Approximating the k-Multicut Problem

Daniel Golovin* Viswanath Nagarajan† Mohit Singh‡

Abstract

We study the k-multicut problem: Given an edge-weighted undirected graph, a set of l pairs of vertices, and a target $k \leq l$, find the minimum cost set of edges whose removal disconnects *at least* k pairs. This generalizes the well known multicut problem, where $k = l$. We show that the k-multicut problem on trees can be approximated within a factor of $\frac{8}{3} + \epsilon$, for any fixed $\epsilon > 0$, and within $O(\log^2 n \log\log n)$ on general graphs, where n is the number of vertices in the graph.

For any fixed $\epsilon > 0$, we also obtain a polynomial time algorithm for k-multicut on trees which returns a solution of cost at most $(2 + \epsilon) \cdot OPT$, that separates at least $(1 - \epsilon) \cdot k$ pairs, where OPT is the cost of the optimal solution separating k pairs.

Our techniques also give a simple 2-approximation algorithm for the multicut problem on trees using total unimodularity, matching the best known algorithm [8].

1 Introduction

Graph cut problems are widely studied in the area of approximation algorithms. The most basic cut problem is the s-t minimum cut problem, for which Ford and Fulkerson gave an exact algorithm [3]. Since then, many different cut problems have been considered, some examples are [7, 8, 15]. Many of these cut problems are useful sub-routines in solving other problems [22]. The study of these cut problems has also given rise to many new and interesting techniques.

In a multicut problem, we are a given a graph $G = (V, E)$, a cost function on the edges, and a set of pairs $(s_i, t_i) : 1 \leq i \leq l$. The objective is to select a minimum cost set of edges which separates all the pairs. Garg, Vazirani and Yannakakis [7, 8] gave an approximation algorithm for the multicut problem.

In this paper, we study a generalization of the multicut problem, the k-multicut problem. Here, in addition to the set of pairs, we are given a target $k \leq l$, and the goal is to select a minimum cost set of edges which separates at least k pairs.

Similar generalizations for other optimization problems like Set Cover [5], and MST [6] have been studied before, where the objective is not to satisfy all of the constraints, but rather to satisfy only a specified number of them. These problems are motivated by the presence of outliers in real life data; one may want to find a solution not affected by these outliers. One way to model this is to ask for a solution that satisfies only a specified number of constraints.

Our work makes use of two concepts from combinatorial optimization: total unimodularity and Lagrangian relaxation. Totally unimodular matrices are a common feature in combinatorial optimization, used to show the integrality of a polytope [16]. For example, the Bipartite Matching polytope and Maximum Flow polytope are integral, since their constraint matrices are totally unimodular(TU). In approximation algorithms, total unimodularity has been used in the context of reducing a hard problem to simpler instances, with a TU constraint matrix [2, 10, 11, 13, 18].

Lagrangian relaxation has been used to obtain approximation algorithms for a variety of problems. These problems typically have a single, or a few *complicating* constraints. Some examples are the k-median problem [12], constrained MST [19], k-MST [6], and bounded degree MST [14]. In the k-median problem [12] , the hard constraint is the requirement to open at most k facilities. Using Lagrangian relaxation, this constraint is brought into the objective function, and the resulting relaxation is the (easier) facility location problem.

1.1 Our Results and Techniques.

Garg *et al* [8] gave a 2-approximation algorithm for the multicut problem in trees via a primal-dual algorithm. We give a different proof showing that the primal (cut) LP has an integrality gap of 2. Our proof involves reducing general instances of multicut on trees to a special family of *non-crossing* instances, which have integral LP relaxations. Although our approach does not show that integrality gap of the dual multicommodity flow problem on trees

*Dept of Computer Science, Carnegie Mellon University. Email: dgolovin@cs.cmu.edu. Supported by NSF ITR grants CCR-0122581 (The ALADDIN project) and IIS-0121678

†Tepper School of Business, Carnegie Mellon University. Email: viswa@cmu.edu. Supported by NSF ITR grant CCR-0122581 (The ALADDIN project)

‡Tepper School of Business, Carnegie Mellon University. Email: mohits@andrew.cmu.edu. Supported by NSF ITR grant CCR-0122581 (The ALADDIN project) and NSF grant CCF-0430751

SODA '06, January 22-26, Miami, FL
 ISBN 0-89871-605-5/06/01

is 2, it leads to a constant factor approximation algorithm for the k-multicut problem on trees. We obtain the following approximation for k-multicut on trees.

THEOREM 1.1. *Given any instance of k-multicut on trees, and a fixed $\epsilon > 0$, there exists an algorithm running in polynomial time, which returns a solution of cost at most $(\frac{8}{3} + \epsilon) \cdot OPT$ where OPT is the cost of the optimal solution.*

We again reduce general instances to non-crossing instances, and show that the linear programming relaxation for special non-crossing instances has a small integrality gap. This is described in Section 3.

We also show that if we can relax the target k by a small factor, then we get an improved guarantee for the cost of the solution.

THEOREM 1.2. *Given any instance of k-multicut on trees, and a fixed $\epsilon > 0$, there exists a polynomial time algorithm which returns a solution separating at least $(1-\epsilon)k$ pairs, and has cost at most $(2+\epsilon)\cdot OPT$. Above, OPT is the cost of the optimal solution separating at least k pairs.*

Observe that this is not a bi-criteria approximation in the usual sense, where there is a tradeoff between the approximation factors of the two criteria. Here, the two criteria *compete together* with the time complexity in a manner similar to a PTAS. We prove Theorem 1.2 in Section 4. Here again, we use the Lagrangian relaxation to prove the existence of a good integral solution. This involves an interesting tree decomposition, and a bucketing argument.

We also note that solving the k-multicut problem in trees leads to an approximation algorithm in general graphs using the Räcke decomposition [17]. This is discussed in Section 5.

THEOREM 1.3. *Given a polynomial time α-approximation algorithm for k-multicut in trees, there is a polynomial time $O(\alpha \cdot \log^2 n \log\log n)$-approximation algorithm for k-multicut in general graphs.*

1.2 Related Work. Garg *et al* [7] gave an $O(\log l)$-approximation algorithm for multicut in general graphs. As mentioned earlier, Garg *et al* [8] obtained an improved guarantee of 2 for multicut on trees. Hochbaum [11] gave a family of problems for which 2-approximation algorithms could be obtained using total unimodularity. But all these problems had half-integral linear relaxations. Hochbaum [11] left as an open question, whether there is a 2-approximation algorithm for multicut on trees based on TU matrices. In this paper, we answer this question in the affirmative. This is interesting as the linear relaxation for multicut on trees is not half-integral [8]. A similar question for the problem of augmenting a spanning tree to a 2-edge connected subgraph is answered by Ravi [18], matching the 2-approximation algorithm of JáJá and Frederickson [4].

We note that the results in Theorem 1.1, and the 2-approximation algorithm for multicut on trees (Section 2), have been obtained independently by Levin and Segev [21], using similar techniques.

2 Multicut in Trees using Total Unimodularity

An instance of multicut on trees consists of a tree $T = (V, E)$, a cost function on the edges $c : E \to \mathcal{R}^+$, and a set of l pairs of vertices $\mathcal{P} = \{(s_i, t_i) : 1 \le i \le l\}$. We are required to find a minimum cost set of edges F which separates each pair in $\mathcal{P}$. That is, for each pair (s_i, t_i) there must be an edge $e_i \in F$ such that e_i is on the unique path from s_i to t_i in the tree T. We also assume without loss of generality that tree T is rooted at a special vertex r. Thus a tree multicut instance can be denoted as a tuple $(T, r, c, \mathcal{P})$. Such an instance is called *non-crossing*, if for each pair $(s_i, t_i) \in \mathcal{P}$, s_i is a descendent of t_i in the rooted tree (T, r).

