Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 21, no. 3, pp. 265–280 (2017)
DOI: 10.7155/jgaa.00416

Generalized Bounded Tree Cover of a Graph

Barun Gorain[1] *Partha Sarathi Mandal*[2]
Krishnendu Mukhopadhyaya[3]

[1]Indian Statistical Institute, Kolkata, India
[2] Indian Institute of Technology Guwahati, India
[3]Indian Statistical Institute, Kolkata, India

Abstract

A tree cover of a graph is a collection of subgraphs such that each vertex is a part of at least one subgraph and each subgraph is a tree. The bounded tree cover problem (BTC) finds a tree cover with minimum number of trees of bounded weight. This paper considers several generalized versions of BTC. The first problem deals with graphs having multiple metric weight functions. Strong and weak tree cover problems are two variations of the first problem. In *strong tree cover*, every tree must be bounded with respect to all weight functions, whereas in *weak tree cover*, each tree must be bounded with respect to at least one weight function. A 4-approximation algorithm for strong tree cover and an $O(\log n)$-approximation algorithm for weak tree cover problem are proposed. The objective of the second problem is to find a tree cover where bounds of the trees in the tree cover are not necessarily same. It is proved that this problem cannot be approximated within a constant factor unless P=NP. A constant factor approximation algorithm is proposed when the ratio of maximum and minimum bounds is bounded by a constant. The third problem considers BTC for a graph with a general weight function which is not necessarily metric. A 3-approximation algorithm is proposed for this problem.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
May 2016	November 2016	December 2016	December 2016	January 2017
		Published:		
		February 2017		
	Article type:		Communicated by:	
	Regular paper		M. Kaykobad and R. Petreschi	

E-mail addresses: baruniitg123@gmail.com (Barun Gorain) psm@iitg.ernet.in (Partha Sarathi Mandal) krishnendu@isical.ac.in (Krishnendu Mukhopadhyaya)

1 Introduction

Covering vertices of a graph with subgraphs like trees, tours or paths is a widely studied topic of research [1, 7]. A collection of trees (tours/paths) is called a tree (tour/path) cover of a graph $G = (V, E)$ if every vertex of G appears in at least one of the trees (tours/paths). This problem has applications in vehicle routing [1, 9], where the objective is to serve a set of clients by assigning some vehicles with proper scheduling. The objectives of the problems vary depending on the applications. Two classes of problems are popular in literature. The Min-Max problems [1, 3, 7] aim to minimize the maximum weight of a tree in the tree cover where the maximum size of the tree cover is given. In the Mincover or Bounded cover problems [1, 7], the maximum weight of a tree is given. The objective of Mincover problem is to find a tree cover with minimum number of trees that covers all the vertices of the graph. Both the above tree cover problems are NP-hard [1]. In this paper, we concentrate on bounded tree cover problem.

Definition 1 (Bounded tree cover problem (BTC)[7]) *Let $G = (V, E, w)$ be an undirected graph with positive weights. For a given bound $\lambda \geq 0$, the objective is to find a tree cover of G with minimum number of trees such that the weight of each tree is at most λ.*

In many applications, multiple weight functions may be associated with a graph [10]. Consider an application, where geographic locations are represented as vertices of a graph. Between a pair of locations several attributes like distance, travel time and traveling cost etc. may be associated. These attributes can be represented as multiple weight functions of the graph. Some other problems may also be equivalently restated as BTC of a graph with multiple weight functions. For example, let us consider the following variation of BTC. Let $G = (V, E, w)$ be a weighted graph, λ a positive real number, and p a positive integer. The objective is to find a tree cover of G such that the weight of each tree is at most λ and each tree can have at most p vertices. This problem can be reformulated on G by assigning it another weight function w' where $w'(e) = 1$, for all $e \in E$. Now, the objective of the problem is to find a tree cover of G such that for each tree T of the tree cover, $w(T) \leq \lambda$ and $w'(T) \leq p - 1$.

Throughout this paper, we denote the vertex and edge set of a graph G by $V(G)$ and $E(G)$, respectively. For any subgraph H and a weight function w of G, $w(H)$ denotes the sum of the edge weights of H. For the first two problems, we consider the input graph G is a complete graph with metric weight functions. If G is not a complete graph, it can be transformed into its shortest path metric completion $\tilde{G}$. A tree cover of G can be constructed from a tree cover of $\tilde{G}$ by replacing each edge of a tree in $\tilde{G}$ with the shortest path between the corresponding vertices in G [7]. (v_i, v_j) denotes the edge between two vertices v_i and v_j. A path P of G is denoted by $v_1 v_2 \cdots v_j$, where $v_1, v_2, \cdots, v_j$ are consecutive vertices along P from v_1 to v_j. For any set S, $|S|$ denotes the number of elements in S.

A tour on a subtree of the graph is computed by doubling the edges of the subtree and shortcutting. In this method, we add a copy of each edge of the subtree to make it an Eulerian graph. Then, we select an Euler tour of the graph and shortcutting on this tour by removing repeated nodes, one at a time until all node repetitions are removed.

