Min-Max Multiway Cut

Zoya Svitkina[⋆] and Éva Tardos[⋆⋆]

Cornell University, Department of Computer Science, Upson Hall, Ithaca, NY 14853.
{zoya, eva}@cs.cornell.edu

Abstract. We propose the Min-max multiway cut problem, a variant of the traditional Multiway cut problem, but with the goal of minimizing the maximum capacity (rather than the sum or average capacity) leaving a part of the partition. The problem is motivated by data partitioning in Peer-to-Peer networks. The min-max objective function forces the solution not to overload any given terminal, and hence may lead to better solution quality.
We prove that the Min-max multiway cut is NP-hard even in trees, or with only a constant number of terminals. Our main result is an $O(\log^3 n)$-approximation algorithm for general graphs, and an $O(\log^2 n)$-approximation for graphs excluding any fixed graph as a minor (e.g., planar graphs). We also give a $(2+\epsilon)$-approximation algorithm for the special case of graphs with bounded treewidth.

1 Introduction

The Min-max multiway cut problem is defined by an undirected graph $G = (V, E)$ with edge capacities $c(e) \geq 0$, and a set $X = \{x_1, ..., x_k\} \subseteq V$ of distinguished nodes called terminals. A *multiway cut* is a partition of V into disjoint sets $S_1, ..., S_k$ ($\bigcup_i S_i = V$), so that for all $i \in \{1, ..., k\}$, $x_i \in S_i$. For a partition we will use $\delta(S_i)$ to denote the capacity of the cut separating S_i from the other sets $\bigcup_{j \neq i} S_j$, and the goal of the min-max multiway cut problem is to minimize the maximum capacity $\max_i \delta(S_i)$.

The min-max multiway cut problem models the data placement problem in a distributed database system or a Peer-to-Peer system. In a Peer-to-Peer database, the information is stored on many servers. When a user query is issued, it is directed to the appropriate server. A request for some data item v can lead to further requests for other data. One important issue in such Peer-to-Peer databases is to find a good distribution of data that minimizes requests to any single server. We model this by a graph in which the non-terminal nodes represent the data items and the terminals represent the servers. Nodes in the partition S_i correspond to the data that will be stored on server i. Edges in the graph correspond to the expected communication patterns, i.e., the edge (x_i, v)

⋆ Supported in part by NSF grant CCR-032553.
⋆⋆ Supported in part by NSF grant CCR-032553, ITR grant 0311333, and ONR grant N00014-98-1-0589.

represents the number of queries that users at server i issue for the data v, and the edge (v, w) represents the expected number of times that a request for data v will result in an induced request for data w. Communication costs are incurred when a query from one server is sent to another. The goal then is to distribute the data among the servers so as to minimize the communication cost incurred by any one of them.

The min-max multiway cut problem is closely related to the traditional multiway cut problem of [2]. The difference is in the objective function. Unlike the min-max multiway cut, in which we seek to minimize the maximum capacity $\max_i \delta(S_i)$, the multiway cut problem evaluates a partition by the sum of the capacities of all edges that connect the parts, thus minimizing the average capacity $\delta(S_i)$. Multiway cut has been used to model similar applications of storing files on a network, as well as other problems such as partitioning circuit elements among different chips [2]. In many situations, however, the min-max objective function may be a better representation of the solution quality. Although the multiway cut minimizes the average communication cost of the terminals, this cost may not be distributed uniformly among them, resulting in a very heavy load on some terminals and almost no load on others. The objective of minimizing the maximum load tries to alleviate this problem by ensuring that no terminal is overloaded.

Multiway cut problem is NP-hard, but there are very good approximation algorithms for it [2, **?**, **?**]. However, they do not translate directly into good approximations for min-max multiway cut, because even the optimal solution to one problem can be up to a factor of $k/2$ worse than the optimum for the other.

Our results. For two terminals, min-max multiway cut reduces to the well-studied minimum s-t cut problem, and hence it can be solved in polynomial time. However, as we show, it is already NP-hard for the case of 4 terminals. As a result, we focus on designing approximation algorithms. In Section 2, we present an $O(\alpha \cdot \log n)$-approximation algorithm for min-max multiway cut, where $\alpha = \log^2 n$ for general graphs, and $\alpha = \log n$ for graphs excluding any fixed graph as a minor. The algorithm uses an α-approximation algorithm for a new graph cut problem, called Maximum size bounded capacity cut (MaxSBCC). We use it as a subroutine in a procedure that resembles the greedy set cover algorithm, incurring an additional factor of $O(\log n)$ in the approximation guarantee for the min-max multiway cut problem. One of the features of our algorithm is that it is able to exhibit flexibility when assigning graph nodes to terminals: if the cut that is found for one terminal is later discovered to be bad for another terminal, then the nodes are reassigned in a way that is good for both.