In this section, we show a simple 2-approximation algorithm for multicut on trees, which matches the best known algorithm [8]. We first show that the natural LP relaxation for non-crossing instances is totally unimodular, and hence integral. Then we show how general multicut instances on trees can be reduced (via an LP) to non-crossing instances while losing a factor of 2, and thus obtain a 2-approximation.

We consider the following LP-relaxation for multicut on trees, as in [8].

$$
\begin{array}{lll}
 & \text{minimize } \sum_{e \in H} c_e x_e & \\
 & \text{subject to} & \\
(MLP) & \sum_{e \in P_i} x_e \ \ge \ 1 & \forall \ 1 \le i \le l \\
 & x_e \ge 0 & \forall \ e \in E
\end{array}
$$

Here P_i is the unique tree path from s_i to t_i. For an instance $\mathcal{I}$ of multicut on trees, we will denote its LP relaxation by $MLP(\mathcal{I})$. We first prove the following lemma for non-crossing instances.

LEMMA 2.1. *For any non-crossing instance $\mathcal{I}$ of multicut on trees, $MLP(\mathcal{I})$ is integral.*

Proof. We prove the lemma by showing that the constraint matrix of $MLP(\mathcal{I})$ is totally unimodular. Direct every edge in the tree T towards the root r. The constraint matrix can now be interpreted as a network matrix: arcs of T correspond to the columns of the matrix, and each constraint corresponds to a directed path from s_i to t_i. All the edges in this directed path are

forward and have a coefficient of 1. Hence, the matrix is totally unimodular ([16], page 548), and the lemma follows.

We now show how Lemma 2.1 can be used to give an alternate proof of the integrality gap of 2 for MLP on general instances of multicut on trees.

THEOREM 2.1. *([8]) Given any instance $\mathcal{I} = (T, r, c, \mathcal{P})$ of multicut on trees, the linear programming relaxation $MLP(\mathcal{I})$ has an integrality gap of at most 2.*

Proof. Let x^* be an optimal solution to $MLP(\mathcal{I})$. Using the solution x^*, we formulate a non-crossing instance $\mathcal{I}' = (T, r, c, \mathcal{P}')$ of multicut on the same rooted tree.

For each pair $(s_i, t_i) \in \mathcal{P}$ (the original instance), we include one pair in $\mathcal{P}'$ (the non-crossing instance) as follows. Let u_i be the highest common ancestor of s_i and t_i. Let P_i denote the path from s_i to t_i, P_i' the path from s_i to u_i, and P_i'' the path from u_i to t_i. Clearly $P_i = P_i' \cup P_i''$. From $MLP(\mathcal{I})$, $\sum_{e \in P_i} x_e^* \geq 1$. Hence, either $\sum_{e \in P_i'} x_e^* \geq \frac{1}{2}$ or $\sum_{e \in P_i''} x_e^* \geq \frac{1}{2}$. If the former is true, then include the pair (s_i, u_i) in $\mathcal{P}'$; else include the pair (u_i, t_i) in $\mathcal{P}'$.

Clearly, the instance constructed is non-crossing. Also, $2x^*$ is a feasible solution to the linear programming relaxation $MLP(\mathcal{I}')$ for the new instance. But from Lemma 2.1, $MLP(\mathcal{I}')$ is integral. So, $2x^* \geq \sum_j \lambda_j y_j$, where $\sum_j \lambda_j = 1$, $\lambda_j \geq 0$, and the y_j are *integral* vertices of the polytope $MLP(\mathcal{I}')$. Hence, for some j, we must have $2c \cdot x^* \geq c \cdot y_j$. Note that y_j is also a feasible solution to the original multicut instance $\mathcal{I}$: any solution that separates every pair in $\mathcal{P}'$ also separates every pair in $\mathcal{P}$. Thus the integrality gap of $MLP(\mathcal{I})$ is at most 2.

The above proof can also be used to obtain a polynomial time 2-approximation algorithm for the multicut problem on trees.

3 k-Multicut in Trees

An instance of k-multicut on trees consists of a rooted tree (T, r), a non-negative cost function c on the edges of T, a set of pairs of vertices $\mathcal{P} = \{(s_i, t_i) | 1 \leq i \leq l\}$, and a target $k \leq l$. The goal is to choose a minimum cost set of edges F, such that it separates *at least* k pairs in $\mathcal{P}$. We denote such an instance by a tuple $(T, r, c, \mathcal{P}, k)$. In this paper, when the k-multicut instance is clear from the context, we refer to a "pair in $\mathcal{P}$" simply as a "pair".

In this section, we present a constant factor approximation algorithm for k-multicut on trees. Our approach is similar to the previous section. We reduce general instances on the tree to non-crossing instances, and prove that the LP relaxation for non-crossing instances (suitably modified) has a small integrality gap. We consider the following LP relaxation for k-multicut on trees. Again, P_i denotes the unique s_i-t_i path in T.

$$
(BLP) \quad
\begin{array}{ll}
\text{minimize } \sum_e c_e x_e & \\
\text{subject to} & \\
\sum_{e \in P_i} x_e \geq y_i & \forall\ 1 \leq i \leq l \\
\sum_{i=1}^{l} y_i \geq k & \\
0 \leq y_i \leq 1 & \forall\ 1 \leq i \leq l \\
x_e \geq 0 & \forall\ e \in E
\end{array}
$$

The following theorem formalizes why a reduction from general tree instances to non-crossing instances would work. The proof is similar to Theorem 2.1, and is given in the appendix.

THEOREM 3.1. *Suppose $BLP(\mathcal{I}')$ has an integrality gap of at most ρ for every non-crossing instance $\mathcal{I}'$ of k-multicut on trees. Then $BLP(\mathcal{I})$ has an integrality gap of at most 2ρ for any instance $\mathcal{I}$ of k-multicut on trees.*

However, the integrality gap of BLP can be arbitrarily bad even for non-crossing instances (see Figure 1). We show that the integrality gap is due to costly edges in the optimum solution. We get rid of the costly edges by pruning, and then prove that BLP for such modified non-crossing instances has a small integrality gap.

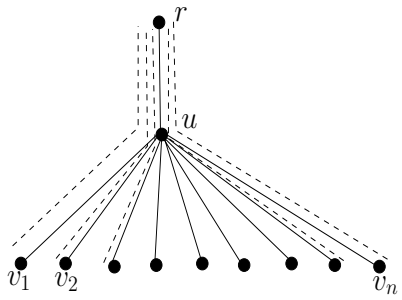

Figure 1: The pairs to be separated are (r, v_i) for $1 \leq i \leq l$ and the target $k = 1$. All edges cost 1. The LP solution has cost $\frac{1}{l}$ as it selects the edge (r, u) with weight $\frac{1}{l}$ while any integral solution must pay cost 1.

In Section 3.1, we show that, for any fixed $\epsilon > 0$, we can obtain a $(\frac{4}{3} + \epsilon)$ approximation algorithm for non-crossing k-multicut in trees, using the linear relaxation BLP. In Section 3.2, we show how this can be extended to obtain an $(\frac{8}{3} + \epsilon)$ approximation algorithm for general instances of k-multicut in trees.

