is empty, then H has no edges, and we let T be the tree with one vertex, which is assigned all the vertices of H. If $\mathcal{F}$ is nonempty, let $(S, \bar{S})$ be a cut in $\mathcal{F}$. We define two families of noncrossing cuts, a family $\mathcal{F}_1$ on $S \cup \{a_1\}$, and a family $\mathcal{F}_2$ on $\bar{S} \cup \{a_2\}$, where a_1, a_2 are new elements, as follows. Since $\mathcal{F}$ is noncrossing, for every other cut $(Q, \bar{Q})$, one of the two sets in the partition, say Q, has an empty intersection with either S or $\bar{S}$. If $S \cap Q$ is empty, then we include $(Q, \bar{S} - Q \cup \{a_2\})$ in $\mathcal{F}_2$; if $\bar{S} \cap Q$ is empty, then we include $(Q, S - Q \cup \{a_1\})$ in $\mathcal{F}_1$. Construct inductively a tree T_1 for $\mathcal{F}_1$, which is assigned all the vertices in S and a_1, and a tree T_2 for $\mathcal{F}_2$, which is assigned all the vertices in $\bar{S}$ and a_2. Add an edge joining the vertex of T_1 that is assigned a_1 with the vertex of T_2 that is assigned a_2 to form the tree T.

An integral flow in T is a set of edge-disjoint source–sink paths and corresponds to a set of edges of H. It is easy to show inductively that every edge of T corresponds to a cut in $\mathcal{F}$, and that the partition of the vertices of H induced by removing an edge from T corresponds to a cut in $\mathcal{F}$. Hence an integral flow obeys the capacity constraints iff the corresponding set of edges of H is a cross-free-cut matching. □

THEOREM 4.1. *There is a polynomial-time algorithm for finding a maximum integral flow on trees with unit capacity edges, and hence for the maximum cross-free-cut matching problem.*

PROOF. Since all edge capacities are unity, and we want an integral flow, at most one commodity can flow through an edge. As shown in Proposition 4.1, for a tree of height 1, this is simply a maximum matching problem.

Our algorithm starts by rooting the tree at an arbitrary vertex. It then does two passes over the tree, level by level—an upward pass followed by a downward pass. Consider a tree of height 2 and let v be a vertex at level 1 and let T_v be the subtree rooted at v. Consider the commodities that have both their source and sink in T_v. As in Proposition 3.1, we can define a graph G_v whose vertices are the children of v, and solve a maximum matching problem on G_v to route the maximum flow in T_v. However, it may also be advantageous to route a commodity that exits T_v, i.e., that has one endpoint in T_v and the other outside.

Such a commodity must flow along the edge from v to the root, r, and hence we can only route at most one unit of one such commodity. This will be strictly advantageous only if we can still route the maximum amount of flow in T_v.

We determine the commodities for which we get a strict advantage as follows. Suppose that one endpoint of commodity, i, say s_i, is in T_v and the other outside. If $s_i = v$, then clearly we can still route the maximum flow within T_v. If s_i is a child of v, then it is advantageous to route commodity i only if there is a maximum matching in G_v in which s_i is free (unmatched). It is easy to compute these vertices once we have found a maximum matching M_v of G_v: a vertex s_i is free in some maximum matching if and only if there is an alternating path, with respect to M_v, from a free vertex to s_i. Once a maximum matching is found, it is well known that the vertices that are reachable from the free vertices by some alternating path can be computed in linear time. In this way we can compute those commodities exiting T_v, routing which might be strictly advantageous. Vertex v can now be considered the source or sink of these commodities. This is done for all vertices at level 1. Then a height 1 problem is solved at the root. In solving this, we pick the commodity, if any, that is routed on the (v, r) edge. Once this is done, the rest of the routing in the subtree rooted at v can now be fixed.

This is the essential idea of the algorithm for arbitrary height trees as well. In the upward pass we consider vertices level by level. At a vertex v we solve a height 1 problem using matching, and determine for each commodity exiting T_v whether routing it is strictly advantageous. Then we remove the children of v and consider the commodities giving strict advantage as originating at v itself.

