Multiway Cuts in Directed and Node Weighted Graphs

Naveen Garg[*] *Vijay V. Vazirani*[*] *Mihalis Yannakakis*[†]

1 Introduction

Given an undirected graph $G = (V, E)$ with weights on the edges and a set of k terminals, $s_1, s_2 \ldots s_k$, a *multiway cut* is a set of edges whose removal disconnects every pair of terminals. This generalizes the fundamental notion of an s-t cut. Whereas the problem of computing a minimum s-t cut can be solved in polynomial time using a maximum flow algorithm, the problem of computing the minimum weight multiway cut was shown to be **NP**-hard and **max SNP**-hard even for fixed $k \geq 3$ by Dahlhaus, Johnson, Papadimitriou, Seymour and Yannakakis [2]. For trees, the minimum multiway cut can be found in polynomial time using dynamic programming [6]. The problem is also polynomial time solvable in planar graphs for fixed k; for general k it is **NP**-hard [2].

Dahlhaus et.al. [2] also give a $(2 - \frac{2}{k})$-approximation algorithm for the minimum multiway cut problem using the idea of *isolating cuts*. Cunningham [1] addresses the linear programming aspects of the multiway cut problem and describes lower bounds based on these. One such lower bound is the maximum multicommodity flow obtained by allowing flow between every pair of terminals; the objective is to maximize the sum of the flows. Cunningham shows that the minimum multiway cut is at most $(2 - \frac{2}{k})$ times the maximum flow.

Closely related to the notion of a multiway cut is that of a *k-cut*. A k-cut is a set of edges whose deletion disconnects the graph into k components. Goldschmidt and Hochbaum [3] show a polynomial time algorithm for computing the minimum k-cut, when k is fixed. For arbitrary k the problem is **NP**-hard and Saran and Vazirani [5] give a $(2 - \frac{2}{k})$-approximation algorithm for it. However, approximating the node version of the k-cut problem is **NP**-complete since identifying if the graph has a node k-cut involves checking if the graph has an independent set of cardinality k.

In this paper we consider node multiway cuts; the problem of computing a minimum weight node multiway cut is known to be **NP**-hard and **max SNP**-hard [1]. It turns out that the approximation algorithm in [2] for edge multiway cuts does not extend to the node multiway cut problem. Let us give a reason for this. Define an *isolating cut* for terminal s_i to be a cut that separates s_i from the rest of the terminals. A minimum isolating cut for s_i can be computed in polynomial time by identifying the remaining terminals, and finding a minimum cut separating them from s_i. The algorithm in [2] finds such cuts for each terminal, discards the heaviest cut, and picks the union of the remaining. The approximation factor is proven by observing that on doubling each edge in the optimum multiway cut, we can partition these edges into k isolating cuts, one for each

[*]Department of Computer Science and Engg., Indian Institute of Technology, New Delhi, India
[†]AT & T Bell Laboratories, Murray Hill, NJ 07974

terminal. Each of these cuts must be at least as heavy as the corresponding isolating cut found by the algorithm. Isolating cuts can be defined appropriately and found efficiently for the node setting as well. However, the second part of the argument crucially rests on the fact that each edge has two end points, and hence contributes to two isolating cuts. This does not carry over to the node case.

Interestingly enough, this problem does have an approximation algorithm with the same factor of $(2 - \frac{2}{k})$; this is the main result of this paper. We show that the **LP**-relaxation of an integer program for the minimum node multiway cut problem always has a half-integral optimal solution; this property is the basis of our algorithm. Moreover, we show how to obtain, in linear time, a half-integral optimal solution, given any optimal solution to the **LP**. A factor 2 approximation algorithm is then obvious – simply pick all nodes that are set to 1 or half; we show how to improve the guarantee to $(2 - \frac{2}{k})$. We also give a family of instances to show that the analysis of our algorithm is tight.