3.1 LP-based Approximation for Non-crossing Instances.

In this section we consider only non-crossing instances. To overcome the high integrality gap

of BLP, we perform a preprocessing step. A similar idea, of pruning to reduce the integrality gap has been employed in the constrained MST problem [19], and the k-MST problem [6].

Given a non-crossing instance $\mathcal{I}$ of the k-multicut problem, let OPT denote the cost of its optimum solution. For any fixed $\epsilon > 0$, let H be the set of all edges of cost more than $\epsilon \cdot OPT$. Clearly, the optimum solution can pick at most $\frac{1}{\epsilon}$ edges from H. We first guess the value of OPT, and the edges in H picked by the optimum solution. For each such guess, we create a new reduced instance. We argue later (in Section 3.2) that there are at most $n \cdot n^{1/\epsilon}$ reduced instances. For every reduced instance, we solve its LP relaxation, and show that its fractional optimum can be rounded to an integral solution within a factor of $4/3 + \epsilon$. Finally, we output the cheapest solution found over all the guesses.

THEOREM 3.2. *Given any non-crossing instance $\mathcal{I}$ of k-multicut on trees, there exists an integral solution of cost at most $\frac{4}{3}OPT_{LP} + C_{max}$, where OPT_{LP} is the optimal value of $BLP(\mathcal{I})$, and C_{max} is the maximum edge cost in $\mathcal{I}$.*

Proof. We begin by solving the Lagrangian relaxation of $BLP(\mathcal{I})$ after dualizing the target constraint.

$$\begin{array}{rcll} LR(\lambda) & = & \min \sum_e c_e x_e + \lambda(k - \sum_{i=1}^{l} y_i) & \\ & & \text{subject to} & \\ & & \sum_{e \in P_i} x_e \ \geq \ y_i & \forall \ 1 \leq i \leq l \\ & & 0 \leq y_i \leq 1 & \forall \ 1 \leq i \leq l \\ & & x_e \geq 0 & \forall \ e \in E \end{array}$$

The constraint matrix of the above linear program is totally unimodular. This is a simple corollary of Lemma 2.1 (see also [16]). So, the optimal solution to this linear program for any fixed $\lambda > 0$ is integral. Solving the Lagrangian relaxation and its dual, we obtain two integral solutions S_1 and S_2 such that the optimal solution to BLP is a convex combination of these two solutions. Formally,

FACT 3.1. *At the optimal Lagrange multiplier λ^*, there exist two integral solutions $S_1 = (x^1, y^1)$ and $S_2 = (x^2, y^2)$ such that,*

1. *S_1 and S_2 are both optimal solutions to $LR(\lambda^*)$*
2. *Solution S_1 satisfies $k_1 = \sum_{i=1}^{l} y_i^1 \leq k$, and solution S_2 satisfies $k_2 = \sum_{i=1}^{l} y_i^2 \geq k$.*
3. *$\mu_1(x^1, y^1) + \mu_2(x^2, y^2)$ is an optimal solution to the linear program $BLP(\mathcal{I})$, where μ_1 and μ_2 satisfy $\mu_1 + \mu_2 = 1$ and $\mu_1 k_1 + \mu_2 k_2 = k$.*

For a discussion on Lagrangian relaxation, see chapter II.3 in [16].

From Fact 3.1, the optimum value of $BLP(\mathcal{I})$, $OPT_{LP} = \mu_1 \cdot c(S_1) + \mu_2 \cdot c(S_2)$, where $c(S_1)$ and $c(S_2)$ are the costs of the edges chosen in S_1 and S_2 respectively.

Using solutions S_1 and S_2, we construct two feasible integral solutions Q_1 and Q_2 such that, the cheaper of Q_1 and Q_2 costs at most $\frac{4}{3}OPT_{LP} + C_{max}$. If $k_1 = k$ (respectively, $k_2 = k$), solution S_1 (respectively, S_2) would be an optimal integer solution to $BLP(\mathcal{I})$. Below, we assume that $k_1 < k < k_2$.

Computing solution Q_1: The first solution Q_1 is constructed by picking a subset of edges from S_2 that separates at least k pairs. Let $F_2 = \{e \in E(T) | x_2(e) = 1\}$ be the set of edges chosen by solution S_2. Consider an assignment of the pairs separated in S_2 to edges of F_2 as follows: edge $e \in F_2$ is assigned the pairs it separates in solution S_2. If a pair is separated by two or more edges, it is assigned arbitrarily to any *one* of these edges. Let $\Pi(e)$ denote the set of pairs assigned to edge e. For a set of edges Q, let $\Pi(Q)$ denote $\bigcup_{e \in Q} \Pi(e)$. Clearly, $\{\Pi(e) | e \in F_2\}$ forms a partition of all the pairs separated in S_2. Define the price of an edge to be $p(e) = \frac{c_e}{|\Pi(e)|}$.

Constructing Solution Q_1:
Order the edges in F_2 in increasing order of prices.
Let the ordering be $(e_1, e_2, \ldots, e_r)$.
Initialize $Q_1 \leftarrow \phi$, $i \leftarrow 1$.
While $(|\Pi(Q_1)| < k)$
    $Q_1 \leftarrow Q_1 \cup \{e_i\}$
    $i \leftarrow i + 1$

CLAIM 1. *Q_1 is a feasible solution to the k-multicut instance $\mathcal{I}$.*

Proof. Since $|\Pi(F_2)| = k_2 > k$, the procedure eventually terminates , and from the termination condition we get $|\Pi(Q_1)| \geq k$.

CLAIM 2. $c(Q_1) \leq \frac{k}{k_2} c(S_2) + C_{max}$.

Proof. We show that the edges in $Q_1' = \{e_1, \ldots, e_{i-1}\} = Q_1 \setminus \{e_i\}$ (i.e. Q_1 except its last edge) cost no more than $\frac{k}{k_2} c(S_2)$. As Q_1' has the edges of smallest price, the weighted average price of edges in Q_1' is at most that of all edges in F_2. i.e.,

$$\frac{\sum_{e \in Q_1'} |\Pi(e)| \cdot p(e)}{\sum_{e \in Q_1'} |\Pi(e)|} \leq \frac{\sum_{e \in F_2} |\Pi(e)| p(e)}{\sum_{e \in F_2} |\Pi(e)|}$$

This implies $\sum_{e \in Q_1'} c_e \leq \frac{c(S_2)}{k_2} \cdot \sum_{e \in Q_1'} |\Pi(e)| < \frac{k}{k_2} c(S_2)$. The last inequality $|\Pi(Q_1')| < k$ follows from the

termination condition. Now an additional edge e_i can cost at most C_{max}, and we have the claim.

Computing solution Q_2: Now we construct the second solution Q_2. First, we pick all the edges in S_1. This separates $k_1 \leq k$ pairs. Hence, there are at least $k_2 - k_1$ pairs in S_2 which have not been separated by edges in S_1. We select a set of edges $F \subseteq F_2$ such that it separates at least $k - k_1$ of the $k_2 - k_1$ remaining pairs of S_2. We can do this in a manner identical to the construction of solution Q_1. The solution Q_2 then consists of the edges $F_1 \bigcup F$. So Q_2 is a feasible solution, satisfying:

CLAIM 3. $c(Q_2) \leq c(S_1) + \frac{k-k_1}{k_2-k_1} c(S_2) + C_{max}$

The following lemma shows that the better of the two solutions Q_1 and Q_2 is a good approximation.