In the downward pass we start at the root, fixing commodities. The vertex *parent*(v) decides which commodity gets routed on $(v, \textit{parent}(v))$. Once this is done vertex v fixes the commodities routed on the edges to its children. □

4.3. *Trees with Edge Capacities.* We generalize the notion of b-matchings by allowing constraints for any family of noncrossing cuts (not just singleton cuts) and call this a *cross-free-cut b-matching*. Thus, given a graph $G = (V, E)$, a family, $\mathcal{F}$, of noncrossing cuts, and a function $b\colon \mathcal{F} \to \mathbf{Z}^+$, a cross-free-cut b-matching is a set of edges, $E' \subseteq E$ with associated multiplicities $m\colon E' \to \mathbf{Z}^+$, such that the sum of the multiplicities of the edges in each cut $(S, \bar{S}) \in \mathcal{F}$ is at most $b((S, \bar{S}))$. The *maximum cross-free-cut b-matching* problem is to find a cross-free-cut b-matching of maximum cardinality.

A variant of the problem, in which edges can be picked only with multiplicity 1, or more generally each edge has an upper bound on its multiplicity, could be defined. We remark that this variant can be easily reduced to the problem as defined above. Define a new graph $G' = (V', E')$ which has one vertex u_e for each vertex u of G and edge e incident to u, and has an edge (u_e, v_e) for every edge $e = (u, v)$ of G. That is, G' is a perfect matching whose edges correspond to the edges of G. Define a family $\mathcal{F}'$ as follows. For every cut $(S, \bar{S}) \in \mathcal{F}$ we include in $\mathcal{F}'$ the corresponding cut $(S', \bar{S}')$ of G', where S' contains the nodes u_e for all nodes $u \in S$ and $b(S', \bar{S}') = b(S, \bar{S})$. In addition, $\mathcal{F}'$ includes the singleton cuts $(u_e, V' - u_e)$ with the value of the function b on these cuts equal to the allowed multiplicity for edge e. It is easy to see that $\mathcal{F}'$ is also a noncrossing family.

Along the same lines as Proposition 4.3 we can show:

PROPOSITION 4.4. *The maximum integral flow problem on trees with arbitrary edge capacities is equivalent to the maximum cross-free-cut b-matching problem.*

The maximum integral flow problem for trees with arbitrary edge capacities is NP-hard, and so is the maximum cross-free-cut b-matching problem. It is intriguing that generalizing the maximum matching problem to a family of noncrossing cuts results in a polynomial-time solvable problem (the maximum cross-free-cut matching problem), whereas the same generalization of the maximum b-matching problem results in an NP-hard problem.

THEOREM 4.2. *The maximum integral flow problem is* NP-*hard and MAX* SNP-*hard for trees with edge capacities* 1 *and* 2.

We reduce the NP-hard three-dimensional matching problem to the maximum integral flow problem. Given three disjoint sets X, Y, Z, $|X| = |Y| = |Z| = n$, and a set of triples $S = \{(x_i, y_j, z_k) | x_i \in X, y_j \in Y, z_k \in Z\}$, the three-dimensional matching problem is to find the maximum number of disjoint triples.

Given an instance of the three-dimensional matching problem, we construct a tree, T, of height 3. The vertices at level 1 correspond to the elements of $X \cup Y \cup Z$. A vertex corresponding to the element $x_i \in X$ has p_i children, where p_i is the number of occurrences of x_i in S. We label these vertices $\underline{x_i, l}$, $1 \le l \le p_i$. Each of the vertices $\underline{x_i, l}$ has two children labeled $\underline{x_i, l, a}$ and $\underline{x_i, l, b}$. Thus there are $|S|$ vertices at the second level and $2|S|$ vertices at the third level of the tree. Edges (r, x_i), $1 \le i \le n$, and $(x_i, \underline{x_i, l})$, $1 \le l \le p_i$, $1 \le i \le n$, have a capacity 2. All other edges have unit capacity. Figure 2 shows the construction of the tree.