We extend our algorithm to a generalization of the problem, in which there is a separate bound B_i for each terminal x_i, and the goal is to find a partition in which $\delta(S_i)$ does not exceed B_i. This generalization is useful when the different peers corresponding to the terminals have different communication capabilities, and can withstand different loads.

Turning to special cases of min-max multiway cut, we show that it is NP-complete even on trees, and develop a $(2+\epsilon)$-approximation algorithm for trees and graphs with bounded treewidth. What makes the problem hard on trees is that an optimal solution does not necessarily assign connected components of the tree to each terminal (see figure 3, in which the black nodes are the terminals, and the optimal solution must assign the white node in the middle to one of the leaves). As a result, even if we know which edges should be cut, it may be hard to determine how to divide the resulting components among the terminals. The key idea of our $(2+\epsilon)$ approximation algorithm is to separate the stage of finding connected pieces of the graph from the stage of partitioning them among the terminals. Then, in the first stage, the problem of finding "good" pieces is solved optimally, and in the second stage these pieces are combined to form a 2-approximate solution. To make the dynamic programming algorithm of the first stage run in polynomial time, the edge capacities are rounded, leading to an overall $(2+\epsilon)$-approximation.

2 Min-Max Multiway Cut in General Graphs

2.1 Approximation Algorithm

Our main goal in this section is to provide an approximation algorithm for the min-max multiway cut problem and its extension with nonuniform bounds on the capacities.

First we briefly recall the 2-approximation algorithm of Dahlhaus et al. [2], as it is useful to understand why it does not work for the min-max version of the problem. Assume that there is a multiway cut with maximum capacity at most B. The algorithm finds a minimum capacity cut (S_i, T_i) separating each terminal x_i from all other terminals. It is not hard to see that the minimum cuts with smallest source sides S_i are disjoint. Let S_0 be the nodes not in any S_i, and let $\delta_i = \delta(S_i)$ be the capacity of the cut (S_i, T_i). The cut (S_i, T_i) is of minimum capacity, so we must have $\delta_i \leq B$. The algorithm of [2] assigns each set S_i to terminal x_i, and assigns the remaining nodes S_0 to one of the terminals (the one with maximum $\delta(S_i)$). This yields a 2-approximation (or, more precisely, a $2(1-1/k)$ approximation) algorithm for the multiway cut problem, but it is only a $(k-1)$-approximation for the min-max multiway cut problem, as the part $S_i \cup S_0$ can have capacity as high as $\sum_{j \neq i} \delta(S_j) \leq (k-1)B$.

The idea of our algorithm is to take cuts around each terminal that are larger in size than the minimum cut. Assume that we are given a bound B, and assume that there is a multiway cut where each side has capacity at most B. We will use a binary search scheme to optimize B. For a given value of B, we will need a subroutine for the following *maximum size bounded capacity cut* problem.

Definition 1. *Given a graph $G = (V, E)$ with two distinguished vertices s and t, weights on vertices $w(v)$, capacities on edges $c(e)$, and an integer B, the* Maximum size bounded capacity cut (MaxSBCC) *problem is to find an s-t cut (S, T) such that $\delta(S) \leq B$ and $w(S) = \sum_{v \in S} w(v)$ is maximized.*

The MaxSBCC problem can be shown to be NP-hard using a reduction from KNAPSACK. For $\alpha \geq 1$ and a constant $0 < \beta \leq 1$, let us define an (α, β)-approximation algorithm for MaxSBCC as an algorithm that, given an instance of MaxSBCC problem with a bound B and an (unknown) optimal solution (S^*, T^*), produces in polynomial time an s-t cut (S', T'), such that $\delta(S') \leq \alpha B$ and $w(S') \geq \beta w(S^*)$.

First we show how to use any such approximation algorithm as a subroutine for solving the min-max multiway cut problem, and later we give a specific $(log^2 n, 1)$ algorithm. The idea is analogous to the greedy $\log n$-approximation for the set-cover problem. Starting from the set V of unassigned nodes of the graph, our algorithm iteratively finds (approximate) maximum size bounded capacity cuts around each terminal, and temporarily assigns nodes to terminals, until no unassigned nodes remain. One important difference is that our algorithm is not greedy, in the sense that assignment made to terminals in one iteration can be revised in later iterations if that becomes useful. The full algorithm is shown in Figure 1.