LEMMA 3.1. $\min\{c(Q_1), c(Q_2)\} \leq \frac{4}{3} OPT_{LP} + C_{max}$.[1]

Proof. We will prove

$$\min\{\frac{\frac{k}{k_2} c(S_2)}{OPT_{LP}}, \frac{c(S_1) + \frac{k-k_1}{k_2-k_1} c(S_2)}{OPT_{LP}}\} \leq \frac{4}{3} \quad (1)$$

Using Claims 2 and 3, this implies Lemma 3.1. Recall that the optimal LP value $OPT_{LP} = \mu_1 c(S_1) + \mu_2 c(S_2)$, and $\mu_1 = \frac{k_2-k}{k_2-k_1}$ and $\mu_2 = \frac{k-k_1}{k_2-k_1}$. For notational convenience, we let $k_1 = a_1 \cdot k$, $k_2 = a_2 \cdot k$, and $r = \frac{c(S_1)}{c(S_2)}$. It is clear that for any instance, and solutions S_1 and S_2, $0 < a_1, r < 1$ and $a_2 > 1$. To prove (1), it suffices to show

$$\max_{0<a_1,r<1 \; a_2>1} \min\{f(a_1, a_2, r), g(a_1, a_2, r)\} \leq \frac{4}{3}$$

$$f(a_1, a_2, r) = \frac{a_2 - a_1}{a_2(r(a_2 - 1) + 1 - a_1)}$$

$$g(a_1, a_2, r) = \frac{r(a_2 - a_1) + 1 - a_1}{r(a_2 - 1) + 1 - a_1}$$

For any fixed values of $0 < a_1 < 1$ and $a_2 > 1$, f is decreasing with r while g is increasing. Thus the maximum value of $\min\{f(a_1, a_2, r), g(a_1, a_2, r)\}$ as r varies over the interval $(0, 1)$, is attained as $r \to 0$, $r \to 1$, or at $f(a_1, a_2, r) = g(a_1, a_2, r)$. It is easy to check that the minimum of these two functions as r goes to 0 or 1 is 1. Now we consider the case $f(a_1, a_2, r) = g(a_1, a_2, r)$. Solving for r, we get $r = \frac{a_1(a_2-1)}{a_2(a_2-a_1)}$. At this value of r, $f(a_1, a_2, r) = g(a_1, a_2, r) = \frac{(a_2-a_1)^2}{a_1(a_2-1)^2 + a_2(1-a_1)(a_2-a_1)} \leq \frac{4}{3}$ for $0 < a_1 < 1$ and $a_2 > 1$. This can be verified by rearranging the terms in the inequality to get $(a_1+a_2-2)^2 + 4(a_2-1)(1-a_1)^2 \geq 0$, which is clearly true.

[1]Segev [20] pointed out that it suffices to use solution S_2 instead of Q_1 for the lemma to hold.

Theorem 3.2 now follows.

We note that non-crossing instances of k-multicut can be solved in polynomial time by dynamic programming. We assume, without loss of generality, that the tree is binary (by adding edges of cost ∞). This is for the sake of the dynamic program only. A state in the dynamic program corresponds to: the subtree rooted at a vertex v, the first edge e_{up} that is cut on the path from v to r, and the number of pairs t with one end point below v that are separated. So we have a table $M[v, e_{up}, t]$ with $n^2 l$ entries. $M[v, e_{up}, t]$ is the minimum cost of a set of edges that separates exactly t pairs among those with an end point below v, satisfying the condition that e_{up} is the first edge cut on the path from v to r. The entry for leaves is trivial. The entry $M[v, e_{up}, t]$ for an internal vertex v is obtained from the entries $M[v_1, *, *]$ and $M[v_2, *, *]$, where v_1 and v_2 are the two children of v. Formally, $M[v, e_{up}, t]$ is given by the following recurrence

$$\min_{0 \leq z \leq t} \begin{cases} M[v_1, e_{up}, z] + M[v_2, e_{up}, t - z - q_1 - q_2] \\ \quad \text{(neither of } (u, v_1) \text{ and } (u, v_2) \text{ is picked)}, \\ M[v_1, (u, v_1), z] \\ + M[v_2, e_{up}, t - z - p_1 - q_2] \\ \quad \text{(only } (u, v_1) \text{ is picked)}, \\ M[v_1, e_{up}, z] \\ + M[v_2, (u, v_2), t - z - q_1 - p_2] \\ \quad \text{(only } (u, v_2) \text{ is picked)}, \\ M[v_1, (u, v_1), z] \\ + M[v_2, (u, v_2), t - z - p_1 - p_2] \\ \quad \text{(both } (u, v_1) \text{ and } (u, v_2) \text{ are picked)} \end{cases}$$

Above, p_1 (respectively, p_2) is the number of pairs with the lower vertex at v_1 (respectively, v_2). Similarly, q_1 (respectively, q_2) is the number of pairs with the lower vertex at v_1 (respectively, v_2) *and* which are separated by e_{up}.

However, we need a bounded integrality gap of the linear relaxation BLP, for the reduction in Theorem 3.1 to apply. Thus we do not directly get a 2-approximation algorithm for k-multicut on trees. In Theorem 3.2, we proved an integrality gap $\backsim 4/3$ for pruned non-crossing instances. In the next section, we use this to obtain an approximation algorithm for k-multicut on trees with a guarantee $\backsim 8/3$.

3.2 From Non-crossing Instances to General Tree Instances.

In this section, we present our approximation algorithm for k-multicut on trees. As mentioned, the basic idea is to reduce general instances to non-crossing instances. However, it is important to perform the preprocessing of large cost edges *before* reducing to non-crossing instances. We summarize our algorithm k-**TM** (for a fixed $\epsilon > 0$) in Figure 2.

Input: A k-multicut instance $\mathcal{I} = (T, r, c, \mathcal{P}, k)$ on a tree.

1. Guess the value of the optimal solution, OPT. Let $H = \{e \in E(T) | c_e > \epsilon \cdot OPT\}$.
2. For every $M \subseteq H$, of size at most $\frac{1}{\epsilon}$, do
   (a) Let $\mathcal{P}'$ be the pairs in $\mathcal{P}$ separated by M.
   (b) Let T' be the tree obtained from T, by contracting all edges in H.
   (c) Form a new instance $\mathcal{I}' = (T', r, c, \mathcal{P} \setminus \mathcal{P}', k - |\mathcal{P}'|)$, and solve $BLP(\mathcal{I}')$ to get a fractional solution x^*.
   (d) Use Theorem 3.1 to obtain a non-crossing instance $\mathcal{I}''$ from x^*.
   (e) Solve $\mathcal{I}''$ (with target $k - |\mathcal{P}'|$) optimally by dynamic programming, to get a solution y^M (if feasible).
3. Return the cheapest (feasible) solution $y^M \bigcup M$ that is found.

Output: An $\frac{8}{3} + \epsilon$ approximate solution to $\mathcal{I}$.