The occurrences of x_i in S are numbered arbitrarily from 1 to p_i, and the lth occurrence corresponds to the vertex $\underline{x_i, l}$. If $(x_i, y_j, z_k) \in S$ is the lth occurrence of x_i, we add three source–sink pairs, $(\underline{x_i, l, a}, \underline{x_i, l, b})$, $(\underline{x_i, l, a}, y_j)$, and $(\underline{x_i, l, b}, z_k)$. Thus this instance of the multicommodity flow problem has $3|S|$ commodities in all. NP-hardness now follows from:

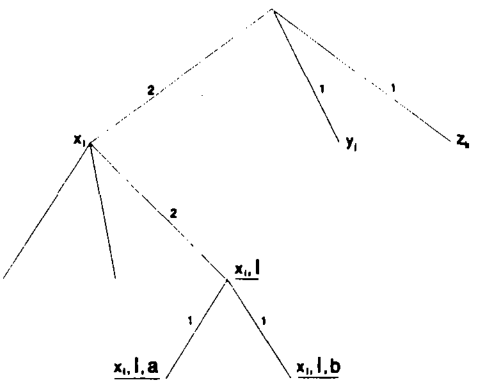

Fig. 2. The tree for the NP-hardness proof of maximum integral flow.

LEMMA 4.3. *The instance of the three-dimensional matching problem has t disjoint triples iff T has an integral flow of $t + |S|$ units.*

PROOF. Suppose that the instance of the three-dimensional matching problem has a set S' of t disjoint triples. Define an integral flow in T as follows. If an element x_i of X is not covered by S', then for all m, $1 \leq m \leq p_i$, we route a unit flow for the source–sink pair $(\underline{x_i, m, a}, \underline{x_i, m, b})$. If x_i is covered, and $(x_i, y_j, z_k) \in S'$ corresponds to the lth occurrence of x_i, then we route one unit of the commodities corresponding to the source–sink pairs $(\underline{x_i, l, a}, y_j)$ and $(\underline{x_i, l, b}, z_k)$; also, for all m such that $1 \leq m \leq p_i, m \neq l$, we route a unit flow for the source–sink pair $(\underline{x_i, m, a}, \underline{x_i, m, b})$. Thus, a covered x_i gives rise to $p_i + 1$ units of flow, and an uncovered x_i to p_i units. Therefore, the total flow routed is $t + \sum_{i=1}^{n} p_i = t + |S|$.

Conversely, assume there is an integral flow of $t + |S|$ units. Note that the maximum integral flow over the commodities that have at least one endpoint in the subtree rooted at x_i is $p_i + 1$. Moreover, this flow can be achieved only by routing one unit for the source–sink pairs $(\underline{x_i, l, a}, y_j)$ and $(\underline{x_i, l, b}, z_k)$, for some l, and one unit for each of the remaining $p_i - 1$ pairs $(\underline{x_i, m, a}, \underline{x_i, m, b})$, $1 \leq m \leq p_i, m \neq l$.

Since the integral flow has value $t + |S|$, there must be at least t elements x_i such that the flow routes $p_i + 1$ units over commodities with at least one endpoint in the subtree rooted at x_i. If flow is routed for the source–sink pairs $(\underline{x_i, l, a}, y_j)$ and $(\underline{x_i, l, b}, z_k)$, then (x_i, y_j, z_k) is a triple in S. Let S' be the set of these (at least t) triples. Because of the capacities of the edges (r, x_i), (r, y_j), and (r, z_k), each x_i, y_j, z_k is included in at most one of these triples. □

The MAX SNP-hardness of the maximum integral flow problem follows from the fact that the three-dimensional matching problem is MAX SNP-hard even if every element occurs a bounded number of times [18], and in this case the above transformation is an L-reduction [25]. Recall that an L-reduction from a problem A to a problem B is a polynomial-time tranformation f from instances of A to instances of B, such that, for some constants α, β, the following two conditions are satisfied:

1. $opt(f(I)) \leq \alpha \cdot opt(I)$, where $opt(I)$, $opt(f(I))$ are the optimal values of the instance I of A and $f(I)$ of B, respectively.
2. Given a solution y of $f(I)$ we can find in polynomial time a solution x of I, so that the values of x and y obey $|opt(I) - value(x)| \leq \beta |opt(f(I)) - value(y)|$.