The dual of this linear programming relaxation is a multicommodity flow problem in which we have a commodity for every pair of terminals and the objective is to maximize the sum of the flows routed subject to flow conservation and throughput constraints on the non-terminal nodes. The maximum multicommodity flow is a lower bound on the weight of the minimum node multiway cut; however equality of maximum flow and minimum multiway cut does not hold in general. The best that one can do in such a situation is to establish an approximate max-flow min-multiway cut theorem. We prove such a theorem (Theorem 2.4) and show by means of an example that the bounds claimed by the theorem are tight.

Section 3 deals with multiway cuts in directed graphs. An edge or node multiway cut in a directed graph with a given set of k terminals is a set of edges or nodes whose removal gives a graph that has no path from any terminal to any other terminal. It is easy to see that in the case of directed graphs the edge and node versions of the problem are polynomially equivalent. We show that it is **NP**-hard and **max SNP**-hard to compute a minimum multiway cut in directed graphs even for 2 terminals (Theorem 3.1).

Moreover, in this case, the **LP**-relaxation may not have a half-integral optimal solution – we give an example showing this. As in the node case, the dual **LP** is a multicommodity flow problem and we give a polynomial time algorithm to find a multiway cut of weight at most $2 \log k$ times the maximum multicommodity flow. This gives us a $2 \log k$-approximation algorithm for the minimum multiway cut, as well as a max-flow min-multiway cut relation for directed graphs (Theorem 3.2). We also comment on two variants of directed multiway cuts: If the cut must also break every path going from a terminal to itself (i.e. a cycle containing a terminal) then the problem is solvable in polynomial time; if the cut has to break, for each pair of terminals, all cycles containing the pair, then the problem is hard for $k \geq 3$.

2 Node Weighted Graphs

Let $G = (V, E)$ be an undirected graph with node weights, $w : (V - T) \rightarrow \mathbf{R}^+$, where T is the set of k terminals, $s_1, s_2 \ldots s_k$. A *node multiway cut* is a set of non-terminal nodes whose deletion disconnects every pair of terminals. The graph has a node multiway cut only if the terminals form an independent set.

Cunningham [1] showed that the problem of computing the minimum node multiway cut is **NP**-hard. We give a different hardness proof by showing an approximation preserving reduction from the *vertex cover problem.*

Theorem 2.1 *An α-approximation algorithm for the minimum node multiway cut problem is also an α-approximation algorithm for the minimum vertex cover problem.*

Proof: Given a graph $G = (V, E)$ with node-weights, $w : V \to \mathbf{R}^+$, we construct another graph G' by adding a new node v' for each node $v \in V$ and the edge (v', v). The new nodes added are the set of terminals that we wish to diconnect. It is now easy to see that a node multiway cut in G' is a vertex cover in G and vice-versa. ■

2.1 A Multicommodity Flow Problem

With every pair of terminals s_i, s_j we associate a commodity and designate one of s_i, s_j as the source and the other as the sink for this commodity. Thus the total number of commodities is $\frac{k(k-1)}{2}$. The multicommodity flow problem is to maximize the sum of the flows of the commodities subject to flow conservation (a commodity is conserved at each node except the source and sink for that commodity) and throughput constraints (the sum of the flows of all commodities through a node cannot exceed the weight of the node).

The throughput and conservation constraints can be formulated as linear inequalities and the objective function, which is the sum of flows of all commodities is also linear. Thus, it is straightforward to formulate this multicommodity flow problem as a linear program and an optimal solution to it can be found in polynomial time.

Another way of writing a linear program for this problem, which although has exponentially many variables is interesting from the viewpoint of multiway cuts is as follows: Let p_i be a path between two distinct terminals and let f_i be a variable denoting the flow along this path. We need to ensure that the sum of the flows along paths that go through a node does not exceed the weight of the node, i.e.

$$\sum_{i:v \in p_i} f_i \leq w_v, \quad v \in V - T$$

Thus the linear program is

$$\begin{array}{lrcll} \text{maximize} & \sum_i f_i & & & \\ \text{subject to} & & & & \\ & \sum_{i:v\in p_i} f_i & \leq & w_v & v \in V - T \\ & f_i & \geq & 0 & \end{array}$$

and its dual is

$$\begin{array}{lrcll} \text{minimize} & \sum_{v\in V-T} d_v w_v & & & \\ \text{subject to} & & & & \\ & \sum_{v\in p_i} d_v & \geq & 1 & \\ & d_v & \geq & 0 & v \in V - T \end{array}$$