Figure 2: Algorithm k-**TM**

We first argue the correctness of this algorithm, and establish an approximation ratio of $8/3 + \epsilon$. Let y^* denote the optimal solution to $\mathcal{I}$, and OPT its cost. Assume that the algorithm guessed the correct value of the optimal solution, OPT, and also the edges $M \subseteq H$ chosen by y^*. $\mathcal{I}'$ is the new (reduced) instance, and x^* is the optimal solution to $BLP(\mathcal{I}')$. Let z^* denote the restriction of y^* to the edges of T'. Note that $c(y^*) = c(z^*) + c(M)$. Since y^* is a feasible solution to $\mathcal{I}$, z^* is clearly a feasible solution to $\mathcal{I}'$, and hence also to $BLP(\mathcal{I}')$. So, the optimal value of $BLP(\mathcal{I}')$, $opt(BLP(\mathcal{I}')) = c(x^*) \leq c(z^*)$. From Theorem 3.1, the optimal value of the non-crossing instance $\mathcal{I}''$, $opt(BLP(\mathcal{I}'')) \leq 2 \cdot opt(BLP(\mathcal{I}')) \leq 2 \cdot c(z^*)$. Using Theorem 3.2, the best integer solution to $\mathcal{I}''$, y^M has cost at most $\frac{4}{3} \cdot opt(BLP(\mathcal{I}'')) + \max\{c_e | e \in T'\} \leq \frac{8}{3} c(z^*) + \epsilon \cdot OPT$. Thus, the solution $y^M \bigcup M$ to $\mathcal{I}$ has cost at most $\frac{8}{3} c(z^*) + \epsilon \cdot OPT + c(M) \leq \frac{8}{3} c(y^*) + \epsilon \cdot OPT = (\frac{8}{3} + \epsilon) OPT$.

It is easy to see that this algorithm runs in polynomial time. We could run through possible values of OPT in time $\log_{1+\epsilon} C_{max}$ using a binary search (assuming minimum edge cost is 1). However, note that there are only n distinct possibilities for the set H of costly edges (by considering edges in a sorted order). So, instead of guessing OPT, we run this algorithm n times, one for each possible subset H. In each run of the algorithm, step 2 is run at most $n^{1/\epsilon}$ times. We can solve non-crossing instances of k-multicut (Step 2e) using a dynamic program in time $O(n^2 \cdot l^2)$. So the time taken by step 2 is dominated by the time taken to solve the LP in step 2c, t_{LP}. Thus, we can bound the running time of the whole algorithm by $O(n^{1+1/\epsilon}(n^2 \cdot l^2 + t_{LP}))$ which is polynomial. This proves Theorem 1.1.

4 Bi-criteria Approximation Algorithm

In this section, we present a $(2 + \epsilon, 1 - \epsilon)$ bi-criteria approximation algorithm for k-multicut on trees, *i.e.*, for every fixed $\epsilon > 0$, our algorithm finds a solution in polynomial time, that separates at least $(1 - \epsilon)k$ pairs, and costs at most $(2 + \epsilon)OPT$. Here OPT is the cost of the optimal solution to the k-multicut instance. This is unlike usual bi-criteria approximations, where there is a trade off between the two criteria being approximated. Here the trade off is between the running time and the approximation of *both* criteria, similar to a PTAS.

The bi-criteria approximation algorithm **Relax-k-TM** is the same as algorithm k-**TM** (Section 3.2), with the following modifications: in the pruning step (step 1), we consider all edges of cost larger than $(\frac{\epsilon}{16})^9 \cdot OPT$; and in solving the non-crossing instance $\mathcal{I}''$ (step 2e), we set the target to $(1 - \frac{\epsilon}{16})(k - |\mathcal{P}'|)$.

As before, to prove that the solution returned by the dynamic program has small cost, it suffices to show that non-crossing instances have small integrality gap. The following theorem proves this integrality gap guarantee.

THEOREM 4.1. *Given any non-crossing instance $\mathcal{I}$ of k-multicut on trees and a fixed $0 < \delta \leq \frac{1}{81}$, there exists an integral solution that separates at least $(1 - \delta^{1/4})k$ pairs in $\mathcal{I}$, and costs at most $(1 + 4\delta^{1/4})OPT_{LP} + \frac{2}{\delta^2} C_{max}$. Here OPT_{LP} is the optimal value of $BLP(\mathcal{I})$, and C_{max} is the maximum edge cost in $\mathcal{I}$.*

Before we prove Theorem 4.1, we prove Theorem 1.2

using Theorem 4.1.

Proof of Theorem 1.2: Consider the run of **Relax-k-TM** when it correctly guesses the optimal value OPT, and the edges $M \subseteq H$ of the optimal solution. We follow the notation in the proof of Theorem 1.1 (Section 3.2). y^* is the optimal solution to $\mathcal{I}$, and z^* denotes the restriction of y^* to the tree T'. Clearly, $c(y^*) = c(z^*) + c(M)$. Recall that the optimal value of the linear program $BLP(\mathcal{I}')$, $opt(BLP(\mathcal{I}')) \leq 2 \cdot c(z^*)$. Using Theorem 4.1 on the non-crossing instance $\mathcal{I}''$ with $\delta = (\frac{\epsilon}{16})^4$, we know that there exists an integer solution s^*, separating at least $(1-\frac{\epsilon}{16})(k-|\mathcal{P}'|)$ pairs, and having cost,

$$\begin{aligned} c(s^*) &\leq (1+\tfrac{\epsilon}{4}) opt(BLP(\mathcal{I}')) + \tfrac{2}{(\epsilon/16)^8} C_{max} \\ &\leq 2(1+\tfrac{\epsilon}{4}) \cdot c(z^*) + \tfrac{2}{(\epsilon/16)^8} C_{max} \\ &\leq 2(1+\tfrac{\epsilon}{4}) \cdot c(z^*) + \tfrac{\epsilon}{8} \cdot OPT \end{aligned}$$

where the last inequality follows from the pruning step. Now, y^M is the best solution separating at least $(1-\frac{\epsilon}{16})(k-|\mathcal{P}'|)$ pairs in $\mathcal{I}''$. So the cost of the final solution,

$$\begin{aligned} c(y^M) + c(M) &\leq c(s^*) + c(M) \\ &\leq 2(1+\tfrac{\epsilon}{4}) \cdot c(z^*) + \tfrac{\epsilon}{8} \cdot OPT + c(M) \\ &< (2+\epsilon) \cdot OPT \end{aligned}$$

Also, the number of pairs of $\mathcal{I}$ separated by the final solution is at least $(1-\frac{\epsilon}{16})(k-|\mathcal{P}'|) + |\mathcal{P}'| \geq (1-\frac{\epsilon}{16})k$. This proves Theorem 1.2.

Proof of Theorem 4.1 As in the proof of Theorem 3.2, we use the two integral solutions S_1 and S_2 obtained at the optimal Lagrange multiplier (Fact 3.1). Recall that k_1 (respectively, k_2) is the number of pairs separated by solution S_1 (respectively, S_2), and $k_1 < k < k_2$. Also Q_1 and Q_2 denote the solutions constructed in the proof of Theorem 3.2. The idea of this proof is to construct a new family of solutions using S_1 and S_2. We then show that one of the new solutions, or one of the solutions S_1, S_2, Q_1 or Q_2 satisfies the conditions of Theorem 4.1. Observe that each of S_2, Q_1 and Q_2 separates at least k pairs, and are feasible solutions for our problem. Whenever we use one of the other solutions, we will show that they separate the required number of pairs.