If we consider the restriction of the three-dimensional matching problem to instances I where every element occurs at most d times, then the optimal value $opt(I)$ is at least $|S|/(3d - 2)$ because every triple intersects at most $3(d - 1)$ other triples. Thus the optimal value of the instance $f(I)$ of the maximum integral flow problem satisfies $opt(f(I)) = opt(I) + |S| \leq (3d - 1)opt(I)$, and thus condition (1) is satisfied with $\alpha = 3d - 1$. Furthermore, from a solution y to the flow instance with value $t + |S|$ we construct a solution x to the three-dimensional matching instance of size t, thus, $|opt(I) - value(x)| = |opt(f(I)) - value(y)|$ and condition (2) is satisfied with $\beta = 1$.

5. Approximating Integral Flow and Multicut. In this section we present an algorithm that finds a multicut, M, and an integral flow, F, such that the multicut is of weight at most twice the integral flow, i.e., $M \le 2F$. Since maximum multicommodity flow (and hence maximum integral flow) is a lower bound on the weight of the minimum multicut we have

$$M \le 2F \le 2 \cdot \text{maximum integral flow} \le 2 \cdot \text{weight of minimum multicut}$$

and

$$F \ge \tfrac{1}{2}M \ge \tfrac{1}{2} \cdot \text{weight of minimum multicut} \ge \tfrac{1}{2} \cdot \text{maximum integral flow.}$$

Our algorithm follows a primal- dual approach, the elements of which have been enunciated in [14] and [30]; see also [15] for a comprehensive exposition. This approach when applied to approximation algorithms consists of starting with arbitrary solutions to the primal and dual linear programs, and making alternate improvements to each, until "good" integral solutions to both are found. The improvements are guided by the complementary slackness conditions. The two complementary slackness conditions for our setting (recall the LPs from Section 2) are as follows:

1. $f_i > 0 \Rightarrow \sum_{e \in p_i} d_e = 1$, i.e., if the commodity i has a nonzero flow, then the sum of the distance labels along path p_i is exactly 1.
2. $d_e > 0 \Rightarrow \sum_{i:e \in p_i} f_i = c_e$, i.e., an edge with a positive distance label is saturated.

Enforcing both complementary slackness conditions would give us optimal solutions to the primal and dual linear programs. Since we are looking for good integral solutions to these programs and the optimal solutions are in general not integral, we cannot enforce all these complementary slackness conditions. We enforce the second complementary slackness condition and relax the first to

$$f_i > 0 \quad \Rightarrow \quad 1 \le \sum_{e \in p_i} d_e \le 2. \tag{1}$$

This implies that we pick only saturated edges in the multicut ($d_e > 0 \Rightarrow \sum_{i:e \in p_i} f_i = c_e$) and that, for any commodity that is routed, the flow path contains at most two edges of the multicut. It is easy to see that ensuring these two conditions would imply that the capacity of the multicut is at most twice the value of the flow.

We now describe an algorithm for finding a multicut and an integral flow (Figure 3) that meet these two requirements.

We begin by rooting the tree at an arbitrary vertex, say r. The level of a vertex is its distance from the root. A commodity is *contained in the subtree rooted at* v if the path corresponding to it lies completely within this subtree. A commodity is *contained in level* i if it is contained in a subtree rooted at some vertex in level i. An edge e_1 is an *ancestor* of an edge e_2 if e_1 lies on the path from e_2 to the root. The algorithm makes two passes over the tree.