The dual can be viewed as an assignment of non-negative distance labels, d_v, to the non-terminal nodes such that the distance between any two terminals is at least 1; the objective is to minimize $\sum_{V-T} d_v w_v$. Note that we never need to assign any node a distance label more than 1. Hence, any integral solution to this **LP** is a 0,1 assignment of distance labels to the nodes such that each path, p_i, contains at least one node with $d_v = 1$. Thus, any integral solution to the dual **LP** corresponds to a node multiway cut in G which is of weight equal to the value of the solution. The converse is also true, i.e. every node multiway cut in G yields an integral solution to the dual **LP** of value equal to the weight of the multiway cut. This correspondence between the node multiway cuts in G and integral solutions to the dual linear program implies that the optimal solution to this linear program, which by the Duality theorem of linear programming is equal to the maximum multicommodity flow, is a lower bound on the value of the minimum node multiway cut.[1]

Figure 1: An example to show that maximum flow is not equal to the minimum node multiway cut.

Is maximum multicommodity flow equal to the minimum multiway cut? The example in Figure 1 shows that this is not the case. The graph in this example has, besides the k terminals, k nodes of unit weight and a center node of weight k. The maximum multicommodity flow routes a total flow of $\frac{k}{2}$ while at least $k-1$ non-terminal nodes are needed to form a multiway cut and hence the weight of the minimum node multiway cut is $k-1$.

The optimal solution to the dual **LP** for this example has distance labels of $\frac{1}{2}$ on each non-terminal node; each pair of terminals is a unit distance apart and the objective function has value $\frac{k}{2}$ which is equal to the maximum multicommodity flow. In the case of s-t cuts the dual of the maximum s-t flow **LP** has an optimal solution that is integral and hence the max-flow min-cut theorem follows as a simple consequence of the Duality theorem of linear programming. Since, in the case of multiway cuts the dual **LP** might not have an optimal solution that is integral, all that the Duality theorem can say is that the maximum multicommodity flow is equal to the minimum *fractional* multiway cut.

For multiway cuts, the example of Figure 1 is really the worst that things get, in the sense that the dual linear program always has an optimal solution that is half-integral.

[1] This also follows directly from the fact that the multiway cut acts as a bottleneck to flow and hence the value of the flow cannot exceed the capacity of any multiway cut.

2.2 Half-integrality of the Optimum

Consider the optimal solutions to the primal and dual linear programs. Let $d : (V - T) \rightarrow \mathbf{R}^+$ be the assignment of distance labels corresponding to this. By the Duality theorem, $\sum_{v \in V-T} d_v w_v$ is equal to the maximum multicommodity flow.

We define the *length* of a path to be the sum of the distance labels of the nodes along the path. The *distance* between two nodes is the length of the shortest path between them. Note that the terminals have no distance labels but for the purpose of computing distances we assign them a label 0. A *flow path* is a path between two distinct terminals such that there is a non-zero flow along this path. The two complementary slackness conditions are

1. A node that has a non-zero distance label must be saturated.
2. Any flow path must have length exactly one.

Any two terminals are at least a unit distance apart under the distance assignment d. We use this fact and the two complementary slackness conditions to show that there is an optimal solution to the dual **LP** that is half integral.

With each terminal s_i we associate a *region*, S_i, which is the set of nodes at a zero distance from s_i. The *boundary* of this region is the set of nodes adjacent to this region, and is denoted by $?(S_i)$. Since any two terminals are at least a unit distance apart, the regions $S_1, S_2 \ldots S_k$ are disjoint. Further, the boundary of a region does not have a node in common with any region.