We begin by describing how to create the new family of solutions. We break the tree into a collection $\mathcal{T}$ of small subtrees by removing a few edges in the tree. From each subtree, we pick the edges of either solution S_1 or solution S_2 (not both). We do this in such a way as to separate at least $(1-2\sqrt{\delta})k$ pairs in total. The family of subtrees $\mathcal{T}$ will be edge disjoint, and the pairs separated by solutions S_1 and S_2 in these subtrees will also be disjoint. This gives us the freedom to combine the solutions S_1 and S_2 from each subtree *independently*. Below, we say that a pair (s,t) is contained in a subtree if the tree path between s and t lies completely in the subtree.

LEMMA 4.1. *There exists a set of at most $\frac{2}{\delta^2}$ edges F such that $T \setminus F$ can be decomposed into a family $\mathcal{T} = \{T_1, \ldots T_r\}$ of edge disjoint subtrees satisfying:*

1. *Any two distinct subtrees in $\mathcal{T}$ have at most one vertex in common.*
2. *Any pair that is not contained in a single subtree in $\mathcal{T}$ is separated by some edge in F.*
3. *In each subtree $T_i \in \mathcal{T}$, the number of pairs contained in T_i that are separated by edges in S_1 (respectively, S_2) is at most $\delta^2 k_1$ (respectively, $\delta^2 k_2$).*

Proof. First we show how to construct an edge set F_1, and a decomposition of $T \setminus F_1$ into a family $\mathcal{T}_1$ of subtrees. This family will satisfy properties 1, 2, and 3 only for solution S_1. We start with the rooted tree (T, r). In each step, find a deepest edge $e = (u, v)$, where u is the parent of v, such that the subtree below v has at least $\delta^2 k_1$ pairs separated by edges of S_1 inside the subtree. If there is no such edge, we include each subtree hanging from the root into $\mathcal{T}_1$, and our construction is complete. Otherwise, we include e in F_1 and include each subtree hanging from v (with v as its root) into $\mathcal{T}_1$. Since the k-multicut instance is non-crossing, there is no pair with end points in two different subtrees of v; so property 2 holds. We then recurse on the remaining tree. Clearly, each time an edge is included in F_1, we decrease the number of pairs to be considered by at least $\delta^2 k_1$. Hence, there can be at most $\frac{1}{\delta^2}$ edges in F_1. Each subtree in this family has at most $\delta^2 k_1$ pairs separated by edges of S_1. From the construction, it is clear that properties 1 and 2 hold.

Now consider the same procedure on the original tree T, to get a set of edges F_2 corresponding to solution S_2. We set $F = F_1 \cup F_2$. For each edge $e \in F$, mark the child vertex of e. The root r is also marked. Now in forest $T \setminus F$, each subtree hanging from a marked vertex is returned as a distinct subtree. It is easy to check that these subtrees satisfy all the conditions in the lemma.

Our solution will always contain all edges in F. Clearly, any pair of S_1 or S_2 that has been separated by $e \in F$ belongs to our solution. By property 2 of Lemma 4.1, each of the remaining pairs separated by S_1 or S_2 can be assigned to a unique subtree. We denote the number of pairs in solution S_1 assigned to a subtree T_i by $w_1(T_i)$. Similarly, we define the function

$w_2(T_i)$ for each subtree T_i. Observe that $w_1(T_i) \leq \delta^2 k_1$ and $w_2(T_i) \leq \delta^2 k_2$ for each i. For pairs which are separated by edges in F, we arbitrarily assign the pair to any *one* edge in F that separates it. We then introduce a dummy subtree T_e, for each edge $e \in F$. We assign a w_1(respectively, w_2) weight to the dummy subtree, equal to be the number of pairs separated by S_1 (respectively, S_2) that are assigned to e. If $w_1(T_e) > \delta^2 k_1$ or $w_2(T_e) > \delta^2 k_2$, we introduce multiple dummy subtrees for this edge so that the weight is distributed to satisfy the inequality $w_i(T') \leq \delta^2 k_i$ $(i = 1, 2)$, for each dummy subtree T' of e. Let $T_1, \ldots, T_l$ denote the subtrees given by Lemma 4.1, along with the dummy subtrees. Observe that $\sum_j w_i(T_j) = k_i$ for $i = 1, 2$. Also, if a pair contributes to the w_1 *or* w_2 weight of one subtree, it does not contribute *any* weight to any other subtree.

Next we bucket the subtrees $T_1, \ldots, T_l$ in such a way that each bucket B_j (except the last) has almost equal w_1 weight as well as w_2 weight.

LEMMA 4.2. *There exists a partition $B_1, \ldots, B_{t+1}$ of $\mathcal{T} = \{T_1, \ldots, T_l\}$ into $t+1$ buckets, such that $\delta k_i \leq \sum_{T \in B_j} w_i(T) \leq \delta k_i + 3\delta^2 k_i$ for each $i = 1, 2$ and for each $1 \leq j \leq t$. Also, $\frac{1}{\delta + 3\delta^2} - 1 \leq t \leq \frac{1}{\delta}$.*

Proof. We construct the buckets in a greedy fashion.

1. $\mathcal{S} \leftarrow \{T_1, \ldots, T_l\}$, $n_1, n_2 \leftarrow 0$, $j \leftarrow 1$, $B_1 \leftarrow \phi$.
   // Open bucket B_1
2. While $(\mathcal{S} \neq \phi)$ do:
   (a) $i \leftarrow \text{argmin}_l \{\frac{n_l}{k_l} : l = 1, 2\}$.
   (b) Pick $T' \in \mathcal{S}$ such that $\frac{w_i(T')}{k_i} \geq \frac{w_{3-i}(T')}{k_{3-i}}$.
   (c) $B_j \leftarrow B_j \bigcup \{T'\}$, $\mathcal{S} \leftarrow \mathcal{S} \setminus \{T'\}$.
   (d) $n_1 \leftarrow n_1 + w_1(T')$, $n_2 \leftarrow n_2 + w_2(T')$.
   (e) If $\frac{w_1(B_j)}{k_1} \geq \delta$ and $\frac{w_1(B_j)}{k_1} \geq \delta$,
   // Close bucket B_j and Open bucket B_{j+1}
   $j \leftarrow j + 1$ and $B_j \leftarrow \phi$.
3. $t \leftarrow j - 1$, return buckets $B_1, \ldots, B_j$.

Here, n_1 (n_2) denotes the total w_1-weight (w_2-weight) of the subtrees that have been assigned to buckets so far. Since $\frac{\sum_{j=1}^{l} w_1(T_j)}{k_1} = \frac{\sum_{j=1}^{l} w_2(T_j)}{k_2} = 1$, it is clear that there is always a subtree $T' \in \mathcal{S}$ satisfying the condition of step 2b. Each subtree $T' \in \mathcal{T}$ satisfies $\frac{w_i(T')}{k_i} \leq \delta^2$ $(i = 1, 2)$. So, at each iteration in the while loop of step 2, $|\frac{n_1}{k_1} - \frac{n_2}{k_2}| \leq \delta^2$.