PASS 1. In this pass we move up the tree, one level at a time, routing flow as we go along and picking some edges (a subset of these edges will be retained as the multicut). If v is a vertex in the current level, check if there exists a commodity contained in the

```
Algorithm multicut_integral-flow($r$);
    1. {Pass 1 }
       for current_level = max_level downto 0 do
           for all $v \in$ current_level do
               1.1. for all commodities contained in subtree rooted at $v$ do
                        Route as much flow of commodity as is possible. update $F$
               1.2. Compute frontier($v$)
    2. {Pass 2 }
      2.1. $M \leftarrow \phi$
           {Initializing multicut }
      2.2. for current_level = 0 to max_level do
               for all $v \in$ current_level do
                   for all $e \in$ frontier($v$) do
                       if $\nexists e' \in M$ such that $e'$ is on the path from $e$ to $v$ then
                         $M \leftarrow M \cup \{e\}$
    3. return $(M, F)$
end.
```

Fig. 3. Finding a multicut, M, and an integral flow, F, such that $M \leq 2F$.

subtree rooted at v. If yes, send as much flow of this commodity as is possible. Repeat this procedure until no more flow can be routed for commodities contained in this subtree.

We also need to pick a set of edges to include in the multicut. Let Q be the set of edges saturated in this step and let I be the set of commodities such that, for $i \in I$, the path p_i did not contain a saturated edge before this step but contains one now. Note that if a commodity in the set I is contained in the subtree rooted at v, then the path corresponding to it must use the vertex v. This is because we are moving up the tree and so would have considered all paths in this subtree that did not contain v at an earlier step in the procedure and saturated some edge along each of these paths. For this same reason, all paths along which flow is routed in this step use the vertex v. Thus if there are two edges in Q such that one is an ancestor of the other, then one of these edges is redundant as far as disconnecting the source–sink pair of commodities in I is concerned. We retain the edge that is the ancestor and denote this subset of Q as *frontier*(v) (the *frontier* of vertex v).

CLAIM 5.1. *The union of all frontiers is a multicut.*

PROOF. As a first step in proving this claim observe that the set of saturated edges is a multicut. If such is not the case, then there exists a commodity i such that no edge along p_i is saturated. We could hence have routed an additional flow of commodity i—a contradiction.

From our definition of frontiers it follows that the union of all frontiers disconnects exactly the source–sink pairs that are disconnected by the set of saturated edges. Hence the union of all frontiers is a multicut. □

We augment the flow in this manner, level by level. A path along which some flow is sent is called a *flow path*. Since all capacities are integral the flow along each flow path is also integral.

In the second pass we move down the tree dropping redundant edges from the multicut; this is done to ensure that no more than two edges of a flow path are included in the multicut (condition 1).

PASS 2. We move down the tree one level at a time and build the multicut. When considering vertex v we include an edge $e \in$ *frontier*(v) in the multicut only if no edge along the path from e to v is already included in the multicut. Let M be the set of edges picked.

CLAIM 5.2. *M is a multicut.*

PROOF. By our previous claim, the frontiers of all vertices put together form a multicut. Note that the edges of *frontier*(v) are required to disconnect a source–sink pair only if the path corresponding to this pair uses the vertex v (the set of commodities, I, in the preceding discussion). If e is an edge of *frontier*(v) not included in M, then there is another edge $e' \in M$ that lies on the path from e to v and hence disconnects all source–sink pairs that e was required to disconnect. □

We now show that the multicut picked, M, contains at most two edges of any flow path. We begin by making the following two claims.

CLAIM 5.3. *Let s_i–t_i be a flow path, and let v be the least common ancestor of s_i, t_i. If $e \in$ frontier(u) is an edge on this path, then u is an ancestor of v.*

PROOF. For contradiction, assume that v is an ancestor of u. Hence u is considered before v in the first pass. Since $e \in$ *frontier*(u), e is saturated while considering vertex u and flow is sent along the path s_i–t_i later (while processing vertex v). However, this is not possible as edge e lies on the path s_i–t_i. □

CLAIM 5.4. *If u is an ancestor of v, then no edge of frontier(v) is an ancestor of an edge of frontier(u).*