The boundaries of the regions are not necessarily disjoint. If v is a node common to the boundary of two regions, S_i, S_j, then there is a path between terminals s_i and s_j on which v is the only node with a non-zero distance label; hence $d_v = 1$. Let $M = \cup_{i=1}^{k} ?(S_i)$ be the set of nodes in the boundaries and let M^1 be the nodes in M that belong to the boundaries of at least two regions. Then by the previous argument, $\forall v \in M^1 : d_v = 1$; the superscript in M^1 denotes the fact that all nodes in this set have a distance label 1. We next show that there is an optimal solution to the dual **LP** in which the remaining nodes of M, i.e. $M - M^1$, have a distance label $\frac{1}{2}$ (Theorem 2.3). We denote this set of nodes by $M^{1/2} = M - M^1$; every node in this set belongs to the boundary of exactly one region.

Lemma 2.2 *Any flow path from s_i to s_j either uses exactly one node from M^1 or it uses exactly two nodes from $M^{1/2}$.*

Proof: Let v_i, v_j be the first and last nodes from M on this flow path. Clearly, $v_i \in ?(S_i)$ and $v_j \in ?(S_j)$. We have two cases

1. $v_i = v_j$. Then $v_i = v_j \in M^1$. Further, this is the only node of M on this path, or else the length of this flow path would be strictly greater than 1, contradicting the second complementary slackness condition.
2. $v_i \neq v_j$. We first show that $v_i, v_j \in M^{1/2}$. Suppose $v_i \in M^1$. Then $d_{v_i} = 1$ and since $d_{v_j} > 0$ the length of this flow path is strictly greater than 1, contradicting the second complementary slackness condition. For the same reason $v_j \notin M^1$ and so $v_i, v_j \in M - M^1 = M^{1/2}$.

We now show that v_i, v_j are the only nodes from M on this flow path. Let $v_k \in ?(S_k)$ be another node from M on this path. Since $v_k \in ?(S_k)$, there is a path from terminal s_k to node v_k, all nodes on which - with the exception of v_k - have a distance label 0. This coupled with the facts that the length of the flow path $s_i - v_i - v_k - v_j - s_j$ is exactly one and $d_{v_i}, d_{v_j} > 0$ implies that the paths $s_i - v_i - v_k - s_k$ and $s_k - v_k - v_j - s_j$ have length strictly less than 1. We now have a pair of distinct terminals (s_i, s_k or s_j, s_k) the distance between which is less than one; a contradiction.

■

Theorem 2.3 *For any assignment of non-negative weights to the nodes of G there exists an optimal solution to the dual* **LP** *that is half integral. Moreover, given any optimal solution to the dual* **LP** *we can obtain, from it, a half-integral optimal solution in linear time.*

Proof: We start with an arbitrary optimal solution to the dual **LP**, and construct a feasible solution that is half-integral and has value equal to the maximum flow. Hence by the Duality theorem, this half-integral solution is an optimal solution to the dual **LP**.

Corresponding to the optimal solution to the dual **LP**, compute the sets $S_i, ?(S_i)$ for $1 \leq i \leq k$, and the sets $M, M^1, M^{1/2}$ as defined above. Since all nodes of M have a non-zero distance label, they are saturated (the first complementary slackness condition). This together with Lemma 2.2 implies that

$$\text{maximum flow } = W_{M^1} + \frac{1}{2} W_{M^{1/2}} \tag{1}$$

where W_S denotes the sum of the weights of the nodes in set S, i.e. $W_S = \sum_{v \in S} w_v$.

Now consider the following assignment of distance labels to the nodes

$$d^*_v = \begin{cases} 1 & \text{if } v \in M^1 \\ \frac{1}{2} & \text{if } v \in M^{1/2} \\ 0 & \text{otherwise} \end{cases}$$

Claim 2.1 *d^* is a feasible solution to the dual* **LP**.