The following argument is true for all but the last bucket. For bucket B_j $(1 \leq j \leq t)$, let $\beta_j = |\frac{w_1(B_j)}{k_1} - \frac{w_2(B_j)}{k_2}|$. We will show that $\beta_j \leq 2\delta^2$. Let m_1 and m_2 denote the values of variables n_1 and n_2 respectively, at the point when bucket B_j is opened. Similarly, p_1 and p_2 denote the values of variables n_1 and n_2, at the point when bucket B_j is closed. Clearly, $p_1 = m_1 + w_1(B_j)$ and $p_2 = m_2 + w_2(B_j)$. From the preceding argument, $|\frac{m_1 + w_1(B_j)}{k_1} - \frac{m_2 + w_2(B_j)}{k_2}|$ and $|\frac{m_1}{k_1} - \frac{m_2}{k_2}|$ are at most δ^2.

$$\begin{aligned} \delta^2 &\geq |\tfrac{w_1(B_j)}{k_1} - \tfrac{w_2(B_j)}{k_2} + \tfrac{m_1}{k_1} - \tfrac{m_2}{k_2}| \\ &\geq |\tfrac{w_1(B_j)}{k_1} - \tfrac{w_2(B_j)}{k_2}| - |\tfrac{m_1}{k_1} - \tfrac{m_2}{k_2}| \\ &\geq \beta_j - \delta^2 \end{aligned}$$

Thus $\beta_j \leq 2\delta^2$. So $\frac{w_i(B_j)}{k_i} \leq 2\delta^2 + \frac{w_{3-i}(B_j)}{k_{3-i}}$ $(i = 1, 2)$. Consider the point just before the addition of the last subtree to B_j. Suppose $\frac{w_i(B_j)}{k_i} < \delta$ before adding this subtree. After B_j was closed (step 2e), $\frac{w_i(B_j)}{k_i} \geq \delta$. Since each subtree can increase this ratio by at most δ^2, $\frac{w_i(B_j)}{k_i} \leq \delta + \delta^2$. Thus $\frac{w_{3-i}(B_j)}{k_{3-i}} \leq \delta + 3\delta^2$. So for each bucket B_j, we have $\delta k_i \leq w_i(B_j) \leq (\delta + 3\delta^2) k_i$ $(i = 1, 2)$. Since the total w_1 weight (over $\mathcal{T}$) is exactly k_1, the number of buckets is $\frac{1}{\delta + 3\delta^2} \leq t + 1 \leq \frac{1}{\delta}$. This proves Lemma 4.2.

Recall that $\mu_1 = \frac{k_2 - k}{k_2 - k_1}$ and $\mu_2 = \frac{k - k_1}{k_2 - k_1}$. Let $t_1 = \lfloor \mu_1 t \rfloor$, and $t_2 = \lfloor \mu_2 t \rfloor$. We leave out the last bucket B_{t+1}, and define t integral solutions $R_1, \ldots, R_t$ using buckets $B_1, \ldots, B_t$. For each $1 \leq i \leq t$, solution R_i picks edges of S_1 in all subtrees in buckets $B_i, \ldots, B_{t_1+i-1}$ and edges of S_2 in buckets $B_{t_1+i}, \ldots, B_{t_1+t_2+i-1}$ where indexes are modulo t. In addition, all these solutions contain F.

CLAIM 4. *There exists an $1 \leq i \leq t$ such that $c(R_i) \leq OPT_{LP} + c(E_0) \leq OPT_{LP} + \frac{2}{\delta^2} C_{max}$.*

Proof. The total cost of these t solutions is at most $t_1 \cdot c(S_1) + t_2 \cdot c(S_2) + t \cdot c(F) \leq t \cdot OPT_{LP} + t \cdot c(F)$. So the cheapest of these solutions costs at most $OPT_{LP} + c(F) \leq OPT_{LP} + \frac{2}{\delta^2} C_{max}$.

CLAIM 5. *The number of pairs separated by any R_i is at least $(1 - 4\delta - \frac{\delta}{\mu})k$, where $\mu = \min\{\mu_1, \mu_2\}$*

Proof. We prove it for solution R_1 and the same argument holds for each R_i. Firstly, observe that the number of pairs separated by R_1 is

$$sep(R_1) \geq \sum_{i=1}^{t_1} w_1(B_i) + \sum_{i=t_1+1}^{t_1+t_2} w_2(B_i)$$

Above, the first term in the RHS is just the pairs separated by solution S_1 assigned to subtrees $T_j \in B_i$ for $1 \leq i \leq t_1$, and the second term is the pairs separated by solution S_2 assigned to subtrees $T_j \in B_i$

for $t_1 + 1 \leq i \leq t_1 + t_2$. Recall that none of the pairs is assigned to two distinct subtrees (dummy or real). Now, using $w_j(B_i) \geq \delta k_j$ for each $j = 1, 2$, we have that $sep(R_1) \geq \delta k_1 \cdot t_1 + \delta k_2 \cdot t_2 \geq (\mu_1 k_1 + \mu_2 k_2)\delta t - \delta(k_1 + k_2) \geq \frac{k}{1+3\delta} - \delta k - \frac{\delta k}{\mu} \geq (1 - 4\delta - \frac{\delta}{\mu})k$.

First assume that $\mu > \sqrt{\delta}$. Then the number of pairs separated by each solution is at least $(1 - 2\sqrt{\delta})k$ (for $\delta \leq \frac{1}{9}$) proving Theorem 4.1. Else, $\mu < \sqrt{\delta}$, and from Claim 6, for $\mu \leq \sqrt{\delta}$, and $\gamma = \delta^{\frac{1}{4}}$ we have an integral solution which has cost at most $(1 + 4\delta^{1/4})OPT_{LP} + C_{max}$, and separates at least $(1 - \delta^{1/4})k$ pairs, proving Theorem 4.1 for this case as well.

5 k-Multicut in General Graphs

We extend our solution on trees to general graphs using the Räcke decomposition tree of a graph [17]. Räcke showed how to obtain a hierarchical decomposition, T_G, of any undirected capacitated graph G, that allows one to reduce any multi-commodity flow instance on G to one on T_G. This reduction loses only a poly-logarithmic factor in the congestion, henceforth called the *congestion gap* β. For efficient algorithms to generate such decompositions see [9]. The best congestion gap known to date, $\beta = O(\log^2 n \log\log n)$, was given by Harrelson *et al* [9]. Alon *et al.* [1] showed how this decomposition tree can be used to simulate multicuts in graphs by multicuts in trees. In particular, we get the following lemma.

LEMMA 5.1. *([1]) For any multicut instance $\mathcal{M} = \{(s_i, t_i) | 1 \leq i \leq l\}$, $c_G \leq c_T \leq 2\beta c_G$, where c_G is the optimal value of $\mathcal{M}$ in G, and c_T is the optimal value of $\mathcal{M}$ in T_G.*

Proof of Theorem 1.3: Consider a k-multicut instance $\mathcal{I}_G$ on G with pairs $\mathcal{P}$, and target k. Let the optimal solution E^* separate pairs $P^* = \{(s_i, t_i) | 1 \leq i \leq k\} \subseteq \mathcal{P}$. Then by Lemma 5.1, there exists a multicut E_t in *tree* T_G, separating pairs P^*, of cost at most $2\beta \cdot OPT$. Here OPT is the optimal value of the k-multicut instance in G. Define a k-multicut instance $\mathcal{I}_T$ on T_G with the same pairs $\mathcal{P}$, and target k. E_t is clearly a feasible solution to $\mathcal{I}_T$. Thus, an α-approximate solution to $\mathcal{I}_T$ costs at most $2\alpha\beta \cdot OPT$. From the proof of Lemma 5.1, any feasible solution to $\mathcal{I}_T$ is also feasible in $\mathcal{I}_G$ and of the same cost. Thus we have a $2\alpha\beta$-approximation algorithm for k-multicut on general graphs.