PROOF. Suppose $e_1 \in$ *frontier*(v) is an ancestor of $e_2 \in$ *frontier*(u). Since u is an ancestor of v, vertex v is considered before u in the first pass. As $e_1 \in$ *frontier*(v), e_1 is saturated while considering vertex v and $e_2 \in$ *frontier*(u) is saturated later (while processing vertex u). However, this is not possible as edge e_1 is an ancestor of e_2 and lies on the path from e_2 to u. □

LEMMA 5.1. *Let s_i–t_i be a flow path and let v be the least common ancestor of s_i, t_i. Then M contains at most one edge from the path s_i–v and one edge from the path t_i–v.*

PROOF. Let M contain two edges e_1, e_2 from the path s_i–v. Further, let $e_1 \in$ *frontier*(v_1) be an ancestor of $e_2 \in$ *frontier*(v_2). By the above two claims it follows that v_1 is an ancestor of v_2 which is an ancestor of v. Thus, e_1 occurs on the path from e_2 to v_2 and so while picking edges from *frontier*(v_2) (Pass 2) we would not have included e_2 in M. □

Hence the multicut, M, includes at most two edges of any flow path. Since all the edges in M are saturated, the two conditions that we set out to enforce are met. Hence the capacity of the multicut is at most twice the value of the flow.

6. Integrality Gap for Grid Graphs.

The approximation algorithm for the maximum integral flow on trees uses, implicitly, the fact that the ratio between the maximum fractional and integral flows for trees is at most 2. This, however, is not true for general graphs. In fact, even for grid graphs this gap is quite large.

PROPOSITION 6.1. *The gap between the maximum integral flow and the maximum fractional flow for grid graphs can be as high as $k/2$, where k is the number of commodities.*

PROOF. The graph G is a union of k paths, $p_1, p_2, \ldots, p_k$; the endpoints of the path p_i form the source–sink pair for commodity i. Every pair of paths, $p_i, p_j, i \neq j$, intersects in a unique edge.

The graph G can be embedded on a $k \times k$ grid. If the origin $(0,0)$ is the left bottom corner of the grid, then $s_i = (0, k-i+1)$ and $t_i = (i, 0)$. The path p_i is then the path from $(0, k-i+1)$ to $(i, k-i+1)$ to $(i, 0)$. Any two paths intersect at a unique vertex. To ensure that they intersect in a unique edge we replace each intersection $v \equiv (i, j)$ by two vertices v_a, v_b. The edges incident at vertices v_a, v_b are $(v_a, (i-1, j))$, $(v_a, (i, j+1))$, $(v_b, (i+1, j))$, $(v_b, (i, j-1))$, and (v_a, v_b). Figure 4 shows this embedding.

The maximum integral flow for this instance is one unit, as by routing any one commodity we block the paths of all other commodities. However, the maximum fractional flow is $k/2$ units; half a unit of each commodity can be routed simultaneously. This yields a gap of $k/2$ between the maximum fractional flow and the maximum integral flow. □

In the above example at least k edges are needed to disconnect every source–sink pair. Thus the gap between the maximum integral flow and the minimum multicut is k

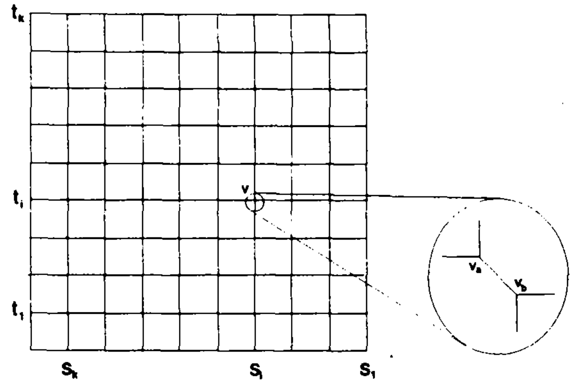

Fig. 4. The grid graph with the $k/2$ gap between the maximum (fractional) flow and the maximum integral flow.

and so we cannot use integral flow as a lower bound on the weight of the multicut when approximating the minimum multicut in general graphs.