Proof: Any path from s_i to s_j must use nodes from both $?(S_i)$ and $?(S_j)$. Let $v_i \in ?(S_i)$ and $v_j \in ?(S_j)$ be two nodes on this path. If $v_i = v_j$ then $v_i \in M^1$ and $d^*_{v_i} = 1$. Else if $v_i \neq v_j$ then $d^*_{v_i}, d^*_{v_j} \geq \frac{1}{2}$. In either case the length of this path is at least one. Hence d^* is dual feasible. ■

For this distance label assignment the value of the objective function is

$$\sum_{v \in V} d^*_v w_v = \sum_{v \in M^1} w_v + \frac{1}{2} \sum_{v \in M^{1/2}} w_v$$

But this is equal to the maximum flow (equation 1). Hence, d^* is an optimal solution to the dual **LP**. ■

2.3 Approximate Max-flow Min-multiway Cut Theorem

Theorem 2.4 (Approximate max-flow min-multiway cut theorem)

$$\textit{maximum flow} \leq \textit{minimum node multiway cut} \leq (2 - \tfrac{2}{k}) \cdot \textit{maximum flow}.$$

Proof: Consider a half-integral optimal solution to the dual **LP**. Since any path between two terminals contains at least one node from M, the set of nodes, M, forms a multiway cut. Further, since each node of M has a distance label that is at least $\frac{1}{2}$, the sum of the weights of these nodes is at most $2\sum_{v \in M} d_v w_v = 2\sum_{v \in V-T} d_v w_v$, which is twice the maximum flow.

We can prove a better bound on the weight of the multiway cut by discarding a part of the boundary of one of the regions. Recall that every node in $M^{1/2}$ belongs to the boundary of only one region. Let $?^{1/2}(S_i) = M^{1/2} \cap ?(S_i)$ be the set of nodes that are unique to the boundary of the region S_i. The sets $?^{1/2}(S_1), ?^{1/2}(S_2), \ldots ?^{1/2}(S_k)$ form a partition of the set $M^{1/2}$ and if $?^{1/2}(S_j)$ is the heaviest set in the partition then

$$W_{\Gamma^{1/2}(S_j)} \geq \frac{1}{k} W_{M^{1/2}}$$

Claim 2.2 *$M - ?^{1/2}(S_j)$ is a node multiway cut in G.*

Proof: Clearly, any path between two terminals different from s_j would use at least one node from $M - ?^{1/2}(S_j)$. Any path between terminals s_i and s_j uses a node of $?(S_i)$. Since $?(S_i)$ is disjoint from $?^{1/2}(S_j)$, such a path uses a node of $M - ?^{1/2}(S_j)$. Hence $M - ?^{1/2}(S_j)$ is a node multiway cut in G. ■

The weight of this node multiway cut can be bounded as follows.

$$\begin{aligned} W_M - W_{\Gamma^{1/2}(S_j)} &= W_{M^1} + W_{M^{1/2}} - W_{\Gamma^{1/2}(S_j)} \\ &\leq W_{M^1} + \left(1 - \frac{1}{k}\right) W_{M^{1/2}} \\ &= W_{M^1} + \frac{1}{2}(2 - \tfrac{2}{k}) W_{M^{1/2}} \\ &\leq (2 - \tfrac{2}{k}) \left(W_{M^1} + \frac{1}{2} W_{M^{1/2}}\right) \end{aligned}$$

Since d is an optimal solution to the dual **LP**, maximum flow equals

$$\sum_{v \in V-T} d_v w_v = W_{M^1} + \frac{1}{2} W_{M^{1/2}}$$

and hence the weight of the multiway cut is at most $(2 - \frac{2}{k})$ times the maximum flow. ■

The graph in Figure 1 also shows that this approximate max-flow min-multiway cut theorem is tight. The maximum multicommodity flow for this instance is $\frac{k}{2}$ while the minimum node multiway cut has weight $k - 1$. This gives a ratio of $(2 - \frac{2}{k})$ between the minimum node multiway cut and the maximum flow.

2.4 Approximating the Multiway Cut

The ideas in Theorem 2.4 can be used to obtain a $(2 - \frac{2}{k})$-approximation algorithm for the minimum node multiway cut problem. We do not know of any other algorithm for this problem that achieves an approximation guarantee better than the trivial k.