1. Initialize $S_i = \{x_i\}$ for $i = 1, ..., k$, and initialize the weights $w(v)$ for all $v \in V$ by setting $w(x_i) = 0$ for all i, and $w(v) = 1$ for all other nodes.
2. While $\bigcup_i S_i \neq V$
   - For each terminal $x_i \in X$,
     (a) Construct a graph G' labeling x_i as source s and contracting all other terminals into a single sink t.
     (b) Find an (α, β)-approximate MaxSBCC (S, T) in graph G' with bound B and weights $w(v)$. Note that the set S does not have to contain S_i and does not have to be disjoint from the other sets S_j for $j \neq i$.
     (c) Consider the intersection $I_j = S \cap S_j$ for each $j \neq i$. We need to delete this intersection either from S_j or from S. If $c(I_j, S_j \setminus I_j) < c(I_j, S \setminus I_j)$, then let $S_j = S_j \setminus I_j$; otherwise let $S = S \setminus I_j$.
     (d) Let $S_i = S_i \cup S$, and set the weights of all $v \in S$ to $w(v) = 0$.
3. Return $S_1, ..., S_k$.

Fig. 1. Min-max multiway cut algorithm

Theorem 1. *If there is an (α, β)-approximation algorithm for MaxSBCC, then the above algorithm is an $O(\alpha \log_{1+\beta} n)$-approximation for the* MIN-MAX MULTIWAY CUT *problem.*

The key to the analysis is to see that each iteration assigns a constant fraction of the remaining nodes. By assumption there is a multiway cut $(S_1^*, ..., S_k^*)$ with capacity B. For each terminal x_i, we use the approximation bound to claim that the application of the MaxSBCC assigned at least as many new nodes to x_i as a β fraction of the remaining nodes in S_i^*.

Lemma 1. *If there is a multiway cut with maximum capacity at most B, then in any iteration of the while loop, if U is the set of unassigned nodes in the beginning of the iteration, and U' is the set of unassigned nodes at the end of this iteration, then $|U'| \leq \frac{1}{1+\beta}|U|$.*

Proof. Let N_i be the set of previously unassigned nodes added to the set S_i in this iteration. Notice that step (2c) of the algorithm only reassigns nodes with zero weight, so N_i has the same weight as the solution to MaxSBCC S obtained in step (2b).

Consider some optimal solution $(S_1^*, ..., S_k^*)$ to the min-max multiway cut instance. Now partition U' into sets $U_1', ..., U_k'$, such that $U_i' = S_i^* \cap U'$. We claim that $w(N_i) \geq \beta \cdot |U_i'|$. To see this, notice that the nodes in U' have weight 1 throughout this iteration of the `while` loop, and since S_i^* is a piece of the optimal partition, $\delta(S_i^*) \leq B$. Therefore, in the i^{th} iteration of the `for` loop, $(S_i^*, V \setminus S_i^*)$ is a feasible solution to the MaxSBCC problem, and $w(S_i^*) \geq w(U_i') = |U_i'|$. By the (α, β)-approximation guarantee, the algorithm for MaxSBCC must find a set with $w(N_i) \geq \beta \cdot |U_i'|$. Summing over all i, we obtain that $|U| - |U'| = \sum_{i=1}^{k} w(N_i) \geq \beta \cdot |U'|$, which proves the claim. □

Proof (of Theorem 1). By using binary search, we can assume that a bound B is given, and our algorithm will either prove that no multiway cut of maximum capacity at most B exists, or it will find a multiway cut with capacity at most $O(\alpha \log_{1+\beta} n)B$.

Throughout the algorithm, $x_i \in S_i$ for all i, and the sets S_i are always disjoint. So the algorithm finds a multiway cut, as required. By Lemma 1 the algorithm terminates in at most $\log_{1+\beta} n$ iterations of the `while` loop, if a min-max multiway cut of capacity at most B exists. If given an infeasible bound $B < B^*$, it may not stop after $\log_{1+\beta} n$ iterations, which proves that $B < B^*$. This shows that the algorithm runs in polynomial time. We will also use this bound to give an approximation guarantee for the algorithm.

