Min-Max Graph Coverings using Trees and Stars

Shashwat Garg, Sanjit Singh Batra, Naveen Garg

September 15, 2014

1 Problem Statement

MMkTC
Given a graph $G = (V, E)$, with a weight function on the edges, $w : c \to \Re^+$, integer $k > 0$. Find trees $T_1, ..., T_k$ such that $\cup_i V(T_i) = V$, so as to minimise $max_i w(T_i)$, where $w(T_i) = \sum_{e \in T_i} w(e)$
Best result so far: 3-approximation on general graphs
Our work: $(2 + \epsilon)$ on trees

2 Our algorithm

Lemma 1. *There exists an edge-disjoint solution of MMkTC such that max tree cost is at most twice the optimum cost. Furthermore, such a solution using only the edges used in optimum solution exists.*

Proof. Consider the optimum trees of MMkTC with max tree cost B. Remove the edges not covered by optimum solution. This breaks the graph into components, where each component is covered by some optimum trees and no tree touches more than one component. Let k_i be the number of optimum trees in the i^{th} component. Then, since all these k_i trees are connected to each other, the cost of edges covered in C_i is at most $k_i B$. Now we can use the edge decomposition of [1] to decompose in range $[B, 2B]$ to get k_i edge-disjoint trees of weight at most $2B$. Since edge-decomposition creates no new edges, the second part of lemma is satisfied. □

Lemma 2. *It is NP-hard to find the optimum set of edge-disjoint trees even for $k = 2$.*

Proof. We will prove this by a reduction from Subset Sum. Given a set of n numbers, of total sum, $2B$, we have to find if there exists a subset of sum exactly B.

Make a star with n edges. Weight of the edge i for $1 \leq i \leq n$ is equal to the i^{th} number of the set. There exists an edge-disjoint solution of MMkTC for $k = 2$ of cost B if and only if there is a subset of sum B. □

We now present a PTAS for the problem of covering a tree with k edge disjoint trees, so as to minimise the maximum weight of a tree. Combined with Lemma 1, this will give a $(2 + \epsilon)$-approximation for MMkTC on trees.

Let $w_T(v)$ be the weight of the subtree rooted at v, $ch(v)$ be the set of children of v, T_v be the subtree rooted at v. Let *approxKnapsack(W, A)* be the algorithm that gives a

$(1-\epsilon)$-approximation, run on instance of sack weight W and elements drawn from set A, with weight and value of the elements being equal. *approxKnapsack(W,A)* returns the set of items it covered. Let *approxMultiSched(k, A)* be a $(1+\epsilon)$-approximation to multiprocessor scheduling , run on k machines with job sizes drawn from the set A. It returns the makespan.

Algorithm 1 MMkTC on a tree: Cover the tree with edge disjoint trees of weight at most $B+\epsilon$

while weight of tree is more than B **do**
  Find the deepest vertex v such that $w_T(v) > B$
  $A_v = \phi$
  for $u \in ch(v)$ **do**
    if $w_T(u) + w(u,v) \geq B$ **then**,
      include T_u as a tree in our solution
      delete u from the graph
    else
      $A_v = A_v \cup \{w_T(u) + w(u,v)\}$
    end if
  end for
  for $k_v = 1$ to n **do**
    for $W = 1$ to B **do**
      $C_v = A_v$
      B_v=*approxKnapsack(W, C_v)*
      $C'_v = C_v \setminus B_v$
      M =*approxMultiSched($k_v - 1, C'_v$).*
      if $M \leq (B+\epsilon)$ **then**
        for each machine schedule, make a tree in our solution
        delete all those vertices from the graph
        exit both for loops
      end if
    end for
  end for
end while

Lemma 3. *Consider a vertex v that is not the root. Then, in an optimal edge disjoint covering by trees, there can be at most one tree which crosses v i.e. which covers portions from T_v and also vertices not in T_v.*

Proof. This follows since the trees are edge disjoint and any such tree will need to cover the edge from v to its parent. □

Lemma 4. *If $w_T(v) \leq OPT$(edge disjoint) for some vertex v, then we can assume that T_v is covered by a single tree in the optimal solution.*

Proof. By the previous lemma, there can be at most one tree crossing v. If more than one tree are covering T_v, then there must be a tree, T', which is fully contained in T_v. Since $w_T(v) \leq OPT$, we can make T' cover all of T_v. If there are other trees fully contained in T_v, we discard them. If there is a tree which crosses v, then we cut it so that it no longer enters T_v. □

Lemma 5. *Algorithm 1 returns at most k edge disjoint trees of weight no more than $B + \epsilon$ if $B \geq$ optimum of edge disjoint MMkTC on tree.*

Proof. In the algorithm, for each vertex, we try to cover its subtree with the fewest number k_v of trees, and for the smallest W, giving preference to smaller k_v over smaller W. We will prove the theorem by comparing with the optimum solution side by side.