Related Work: Many interesting results have been reported in literature on tree cover problems. Evan et al. [3] considered the rooted version of Min-Max k-tree cover problem. In this problem a set of vertices called *roots* are given as input with the graph. The objective is to find a tree cover with k trees such that the weight of the maximum weighted tree is minimum. A 4-approximation algorithm is proposed to solve the problem. Nagamochi [8] proposed a $(3 - \frac{2}{k+1})$-approximation algorithm for the Min-Max k-tree cover problem where each tree of the tree cover has a common root. If the underlying graph is a tree, the authors proposed a $(2 + \epsilon)$-approximation algorithm for the rooted version and a $(2 - \frac{2}{k+1})$-approximation algorithm for the unrooted version of the Min-Max k-tree cover problem, respectively. The objective of the Min-Max tree partition problem is to partition the graph into k equal size vertex sets such that the maximum weight of the minimum spanning trees of each of the vertex sets is minimized. Guttmann-Beck et al. [6] proposed $(2k - 1)$-approximation algorithm to solve this problem for a graph with metric weight function. Frederickson et al. [4] considered the k-TSP problem, where the objective is to cover a graph with k tours rooted at a given vertex such that the total weight of the tours is minimized. An $(e + 1 - \frac{1}{k})$-approximation algorithm was proposed, where e is the approximation ratio for the TSP problem.

Arkin et al. [1] proposed a 3-approximation algorithm for BTC. The algorithm computes paths of bounded weight and the set of paths is returned as the set of trees of the tree cover. For each k, $1 \leq k \leq n$, minimum spanning forest with k connected components is computed. Then paths of desired weights are calculated from the tours which are computed on each of the components by doubling the edges and shortcutting. Shortcutting is a technique to compute a tour from an Eulerian tour by eliminating duplicate entries of all vertices except the first vertex. The minimum number of trees over all iterations is returned as the output of the algorithm. The authors also proposed a 4-approximation algorithm for Min-Max k-tree cover problem. Khani et al. [7] proposed a 2.5-approximation algorithm for BTC. The proposed algorithm joins the smaller trees to reduce the number of trees in the tree cover. The authors also proposed a 3-approximation algorithm for Min-Max k-tree cover problem. These are the best known constant factor approximation algorithms for the above two problems.

Our Contribution: In this paper several variations of BTC are considered. Strong tree cover and weak tree cover problems are introduced on a graph with multiple metric weight functions. In *strong tree cover*, every tree must be bounded with respect to all weight functions, whereas in *weak tree cover*, each tree must be bounded with respect to at least one weight function. A 4-approximation algorithm is proposed for strong tree cover problem. Some modifications of the same algorithm gives an approximation factor 3 for the corresponding path cover problem. For weak tree cover problem, an $O(\log n)$-approximation algorithm is proposed for a graph having two weight functions. The same algorithm is extended to work for arbitrary number of weight functions. An inapproximability result is established for BTC when weights of the trees of the tree cover are bounded by a set of given bounds which are not necessarily same. A constant factor approximation algorithm is proposed for the special case where the ratio of the maximum and minimum bound is bounded by a constant. Further, a 3-approximation algorithm is proposed for BTC for a graph with a general weight function which is not necessarily metric.

2 Tree Cover for Graphs with Multiple Weight Functions

We introduce strong tree cover and weak tree cover on a graph with multiple weight functions. In the strong tree cover, the trees in the tree cover need to be bounded with respect to each of the weight functions. In weak tree cover, the trees in the tree cover need to be bounded with respect to at least one of the weight functions. The formal definitions of the problems are given below.

Definition 2 (Strong tree cover problem) *Let G be a complete graph with multiple metric weight functions $w_1, w_2, \ldots, w_l$, $(l \geq 1)$. For given bounds λ_1, $\lambda_2, \cdots, \lambda_l \geq 0$, the objective is to find a tree cover with minimum number of trees such that for each tree T in the tree cover, $w_i(T) \leq \lambda_i$, for all $i = 1, 2, \cdots, l$.*

Definition 3 (Weak tree cover problem) *Let G be a complete graph with multiple metric weight functions $w_1, w_2, \cdots, w_l$ $(l \geq 1)$. For given bounds λ_1, $\lambda_2, \cdots, \lambda_l \geq 0$, the objective is to find a tree cover with minimum number of trees such that for each tree T in the tree cover, there exists a j, $1 \leq j \leq l$, with $w_j(T) \leq \lambda_j$.*

Both of the above problems are NP-hard, since for $l = 1$, the problems reduce to BTC, which is NP-hard [1].