We claim that for each S_i returned by the algorithm, $\delta(S_i) \leq \alpha \log_{1+\beta} n \cdot B$. To see this, notice that for each application of the MaxSBCC subroutine in (2b), the capacity of the set S returned is at most $\delta(S) \leq \alpha B$. By the choice made in step (2c), the transfer operation does not increase either $\delta(S_j)$ or $\delta(S)$. So in each iteration of the `while` loop, the capacity of each S_i increases by at most αB. Combined with the bound on the number of iterations, this observation concludes the proof. □

Feige and Krauthgamer [3] give an $O(\log^2 n)$ approximation algorithm for the problem of finding cuts with specified number of nodes, and an improved $O(\log n)$ approximation for the case when the input graph G is assumed not to contain a fixed graph as a minor (e.g., for planar graphs). We will use this algorithm to give an $(O(\log^2 n), 1)$ approximation algorithm for the MaxSBCC problem in general graphs and an improved $(O(\log n), 1)$ approximation algorithm in the special case. This will yield the following theorem.

Theorem 2. *There is an $O(\log^3 n)$-approximation algorithm for* Min-max multiway cut *problem on general graphs, and an $O(\log^2 n)$-approximation for graphs excluding any fixed graph as a minor (e.g., planar graphs).*

Proof. Feige and Krauthgamer [3] give an algorithm for finding cuts with specified sizes. For a graph G with n nodes and each number $d < n$, their algorithm finds a cut (S_d, T_d) with $|S_d| = d$ and capacity $\delta(S_d)$ within $\alpha = O(\log^2 n)$ of the minimum capacity for such a cut. For graphs excluding any fixed graph as a minor, this guarantee is improved to $\alpha' = O(\log n)$. The algorithm also works for finding s-t cuts on graphs with node weights and edge capacities.

We claim that the cut that corresponds to the largest value d^* such that $\delta(S_{d^*}) \leq \alpha B$ is an $(\alpha, 1)$-approximate MaxSBCC. By definition, its capacity is at most αB. And, if the optimal MaxSBCC had size $d' > d^*$, then, by the guarantee of the algorithm, $\delta(S_{d'})$ would be at most αB, contradicting our choice of d^*. The result then follows from Theorem 1. □

It is interesting to note that the algorithm can also be used for a version of the multiway cut problem in which there is a separate bound B_i for each $\delta(S_i)$. To obtain the extension, we use the MaxSBCC algorithm in each iteration i of the `for` loop with bound B_i rather than B.

Theorem 3. *Assume we are given a graph G with k terminals, edge capacities, and k bounds $(B_1, \ldots, B_k)$. If there is a multiway cut $(S_1, \ldots, S_k)$ such that $\delta(S_i) \leq B_i$ for each i, then in polynomial time we can find a multiway cut $(S'_1, \ldots, S'_k)$ such that $\delta(S'_i) \leq O(\log^3 n)B_i$, and the bound improves by a factor of $\log n$ for graphs excluding any fixed graph as a minor.*

Remark. Calinescu, Karloff and Rabani [1] and subsequently Karger et al. [6] gave improved approximation algorithms for the multiway cut problem based on linear programming and rounding. It appears that this technique does not yield a good approximation for our problem. To see this, consider the graph which is a star with k terminals and a single additional node at the center, and assume the capacity of each edge is 1. There is no multiway cut where each part has capacity at most $B = 2$, or even approximately 2. By assigning the center of the star to terminal x_i, we can get a multiway cut where the capacity of each part S_j for $j \neq i$ is 1, while the capacity of S_i is $k - 1$. A linear programming relaxation would allow us to take a "linear combination" of these cuts, and thereby have each side have capacity at most 2.

2.2 NP-completeness

We prove using a reduction from bisection that the Min-max multiway cut problem is NP-hard already on graphs with 4 terminals.

Theorem 4. Min-max multiway cut *is NP-hard for any fixed $k \geq 4$ even with unit-capacity edges.*

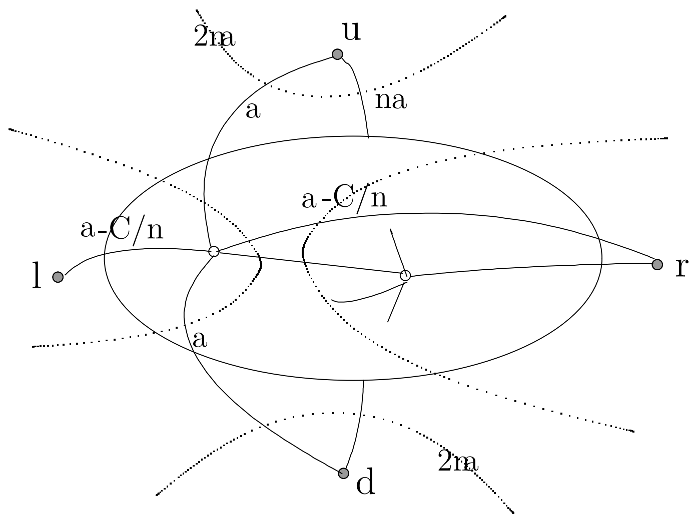

Fig. 2. Reduction from BISECTION to MIN-MAX MULTIWAY CUT.