Consider a deepest vertex v with $w_T(v) > B$. It has all children subtree of weight at most B. Let $\mathcal{T} = \{T_1, ..., T_{k_v^*}\}$ be edge disjoint optimal trees covering T_v. For $u \in ch(v)$, if $w_T(u) + w(u, v) \geq B$, then we include T_u in our solution and delete T_u from the graph. Optimum must have used at least one full tree to cover T_u since $w_T(u) + w(u, v) \geq B$. For all other children u, either the edge connecting them to v is already covered by some tree of of optimum, or we can add it to the tree covering T_u without making the weight of any tree more than B.

Let the tree which crosses v in $\mathcal{T}$ be T_1, having weight $W^* \leq B$ If no tree crosses v, then we take T_1 to be any tree covering T_v. Consider the run of our algorithm for v when $k_v = k_v^*$ and $W = W^*$. In line 15, we will make a tree and fill it with children subtrees (along with the edge connecting them to v) of total weight at least $(1 - \epsilon)W^*$. The weight remaining now is at most ϵW^* and the weight of trees other than T_1 used to cover T_v. For this, we find a feasible solution in line 17. If we get a feasible solution with the same value of k_v^* but a lower W, we only benefit, thus we are justified in taking the smallest W. Now, we can delete all trees covering T_v other than T_1 since they do not cover anything outside T_v, and since $W^* \leq B$, v no longer has $w_T(v)$ more than B.

It is possible that we return a feasible solution with $W > W^*$ and $k_v < k_v^*$. But in such a case, we only benefit, because we are saving a tree, which can be used later on at a parent stage. Another way to see this is that we attach an empty tree to v.

All the trees returned have weight at most $B + \epsilon$. By comparing with the optimum solution, we see that at each stage, we return no more trees than what optimum used. Thus, total number of trees used is no more than k.

□

We can modify the algorithm to do binary search on W and finding the smallest value for which we get a feasible schedule instead of running for all values from 1 to B. With this, the running time of the algorithm is polynomial in the size of the input.

Theorem 6. *We get a $(2 + \epsilon)$-approximation algorithm for MMkTC on a tree.*

Proof. Let OPT be the cost of an optimum solution and suppose we have guessed this value. From lemma 1, we know that optimum of edge-disjoint MMkTC $\leq 2.OPT$. From lemma 5, we can find a solution of cost $(1 + \epsilon)$optimum edge-disjoint $\leq 2(1 + \epsilon)OPT$. We can guess the optimum cost to within an ϵ error by binary search, thus this does not change the approximation guarantee. □

References

[1] Even, Garg, Konemann, Ravi, Sinha *Min-Max Tree Covers of a Graph* 2003

[2] Morsy *Improved Approximation Algorithm for Min-Max Rooted Tree Cover Problem* (unpublished)

[3] Khani, Salavatipour *Improved Approximation Algorithms for the Min-Max Tree Cover and Bounded Tree Cove Problems* 2011

[4] Xu, Wen *Approximation hardness of min-max tree covers* 2010

[5] Nagamochi, Okada *Approximating the minmax rooted-tree cover in a tree* 2007

[6] Nagamochi, Kawada *Approximating the min-max subtree cover problem in a cactus* 2004

[7] Nagamochi, Okada *Polynomial time 2-approximation algorithms for the minmax subtree cover problem* 2003

[8] Ravi, Goemans *The constrained minimum spanning tree problem*

[9] Arkin, Hassin, Levin *Approximations for the minimum and min-max vehicle routing problems*

[10] Even, Garg, Konemann, Ravi, Sinha *Covering graphs using stars and trees* 2003

[11] Skutella *Convex quadratic and semidefinite programming relaxations in scheduling* 2001

[12] Calinescu, Karloff, Rabani *An Improved Approximation Algorithm for Multiway Cut*

[13] Dahlhaus, Johnson, Papadimitriou, Seymour, Yannakakis *The complexity of multi terminal cuts* 1994

[14] Li, Pan *Solving the finite min-max problem via an exponential penalty method* 2003