6 Conclusions and Open Problems

In this paper, we showed a new proof of the 2-approximation for multicut on trees, using totally unimodular matrices. It will be interesting to see more examples of proving integrality gaps using total unimodularity, or other notions of proving integrality like total dual integrality. A similar issue is addressed in Hassin and Segev [10]

In Section 3, we saw that the linear relaxation BLP has an integrality gap of $\lesssim \frac{4}{3}$ for pruned non-crossing instances. We have an example which shows that the analysis of Section 3 is tight. It would be interesting to construct an example showing an integrality gap larger than $1 + c$, for some constant $c > 0$, for pruned non-crossing instances.

To obtain improved guarantees for our algorithm **k-TM**, it suffices to show the *existence* of a small integrality gap of BLP. It is not necessary to have an efficient rounding scheme. This can be seen in Section 4, where the algorithm **Relax-k-TM** was essentially the same as **k-TM**, although the analysis was more intricate.

7 Acknowledgements

We would like to thank R. Ravi for useful discussions and comments on this paper. We also thank Danny Segev for comments on this paper.

References

[1] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph (Seffi) Naor. A general approach to online network optimization problems. *Proc. ACM-SIAM symposium on Discrete algorithms*, pages 577–586, 2004.

[2] Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree. In *Proc. International Colloquium on Automata, Languages and Programming*, pages 410–425, 2003.

[3] Lester R. Ford and Delbert R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[4] Greg N. Frederickson and Joseph JáJá. Approximation algorithm for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981.

[5] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.

[6] Naveen Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. *Proc. ACM Symposium on Theory of Computing*, pages 396–402, 2005.

[7] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *Proc. ACM Symposium on Theory of Computing*, pages 698–707, 1993.

[8] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1997.

[9] Chris Harrelson, Kirsten Hildrum, and Satish Rao. A polynomial-time tree decomposition to minimize congestion. *Proc. ACM Symposium on Parallel Algorithms and Architectures*, pages 34–43, 2003.

[10] Refael Hassin and Danny Segev. Rounding to an integral program. *Proc. International Workshop on Efficient and Experimental Algorithms*, pages 44–54, 2005.

[11] Dorit S. Hochbaum. Instant recognition of half integrality and 2-approximations. *Proc. International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 99–110, 1998.

[12] Kamal Jain and Vijay V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. *Proc. IEEE Symposium on Foundations of Computer Science*, page 2, 1999.

[13] Stavros G. Kolliopoulos and Clifford Stein. Approximation algorithms for single-source unsplittable flow. *SIAM J. Comput.*, 31(3):919–946, 2001.

[14] Jochen Könemann and R. Ravi. A matter of degree: improved approximation algorithms for degree-bounded minimum spanning trees. *Proc. ACM symposium on Theory of computing*, pages 537–546, 2000.

[15] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[16] George L. Nemhauser and Laurence A. Wolsey. Integer and Combinatorial Optimization. 1999.

[17] Harald Räcke. Minimizing congestion in general networks. *Proc. IEEE Symposium on Foundations of Computer Science*, pages 43–52, 2002.

[18] R. Ravi. 2-approximation for augmentating a tree to be 2-edge connected using total unimodularity. Personal Communication. 2004.

[19] R. Ravi and Michel X. Goemans. The constrained minimum spanning tree problem. *Proc. Scandinavian Workshop on Algorithm Theory*, pages 66–75, 1996.

[20] Danny Segev. Personal Communication. 2005.

[21] Danny Segev and Asaf Levin. Partial multicuts in trees. *To appear, Workshop on Approximation and Online Algorithms*, 2005.

[22] David B. Shmoys. Cut problems and their application to divide-and-conquer. *Approximation Algorithms for NP-hard Problems*, pages 192–235, 1997.

A Proofs

A.1 Proof of Theorem 3.1 The proof is similar to proof of Theorem 2.1. Take any instance of k-multicut on trees, i.e., a rooted tree (T, r), a set of pairs $\mathcal{P}$ to separate, and a target k. Let (x^*, y^*) denote the optimum solution to $BLP(\mathcal{I})$. We create a non-crossing instance of the k-multicut problem. For every pair $(s_i, t_i) \in \mathcal{P}$, let u_i denote the highest common ancestor of s_i and t_i. Let P_i denote the path from s_i to t_i, P_i' the path from s_i to u_i, and P_i'' the path from u_i to t_i. Clearly $P_i = P_i' \cup P_i''$. Also, $\sum_{e \in P^i} x_e^* \geq y_i$. Hence, either $\sum_{e \in P'^i} x_e^* \geq \frac{y_i}{2}$ or $\sum_{e \in P''^i} x_e^* \geq \frac{y_i}{2}$. If the former is true, then include the pair (s_i, u_i) in $\mathcal{P}'$ else include the pair (u_i, t_i).

The new instance consists of the pairs in $\mathcal{P}'$ and target k. Clearly, the new instance is non-crossing. We claim that $(2x^*, y^*)$ is feasible solution to $BLP(\mathcal{I}')$. The last three set of constraints are clearly satisfied. The feasibility of the first set of constraints follows from the way $\mathcal{P}'$ was selected. By the assumption in the theorem, the integrality gap of $BLP(\mathcal{I}')$ is at most ρ. Now, an argument as in Theorem 2.1 shows that the integrality gap of $BLP(\mathcal{I})$ is at most 2ρ.

A.2 Claim in proof of Theorem 4.1

CLAIM 6. *Suppose the solutions S_1 and S_2 at the optimal Lagrange multiplier satisfy $\min\{\mu_1, \mu_2\} = \mu$. Then for any $\mu < \gamma \leq \frac{1}{3}$, there is an integer solution that separates at least $(1 - \frac{\mu}{\gamma})k$ pairs, and costs at most $(1 + 4\gamma)OPT_{LP} + C_{max}$*

Proof. Suppose $\mu = \mu_1$. Then solution S_2 has cost at most $\frac{OPT_{LP}}{\mu_2} = \frac{OPT_{LP}}{1-\mu} \leq (1+2\mu)OPT_{LP}$ (for $\mu < 1/2$), which is at most $(1 + 2\gamma)OPT_{LP}$. Since S_2 separates $k_2 \geq k$ pairs, it is the claimed solution.

Now suppose $\mu = \mu_2$. If $k - k_1 \leq \frac{\mu}{\gamma}k$, solution S_1 separates $k_1 \geq (1 - \frac{\mu}{\gamma})k$ pairs, and costs at most OPT_{LP}. So S_1 is the claimed solution.

Else, $k - k_1 \geq \frac{\mu}{\gamma}k$. In this case, we show that the integer solution constructed in Theorem 3.2 is the claimed solution. Clearly that solution separates at least k pairs. Now, $k_2 - k_1 \geq \frac{1}{\mu} \cdot \frac{\mu}{\gamma}k = \frac{k}{\gamma}$. i.e. $k_2 > \frac{k}{\gamma}$. So, in the proof of Lemma 3.1, $a_2 = k_2/k > \frac{1}{\gamma}$. In this case, we can bound the expression $\frac{(a_2-a_1)^2}{a_1(a_2-1)^2+a_2(1-a_1)(a_2-a_1)}$ by $\frac{1}{(1-\gamma)^2} \leq (1 + 4\gamma)$ (for $\gamma \leq 1/3$). Thus, the integer solution given by Theorem 3.2 has cost at most $(1 + 4\gamma)OPT_{LP} + C_{max}$.

To complete the claim made in Theorem 4.1, we use $\mu = \delta^{1/2}$, and $\gamma = \delta^{1/4}$.