Proof. We will show that it is NP-hard for $k = 4$ using a reduction from BISECTION [4]. Our construction uses capacities, but we can replace each edge with multiple parallel paths. An instance of BISECTION consists of a graph $G = (V, E)$ with an even number of vertices n, and an integer C. The question is whether or not there exists a partition of V into two sets X and Y, each of size $n/2$, such that the capacity of the cut $c(X, Y) \leq C$. Given G and C, we construct, in polynomial time, a graph F with 4 terminals and a bound B, so that F has a multiway cut with maximum capacity at most B if and only if G has a bisection with capacity at most C.

We obtain the graph $F = (V', E')$ by adding 4 new terminal nodes $X = \{u, d, l, r\}$ to G, and adding edges that connect nodes of G to the terminals (see Figure 2). E' includes E and the following additional edges, where a is chosen such that $2a > C$:

- Edge (u, d) of capacity na
- Edges (v, u) and (v, d), each of capacity a, for each $v \in V$.
- Edges (v, l) and (v, r), each of capacity $b = a - \frac{C}{n} > 0$, for each $v \in V$.

The bound is set to $B = 2na$.

Suppose F has a min-max multiway cut $(U \cup \{u\}, D \cup \{d\}, L \cup \{l\}, R \cup \{r\})$ where each part has capacity at most B. Then U and D must be empty, as $B = 2na \geq \delta(U \cup \{u\}) \geq 2na + 2b|U|$, just counting the edges to the terminals. So (L, R) is a cut of G, and let $C' = c(L, R)$ denote its capacity. The next observation is that $|L| = |R| = n/2$. To see this, suppose, for contradiction, that

$|L| = k \geq \frac{n}{2} + 1$ (or similarly for $|R|$). Then

$$\delta(L \cup \{l\}) = 2ka + nb + C' \geq 2(\frac{n}{2} + 1)a + n(a - \frac{C}{n}) = 2na + (2a - C) > B,$$

where the last inequality follows from the choice of a. We conclude that the capacity C' of the bisection (L, R) most be at most C. This follows as the capacity of the cut $L \cup \{l\}$ is $na + nb + C' \leq B = 2na$, and by the choice of b this inequality implies $C' \leq C$. To show the opposite direction, given a bisection (X, Y) in G of capacity $C' \leq C$, we produce a min-max multiway cut $(\{u\}, \{d\}, X \cup \{l\}, Y \cup \{r\})$ of F, in which each component has capacity at most B. □

3 Min-Max Multiway Cut on Trees and Bounded Treewidth Graphs

Recall from the Introduction that in an optimal solution to the min-max multiway cut problem on trees the sets of nodes assigned to the terminals do not have to be connected. This can be seen in the example of Figure 3. All nodes except for the middle one are terminals, and all edges have capacity 1. The optimal solution cuts all the edges incident on the middle node and assigns it to one of the leaf (degree-one) terminals, achieving a value of 4. On the other hand, any solution that assigns connected parts of the graph to each terminal would leave the middle node connected to one of its neighbors, incurring a cost of 5.

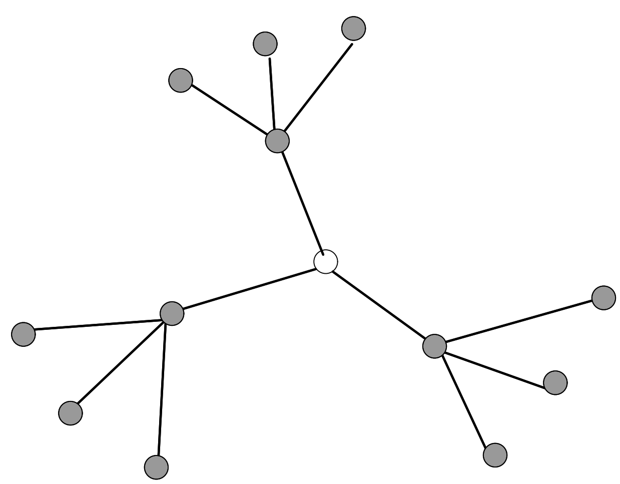

Fig. 3. Example showing that in an optimal min-max multiway cut on a tree, the sets assigned to the terminals need not form connected components. The only non-terminal in this graph is the middle node.