Theorem 2.5 *One can, in polynomial time, find a node multiway cut of weight at most* $(2 - \frac{2}{k})$ *times the weight of the minimum node multiway cut.*

Proof: We begin by finding an optimal solution to the dual **LP**. We now find the regions corresponding to the terminals and the boundaries of these regions. By choosing an appropriate set of nodes from these boundaries, as in Theorem 2.4, we get a multiway cut of weight at most $(2 - \frac{2}{k})$ times the maximum flow. Since the weight of the minimum node multiway cut is only larger than the value of the maximum flow, the node multiway cut found is of weight within $(2 - \frac{2}{k})$ times that of the minimum node multiway cut. ■

Algorithm Node_multiway_cut;

1. Compute an optimal solution to the dual **LP**

2. For each terminal s_i, compute S_i, $?(S_i)$ and $?^{1/2}(S_i)$

3. Let j be such that $?^{1/2}(S_j)$ has maximum weight

4. Output $\cup_{i=1}^{k} ?(S_i) - ?^{1/2}(S_j)$

end.

Figure 2: Algorithm for approximating the minimum node multiway cut

A slight modification to the example in Figure 1 shows that the analysis of our algorithm is tight. The nodes with unit weight are now assigned a weight 2, while the center node has weight $k + \epsilon$. The center node is also the minimum node multiway cut. The optimal solution to the dual **LP** assigns a distance label of 1/2 to each node of weight 2 and a zero label to the center node. Hence, the multiway cut found by our algorithm has weight $2k - 2$ which is about $(2 - \frac{2}{k})$ times the weight of the optimum.

3 Multiway Cuts in Directed Graphs

We are given a directed graph $G = (V, E)$ with edge weights, $w : E \rightarrow \mathbf{R}^+$, and a set of k terminals $S = \{s_1, s_2 \ldots s_k\}$. A (edge) multiway cut in G is a set of edges whose deletion separates every terminal from every other terminal; i.e., the remaining graph does not contain any path from s_i to s_j for $i \neq j$. The node version of the problem is defined analogously. It is easy to see that in the directed case the node and edge versions are polynomially equivalent, both in terms of finding optimal and approximate solutions. Furthermore, now the problems are hard even when we have just two terminals.

Theorem 3.1 *The problem of computing the minimum edge (node) multiway cut in directed graphs is* **NP***-hard and* **max SNP***-hard for all* $k \geq 2$*, even if all edge (node) weights are 1.*

Proof: We reduce the undirected 3-way (edge) cut problem to the directed 2-way cut problem. Let $G = (V, E)$ be an undirected graph with 3 distinguished terminals v_1, v_2, v_3. We replace every edge $e = [u, w]$ of G by a gadget consisting of two new nodes e_1, e_2, and arcs from u and w to e_1, from e_1 to e_2 and from e_2 to u and w; all these arcs have weight 1. We also add two new terminals s_1, s_2, and the following arcs: $s_1 \to v_1, s_1 \to v_2, v_1 \to s_1$, $v_3 \to s_2, s_2 \to v_2$; these arcs are given large weight, say $|E|$. Let D be the resulting directed graph. Clearly, an optimal 2-way cut C in D will not pick any of the heavy edges. Furthermore it is easy to see that if C picks an arc from the gadget corresponding to some edge e of G, then we can assume wlog that it picks the arc $e_1 \to e_2$. One can argue that for any set of edges Q of G, the set of arcs $\{e_1 \to e_2 \mid e \in Q\}$ of D separates s_1 and s_2 in D iff Q separates the terminals v_1, v_2, v_3 in G. Thus, an optimal 2-way cut in D corresponds to an optimal 3-way cut in G. We can get rid of the heavy edges, by replacing them with $|E|$ parallel paths of unit weight edges. ■

It is straightforward to achieve an approximation ratio of k by defining an isolating cut heuristic for the directed case. For each terminal s_i we can find a minimum cut that breaks all paths from s_i to the other terminals. The union of these cuts is a multiway cut of weight at most k times that of the minimum multiway cut. Furthermore, it is easy to construct examples showing that the factor k is essentially tight for this heuristic.