7. The Tree-Representable Set Cover Problem.

The minimum multicut problem for trees can be viewed as a weighted set cover problem. The elements, P, of the set system, (P, E), are the s_i–t_i paths in the tree, $1 \le i \le k$. The sets, E, correspond to edges of the tree. A set includes all s_i–t_i paths (elements) that use this edge; the set has weight equal to the capacity of the edge. Finding the minimum weight multicut for the tree is the same as finding the minimum weight set cover in this set system.

Note, however, that not all set cover problems can be viewed as minimum multicut problems on trees. A set system (P, E) is called a *tree-representable set system* if there exists a tree T, whose set of edges is E such that every path $p \in P$ is a path in T. The problem of deciding whether a given set system is tree-representable is well studied and efficient algorithms are known [5]. Thus, given a set cover problem, it can be checked, in almost linear time, if it corresponds to the minimum multicut problem on some tree, T, and, if so, find T and the $\{s_i, t_i\}$ pairs.

THEOREM 7.1. *There exists a polynomial-time 2-approximation algorithm for the minimum weight set cover problem for set systems that are tree-representable.*

8. Discussion and Open Problems.

The notion of a cross-free family of cuts is quite basic and arises in several contexts (e.g., in approximating TSP by solving the LP relaxation, and doing subtour elimination). It will be interesting to interpret and apply our results to these contexts.

In particular, understanding better the link between our problem and the setting of [14] and [30] is especially interesting. The problem considered in [14] is to find a small subgraph that satisfies specified cut requirements. Let $G = (V, E)$ be a graph, and let $c: E \to \mathcal{Z}^+$ capacities on the edges and $f: 2^V \to \{0, 1\}$ be a function specifying cut requirements. For a set S of vertices, let $\nabla(S)$ denote the set of edges that have exactly one vertex in S and one outside S. The LP relaxation (primal) and dual of their problem is

$$\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{e\in E} x_e c_e & \\
\text{subject to} & \forall S \subset V, & \displaystyle\sum_{e:e\in\nabla(S)} x_e \ge f(S), \\
 & \forall e \in E, & x_e \ge 0.
\end{array}$$

$$\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{S\subset V} f(S) y_S & \\
\text{subject to} & \forall e \in E, & \displaystyle\sum_{S:e\in\nabla(S)} y_S \le c_e, \\
 & \forall S \subset V, & y_S \ge 0.
\end{array}$$

This primal and our primal form a complementary pair—this is a covering problem and ours is the corresponding packing problem. The above-stated dual is a cut packing problem. The approximation algorithm of [14] finds a dual solution in which the nonzero y_S's correspond to a noncrossing family of cuts. Furthermore, our algorithm, as well as those of [14] and [30] are primal–dual algorithms. These relationships seem to suggest a deeper connection. A general framework, in which both algorithms fit nicely, is presented in [15].

Is there a generalization of the maximum cross-free-cut matching problem, allowing certain crossing cuts, that is in P? We can show that allowing arbitrary crossing cuts leads to NP-hardness. Our reduction from the maximum independent set problem is approximation preserving and hence no polynomial-time algorithm can achieve an approximation factor of n^{ε}.

Can our primal–dual approximation algorithm be generalized beyond trees? Proposition 6.1 shows that any such algorithm attempting to approximate the maximum integral flow, will have to compare it with the nonintegral flow.

Acknowledgment. We wish to thank Clyde Momna and Alex Schaffer for providing references for the tree-representable set systems.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Proceedings, 33rd IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[2] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *J. Algorithms*, 2:198–203, 1981.

[3] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. *Proceedings, 25th Annual ACM Symposium on Theory of Computing*, pages 294–305, 1993.

[4] C. Berge. *Graph and Hypergraphs*. North-Holland, Amsterdam, 1976.

[5] R.E. Bixby and D.K. Wagner. An almost linear time algorithm for graph realization. *Math. Oper. Res.*, 13:99–123, 1988.

[6] B.V. Cherkasskij. Solution of a problem of multicommodity flows in a network (in Russian). *Mat. Metody*, 13:143–151, 1977.

[7] S. Chopra and M.R. Rao. On the multiway cut polyhedron. *Networks*, 21:51–89, 1991.