In section 3.1 we will use this observation to prove that the MIN-MAX MULTIWAY CUT problem in NP-hard in trees. Then we provide a $(2+\epsilon)$-approximation on trees, and, finally, in section 3.4 we extend this to graphs with bounded treewidth.

3.1 NP-completeness

Theorem 5. MIN-MAX MULTIWAY CUT *is strongly NP-hard when the graph is a tree with weighted edges.*

Proof. We use a reduction from 3-PARTITION, which is known to be strongly NP-complete [5]. In 3-PARTITION, given a set $A = \{a_1, ..., a_{3m}\}$, a weight w_i for each $a_i \in A$, and a bound B, such that $\forall i\ B/4 < w_i < B/2$ and $\sum_{i=1}^{3m} w_i = mB$, we want to know if A can be partitioned into disjoint sets $S_1, ..., S_m$, such that for each j,

$$\sum_{a_i \in S_j} w_i = B.$$

Given an instance (A, B) of 3-partition, we construct an instance of min-max multiway cut as follows. The tree G consists of separate subtrees connected with zero-capacity edges. There will be $3m$ subtrees T_i, one for each element a_i, and m isolated terminals $x_1, ..., x_m$, one for each of the desired sets. Each T_i consists of six terminals and one non-terminal v_i, with edge capacities as shown in Figure 4.

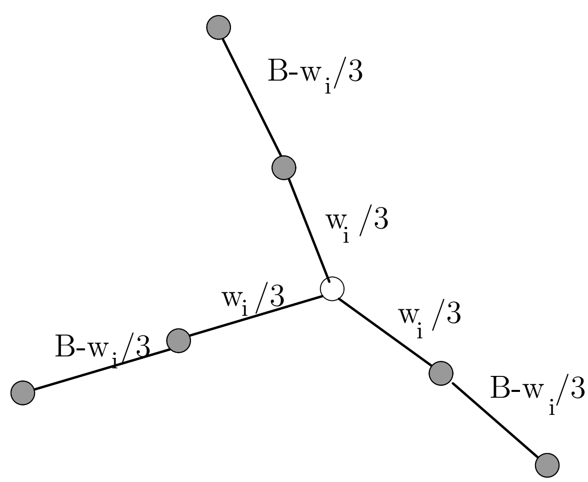

Fig. 4. Component T_i used in the NP-completeness reduction for MIN-MAX MULTIWAY CUT on trees.

We claim that a min-max multiway cut of maximum capacity at most B exists if and only if the 3-partition instance is solvable. Notice that any min-max multiway cut with capacity at most B must cut all edges of T_i and assign all v_i's to the terminals $x_1, ..., x_m$, creating a partition of A. If a set of nodes S'_j is assigned to terminal x_j, then the capacity of the resulting part is $\sum_{v_i \in S'_j} w_i$. This implies that such a cut exists if and only if the 3-partition does. □

3.2 Algorithm for Min-Max Multiway Cut on Trees

In this section we give a $(2+\epsilon)$-approximation algorithm for the min-max multiway cut on trees that have edge capacities.

The algorithm consists of two stages. In the first stage we consider a variant of the problem where we allow the algorithm to create extra parts in the partition that do not contain terminals. More precisely, we consider the following problem.

Definition 2. *The* Tree cutting *problem* (T, X, B) *for a tree* T*, terminals* $X = \{x_1, ..., x_k\} \subseteq V$*, and a bound* B *is to find a partition of* V *into connected subtrees* $T_1, ..., T_h$*, subject to the following constraints: (1) no two terminals are in the same connected component; and (2) for each connected component* T_i*,* $\delta(T_i) \leq B$*. The objective is to minimize* $\sum_i \delta(T_i)$*.*

In the next subsection we give a pseudo-polynomial time algorithm for this problem. Here we show how to use such an algorithm for a $(2+\epsilon)$-approximation for the min-max multiway cut on trees.

Theorem 6. *Using a pseudo-polynomial time algorithm for the* Tree cutting *problem as a subroutine, we can give a polynomial time* $(2+\epsilon)$*-approximation for the* Min-max multiway cut *on trees.*

Proof. As before, we use binary search to find the optimal value B for the problem. Assume first that we have an optimal solution to the tree cutting problem (T, X, B), and we will now describe how to use such a solution to find a 2-approximate solution for the min-max multiway cut problem.