Unfortunately, the method of Section 2 based on the half-integrality of the **LP**-relaxation does not extend to the directed case either. We can again define a multicommodity flow problem by associating a commodity c_{ij} with each ordered pair of distinct terminals (s_i, s_j); terminal s_i is the source and s_j the sink for this commodity. The commodities are routed subject to conservation and capacity constraints with the objective of maximizing the sum of the flows of the commodities. This multicommodity flow problem when written as a linear program has a dual which can be viewed as an assignment of non-negative lengths to the edges, l_e, so that the length of any directed path between any two terminals is at least 1; the objective is to minimize $\sum_{e \in E} l_e w_e$. An integral solution to the dual **LP** is a multiway cut and hence the value of the maximum flow is a lower bound on the weight of the minimum multiway cut.

The optimal solution of the dual **LP** is not necessarily half-integral in this case. Consider for example the graph of Figure 3 with two terminals s_1, s_2, where the arcs $\alpha_1 \to \alpha_2, \beta_2 \to \beta_1, \gamma_2 \to \gamma_1, \delta_1 \to \delta_2$ have weight 1 and the rest of the edges have large weight. The maximum flow is 4/3, consisting of 2/3 units of flow in each direction; the dual **LP** has a unique optimal (fractional) solution that assigns length 1/3 to each of the four unit weight arcs and length 0 to the rest. The optimal (integral) 2-way cut in this case is 2.

3.1 Approximating Directed Multiway Cuts

We show that one can in polynomial time, find a multiway cut of weight $2 \log k$ times the maximum multicommodity flow, thereby giving an approximation algorithm for minimum multiway cut, as well as an approximate max-flow min-multiway cut theorem for directed graphs. Our algorithm is quite simple and uses a divide-and-conquer strategy.

Initially, a terminal can potentially be connected to all the remaining terminals. The algorithm proceeds in phases, and in each phase it halves the number of terminals that a given terminal can potentially be connected to. The algorithm thus terminates in $\log k$ phases. In each phase, the

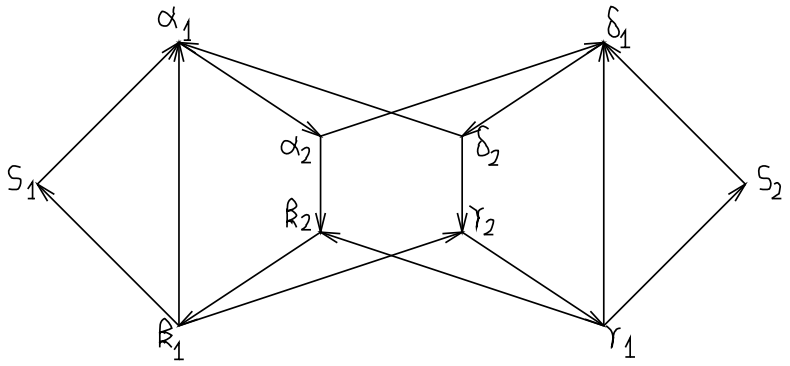

Figure 3: The optimal dual solution is not half-integral

algorithm removes edges having capacity at most twice the maximum multicommodity flow in the original graph. Hence, the capacity of edges removed is at most $2 \log k$ times maximum flow. Since maximum flow is a lower bound on the weight of the optimum multiway cut, we get the claimed bound.

It is no loss of generality to assume that the number of terminals is a power of 2. At the beginning of the i$^{\text{th}}$phase, the algorithm starts with a partition of the terminals into 2^{i-1} equal sized sets, and the edges removed so far ensure that terminals in different partitions are already disconnected from each other. In the i$^{\text{th}}$phase, the algorithm refines this partition by splitting each set into two equal sized subsets, (it names one *upper*, and the other *lower*), and removes edges to ensure that every terminal in the *lower* set is disconnected from every terminal in the *upper* set. This is done by identifying all terminals in the upper sets into a vertex u, and all terminals in the lower sets into a vertex v. and finding a minimum cut separating u from v, and another one separating v from u. The total weight of the edges removed is equal to the sum of the capacities of these two cuts which in turn is equal to the sum of the maximum flow from u to v and the maximum flow from v to u. However each of these maximum flows can be viewed as a multicommodity flow in the original graph. Hence, the edges removed in this phase have capacity at most twice the maximum multicommodity flow.