[8] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23:864–894, 1994. Preliminary version appeared under the title, The complexity of multiway cuts, *Proceedings, 24th Annual ACM Symposium on Theory of Computing*, pages 241–251, 1992.

[9] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5:691–703, 1976.

[10] A. Frank. Packing paths, circuits and cuts—a survey. In B. Korte, L. Lovasz, H.J. Promel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout*, pages 47–100. Algorithms and Combinatorics, volume 9. Springer-Verlag, Berlin, 1991.

[11] H.N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. *Proceedings, 15th Annual ACM Symposium on Theory of Computing*, pages 448–456, 1983.

[12] N. Garg, V.V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *Proceedings, 25th Annual ACM Symposium on Theory of Computing*, pages 698–707, 1993.

[13] N. Garg, V.V. Vazirani, and M. Yannakakis. Approximation algorithms for multiway cuts in node-

weighted and directed graphs. *Proceedings, 21st International Colloquium on Automata, Languages and Programming*, pages 487–498, 1994.

[14] M.X. Goemans and D.P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24:296–317, 1995. Preliminary version in *Proceedings, 3rd Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 307–316, 1992.

[15] M.X. Goemans and D.P. Williamson. The primal–dual method for approximation algorithms and its application to network design problems. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 144–191. PWS Publishing, Boston, 1995.

[16] M. Grotschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 1988.

[17] T.C. Hu. *Integer Programming and Network Flows*. Addison-Wesley, Reading, MA, 1969.

[18] V. Kann. On the approximability of NP-complete optimization problems. Ph.D. Thesis, Royal Institute of Technology, Stockholm, 1992.

[19] P. Klein, A. Agrawal, R. Ravi, and S. Rao. Approximation through multicommodity flow. *Proceedings 31st IEEE Symposium on Foundations of Computer Science*, pages 726–737, 1990.

[20] E. Korach and M. Penn. Tight integral duality gap in the Chinese postman problem. Technical Report, Computer Science Department, Israel Institute of Technology, Haifa, 1989.

[21] L. Lovász. On some connectivity properties of eulerian graphs. *Acta Math. Akad. Sci. Hungar.*, 28:129–138, 1976.

[22] F.T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with application to approximation algorithms. *Proceedings 29th Symposium on Foundations of Computer Science*, pages 422-431, 1988.

[23] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. Assoc. Comput. Mach.*, 41(5):960–981, 1994. Preliminary version appeared in *Proceedings 25th Annual ACM Symposium on Theory of Computing*, pages 286–293, 1993.

[24] W. Mader. Uber die maximalzahl kantendisjunkter a-wege. *Arch. Math.*, 30:325–336, 1978.

[25] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. Comput. System Sci.*, 43:425–440, 1991.

[26] N. Robertson and P.D. Seymour. Graph minors XIII: The disjoint path problem. *J. Combin. Theory Ser. B*, 63:65–110, 1995.

[27] A. Schrijver. Homotopic routing methods. In B. Korte, L. Lovász, H.J. Promel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout*, pages 329–371. Algorithms and Combinatorics, volume 9. Springer-Verlag, Berlin, 1991.

[28] A. Srivastav and P. Stangier. Integer multicommodity flows with reduced demands. *Proceedings European Symposium on Algorithms*, pages 360–372, 1993.

[29] W.T. Tutte. An algorithm for determining whether a given binary matroid is graphic. *Proc. Amer. Math. Soc.*, 11:905–917, 1960.

[30] D.P. Williamson, M.X. Goemans, M. Mihail, and V.V. Vazirani. A primal–dual approximation algorithm for generalized steiner network problems. *Proceedings, 25th Annual ACM Symposium on Theory of Computing*, pages 708–717, 1993.

[31] M. Yannakakis, P.C. Kanellakis, S.C. Cosmadakis, and C. H. Papadimitriou. Cutting and partitioning a graph after a fixed pattern. In *Automata, Languages and Programming*, pages 712–722. Lecture notes in Computer Science, volume 154. Springer-Verlag, Berlin, 1983.