If there exists a solution to the min-max multiway cut instance on tree T with terminals X and maximum capacity B, then the connected components in it give a feasible solution for the tree cutting problem with the objective function at most kB. Since we are solving the tree cutting problem optimally, if the solution $T_1, ..., T_h$ that we obtain does not satisfy $\sum_{i=1}^{h} \delta(T_i) \leq kB$ then $B < B^*$. The algorithm will group the components T_i into k sets $S_1, ..., S_k$ of nodes that will serve as our 2-approximate solution to the min-max multiway cut problem. For each i, start with $S_i = T_l$ such that $x_i \in T_l$, and, while $\delta(S_i) < B$ and there is an unassigned component T_j not containing a terminal, let $S_i = S_i \cup T_j$. Observe that if several components, say $T_1, ..., T_j$, are combined into a set S, then $\delta(S) \leq \sum_{i=1}^{j} \delta(T_i)$. Because $\sum_{i=1}^{h} \delta(T_i) \leq kB$, and each $\delta(S_i)$ is at least B, all components will be packed into the k sets. Also, since for all j, $\delta(T_j) \leq B$, it is clear that for no i will $\delta(S_i)$ exceed $2B$.

Recall that our tree cutting algorithm runs in pseudopolynomial time. We obtain a polynomial-time $(2+\epsilon)$-approximation algorithm via rounding. For a given ϵ, capacity bound B, and $m = |E|$, let

$$\alpha = \frac{\lceil m/\epsilon \rceil}{B}.$$

For each edge $e \in E$, scale the capacity so that $c'(e) = \lfloor \alpha c(e) \rfloor$. Also set $B' = \alpha B = \lceil m/\epsilon \rceil$. If there is a multiway cut with maximum capacity B in the original problem, then there is one of maximum capacity at most B' after rounding. Now we obtain a 2-approximate multiway cut $S_1, ..., S_k$ for the graph with capacities $c'(e)$ and bound B'. The running time is polynomial in $B' = \lceil m/\epsilon \rceil$ and n. The capacity of the parts S_i of this partition is at most $\delta(S_i) \leq (2+\epsilon) \cdot B$ for the original capacities. □

3.3 Algorithm for the Tree Cutting Problem

We now describe an algorithm that solves optimally, in time polynomial in B and the size of the tree n, the tree cutting problem (which we used as a subroutine for the min-max multiway cut on trees). To simplify the presentation of the algorithm, assume, without loss of generality, that (1) T is rooted at a node r and all edges are directed away from the root; (2) T is binary. (To make the tree binary without affecting the solution, replace each node u that has $d > 2$ children with a $\lceil \log_2 d - 1 \rceil$-height complete binary subtree U with edge capacities $B+1$, and attach u's children to the leaves of U, at most 2 per leaf.)

The tree cutting problem will be solved using dynamic programming. First consider the simpler problem with no terminals. To solve this problem, we construct a dynamic programming table $p(v, A)$ for all nodes $v \in V$ and integers $0 \leq A \leq B$, where the entry $p(v, A)$ is the minimum total capacity of edges in the subtree of T rooted at v that can be cut such that the total capacity of edges coming *out* (i.e., toward descendants) of v's component is at most A. We have the separate bound A because the remaining $B - A$ capacity will be used to cut the edges that are incident on v's component, but lie outside of its subtree. The values $p(v, A)$ can be computed in a single pass up the tree. If a node v has one child v_1 then cutting (v, v_1) implies that the component containing v_1 can have at most $B - c(v, v_1)$ capacity below v_1, so the total capacity obtained this way is $c(v, v_1) + p(v_1, B - c(v, v_1))$. If we do not cut the edge (v, v_1), then we get $p(v_1, A)$. This leads to the following recurrence for the case that v has a single child v_1.

$$p(v, A) = \begin{cases} p(v_1, A) \text{ if } c(v, v_1) > A \\ \min\{p(v_1, A), c(v, v_1) + p(v_1, B - c(v, v_1))\} \text{ otherwise.} \end{cases}$$

Now suppose that the internal node v has two children, v_1 and v_2. Then the capacity A available for cutting edges below v has to be partitioned between the edges that belong to the left subtree (including, possibly, (v, v_1)), and the ones that belong to the right subtree (possibly including (v, v_2)). The algorithm tries all possibilities for such a partition $A_1 + A_2 = A$. Then, given A_i, it decides independently for each child node v_i whether or not to cut (v, v_i), using an expression similar to the one above.