Theorem 3.2 (Approximate max-flow min-multiway cut theorem)

$$\textit{max-flow} \leq \textit{min multiway cut} \leq 2 \log k \ \textit{max-flow}.$$

Furthermore, a multiway cut of weight within $2 \log k$ *of the maximum multicommodity flow can be found in polynomial time.*

3.2 Variants of Directed Multiway Cuts

We now comment on two variants of the directed multiway cut. Consider the following slight modification of the definition of a directed multiway cut. We now want to disconnect every pair of terminals, whether distinct or not, i.e. we want to disconnect a terminal from itself as well. It is not hard to see then that the problem can now be solved in polynomial time. This was stated in [6] for fixed k, and the same algorithm works for arbitrary values of k.

Theorem 3.3 *A minimum weight directed multiway cut, where we want to disconnect every pair of terminals, not necessarily distinct, can be computed in polynomial time.*

Another interesting variant of directed multiway cuts is motivated by the *feedback arc set*[2]. The multiway cut is now defined as a set of edges whose removal breaks all such cycles which contain at least two distinct terminals. Equivalently, it is a set of edges whose removal ensures that for each pair of distinct terminals, s_i, s_j, either there is no path from s_i to s_j or from s_j to s_i. Note that when all the nodes of the graph are terminals this variant of multiway cut is the same as the feedback arc set and hence computing the minimum multiway cut is **NP**-hard in general. For $k = 2$ the minumum multiway cut can be found in polynomial time using two max-flow computations. Using a reduction similar to Theorem 3.1 for three or more terminals, we have:

Theorem 3.4 *The problem of computing the minimum multiway cut for k terminals is* **NP***-hard and* **max SNP***-hard for all $k \geq 3$ (even if all weights are 1).*

An $O(\log^2 k)$ approximation algorithm for this problem was recently given by [4].

4 Conclusions

We studied the multiway cut problem for undirected and directed graphs, in both its edge and node versions. The edge version for the undirected case was previously addressed in [2] where it was shown that the problem is **max SNP**-hard for three or more terminals, but can be approximated within a factor of $(2 - \frac{2}{k})$. We observed that the performance of this algorithm does not extend to the node version or to the directed case. We presented new approximation algorithms for these two cases. Both are based on a Linear Programming formulation, but use different methods otherwise. For the undirected node multiway cut problem, we achieve the same approximation ratio – $(2 - \frac{2}{k})$. For the directed (node or edge) multiway cut problem we achieve an approximation factor of $2 \log k$.

There remain a number of interesting open problems in this area. In the case of planar undirected graphs with a fixed number of terminals, the (edge) multiway cut problem can be solved in polynomial time [2]. We do not know whether this result can be generalized to the node version and to directed planar graphs.

For the directed multiway cut problem, we show via an algorithm that the ratio of the best integral and fractional solutions is bounded by $2 \log k$. Is this bound tight? If not, is there a better approximation algorithm for the problem?

References

[1] W. H. Cunningham. The optimal multiterminal cut problem. *DIMACS Series in Disc. Math. and Theor. Comput. Sci.*, 5:105–120, 1991.

[2] A *feedback edge set* in a directed graph is a set of edges such that the digraph formed by deleting this set is acyclic

[2] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. The complexity of multiway cuts. In *Proceedings, 24th Annual ACM Symposium on Theory of Computing*, 1992.

[3] O. Goldschmidt and D.S. Hochbaum. Polynomial algorithm for the k-cut problem. In *Proceedings, 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 444–451, 1988.

[4] P.N. Klein, S.A. Plotkin, S. Rao, and E. Tardos. Bounds on the max-flow min-cut ratio for directed multicommodity flows. Unpublished Manuscript, 1993.

[5] H. Saran and V.V. Vazirani. k-cuts within twice the optimal. In *Proceedings, 31st Annual IEEE Symposium on Foundations of Computer Science*, 1990.

[6] 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*, volume 154 of *Lecture notes in Computer Science*, pages 712–722, Berlin, 1983. Springer.