Next we extend the algorithm to make sure that all terminals in the tree are separated. For this, a binary variable t is added to the parameters of the table. The value of t limits the options available to the above simpler algorithm in each step. It will either require that a given component *not* contain a terminal (t=0), or it will not impose such a restriction ($t = *$). The idea is that if a node is connected to some ancestor which is a terminal, then it may not be connected to any descendants which are terminals, so in this case we will use a table entry with $t = 0$. Also, care has to be taken that a node is not connected to two terminals which are both descendants.

Theorem 7. *The optimal solution to the* TREE CUTTING *problem can be computed in time polynomial in the size of the graph and the bound B.*

3.4 Bounded Treewidth Graphs

Finally, we extend the $(2 + \epsilon)$-approximation algorithm for min-max multiway cut to work on graphs with bounded treewidth (see [7] for an introduction to tree decomposition). The only change is in solving the tree cutting problem.

First we note that the algorithm from section 3.3 can work on trees with degree greater than two. The only potential difficulty is to generate the optimal guesses of $A_1, ..., A_d$, $\sum A_i = A$, in polynomial time. Given the values of the subproblems at the leaves, the problem of finding the best partition to obtain the value $p(v, A)$ is essentially a knapsack problem, and hence solved by optimally solving the problem for each suffix $j, ..., d$ of the set of node's children, with j going from d to 1, and using the optimal result for $j, ..., d$ in order to solve the problem for $j - 1, ..., d$.

Now we sketch how to extend this algorithm to handle graphs with bounded treewidth. Suppose that we are given a graph $G = (V, E)$ and a decomposition tree T for it, such that for each node u in T, there is a set $V_u \subseteq V$ associated with it. The size of each V_u is bounded by a constant b. Let us root T at some node r and assign a height $h(u)$ to each node $u \in T$ so that r is the highest. Now we can associate with each vertex $v \in V$ a *label*, which is defined as $\max\{h(u)|v \in V_u\}$. The algorithm will again build a dynamic programming table. In this case a table entry will be $p(u, \{H_1, ..., H_h\}, \{A_1, ..., A_h\}, \{t_1, ..., t_h\})$, where u is a node in T; H_i's form a partition of V_u that has h components, for some $h \in [1, ..., b]$, with the meaning that different H_i's will be subsets of different components in the solution; A_i is a bound on the total capacity of edges (v, w) such that v is in the same component as H_i, w is not, and $label(v) > label(w)$; and t_i is a variable for H_i that specifies, as before, whether this component is allowed to contain terminals. The computation proceeds in a bottom-up fashion on the tree T. When a node u of T and its child node u_1 are considered, all allowed combinations of vertices in V_u and V_{u_1} are evaluated, subject to the constraints imposed by H_i's and t_i's, and the best one is chosen. If u has multiple children, then each A_i is divided among the children in the same way as before. The running time of the algorithm is exponential in b, but, given that b is a fixed constant, it remains polynomial (or pseudopolynomial, as before).

Theorem 8. *The above algorithm computes the optimal solution to the* TREE CUTTING *problem in graphs with bounded treewidth in time polynomial in the size of the graph and the bound B. As a consequence, we get a $(2+\epsilon)$-approximation for the* MIN-MAX MULTIWAY CUT *in graphs with bounded treewidth.*

Acknowledgements

We would like to thank Johannes Gehrke and Ashwin Machanavajjhala for suggesting the problem to us and for many insightful conversations. The maximum size bounded capacity cut problem (or rather its minimization version) was formulated as joint work with Ara Hayrapetyan, David Kempe, and Martin Pál. The reduction from KNAPSACK is due to David Kempe.

References

1. G. Calinescu, H. Karloff, and Y. Rabani. *An improved approximation algorithm for multiway cut*. STOC 1998.
2. E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D.Seymour, and M. Yannakakis. *The complexity of multiterminal cuts*. SIAM Journal on Computing, 23(4):864-894, 1994.
3. U. Feige and R. Krauthgamer. *A polylogarithmic approximation of the minimum bisection*. SIAM Journal on Computing, 31(4):1090-1118, 2002.
4. M. R. Garey, D. S. Johnson, and L. Stockmeyer. *Some Simplified NP-Complete Graph Problems*. Theoretical Computer Science, (1):237–267, 1976.
5. M. R. Garey and D. S. Johnson 1979. *Computers and Intractability*. W. H. Freeman and Company, New York.
6. D. R. Karger, P. Klein, C. Stein, M. Thorup, N. E. Young. *Rounding algorithms for a geometric embedding of minimum multiway cut*. STOC 1999.
7. J. Kleinberg and E. Tardos. *Algorithms Design*. Addison-Wesley, to appear.