2.1 Strong tree cover

In this section, a 4-approximation algorithm for strong tree cover problem is proposed. We define a weight function w' on $G = (V, E, w_1, w_2, \cdots, w_l)$ as follows. For any edge $e \in E(G)$,

$$w'(e) = \begin{cases} 1 & \text{if there exists a } j,\ 1 \leq j \leq l, \\ & \text{such that } \lambda_j = 0 \\ \min\left\{\max\{\frac{w_1(e)}{\lambda_1}, \frac{w_2(e)}{\lambda_2}, \cdots, \frac{w_l(e)}{\lambda_l}\}, 1\right\} & \text{otherwise.} \end{cases}$$

Note that w' is a metric as the maximum of two metric is a metric and minimum of a metric and a constant is also a metric.

Algorithm 1 (StrongTreeCover) is proposed to solve the strong tree cover problem on G. A formal description of Algorithm 1 is given below.

First, the minimum spanning tree Γ of G is computed with respect to w'. All the edges of Γ with weights more than one with respect to w' are deleted. Let C_1, C_2, $\cdots$ C_h be the connected components of γ after deletion of these edges. For $1 \leq i \leq h$, a tour τ_i is computed on C_i after doubling the edges of C_i and shortcutting. Let P_i be the path after deleting an arbitrary edge of τ_i. The path P_i is split into subpaths of weight atmost one with respect to w'. These subpaths are the trees of the tree cover SOL, returned by Algorithm 1.

Algorithm 1: StrongTreeCover(G)

```
1: $SOL = \{\}$.
2: Find the minimum spanning tree $\Gamma$ of $G$ with respect to $w'$.
3: Delete each edge $e$ from $\Gamma$ for which $w_j(e) > \lambda_j$, for some $j$.
   Let $C_1$, $C_2$, $\cdots$ $C_h$ be the connected components after deletion of the
   edges.
4: for $i = 1$ to $h$ do
5:     Find a tour $\tau_i$ on $C_i$ after doubling the edges and shortcutting.
       Let $P_i$ be the path after deleting an arbitrary edge from $\tau_i$.
6:     while $|V(P_i)| > 0$ do
7:         $P_i := v_i^1 v_i^2 \cdots, v_i^{|V(P_i)|}$.
8:         Let $v_i^j$ be the first vertex on $P_i$ such that $w'(v_i^1 v_i^2 \cdots v_i^{j+1}) > 1$.
9:         $SOL = SOL \bigcup (v_i^1 v_i^2 \cdots v_i^j)$ and $P_i = \Big(P_i \setminus (v_i^1 v_i^2 \cdots v_i^j)\Big) \setminus (v_i^j, v_i^{j+1})$.
10:    end while
11: end for
12: return $SOL$.
```

Now we analyze the approximation factor of Algorithm 1. Let *opt* be the number of trees in an optimal tree cover. The following lemma gives a lower bound on the optimal solution.

Lemma 1 *If Γ is a minimum spanning tree of G with respect to w' then $\lceil w'(\Gamma) \rceil \leq 2opt$.*

Proof: Let T_1, T_2, $\cdots$, T_{opt} be the trees in the optimal tree cover. Clearly, $w'(T_i) \leq 1$ for $i = 1, 2, \cdots, opt$. Let v_i be a vertex on T_i. Construct a spanning tree H of G by adding the set of edges $\{(v_i, v_{i+1}) | i = 1, 2, \cdots, opt - 1\}$ to $\bigcup_{i=1}^{opt} T_i$. Since $w'(v_i, v_{i+1}) \leq 1$,

$$w'(H) = \sum_{i=1}^{opt} w'(T_i) + \sum_{i=1}^{opt-1} w'(v_i, v_{i+1}) \leq opt + opt - 1 \leq 2opt - 1.$$

As Γ is a minimum spanning tree of G with respect to w' so $w'(\Gamma) \leq w'(H) \leq 2opt - 1$.
Hence $\lceil w'(\Gamma) \rceil \leq 2opt$. $\square$

Theorem 4 *The approximation factor of Algorithm 1 is 4.*

Proof: Let *opt* be the number of trees in an optimal tree cover and $|SOL|$ the number of trees calculated by Algorithm 1. Let $C_1, C_2, \cdots, C_h$ be the connected components after deleting $h-1$ edges in step 3 from the minimum spanning tree Γ. Then,

$$w'(\Gamma) = w'(C_1) + w'(C_2) + \cdots + w'(C_h) + h - 1 \tag{1}$$

Since τ_i is a tour found after doubling the edges in C_i and shortcutting, and w' is a metric, we have

$$w'(\tau_i) \leq 2w'(C_i) \tag{2}$$

According to step 9, each time a tree (which is basically a path) is computed from P_i, the weight of modified P_i is reduced by at least one with respect to w'. Thus the number of trees computed from τ_i can be at most $\lceil w'(\tau_i) \rceil$. If no edge is deleted from Γ in step 3, then $|SOL| \leq 2\lceil w'(\Gamma) \rceil \leq 4opt$. When $h \geq 2$, i.e., at least one edge is deleted from Γ in step 3,

$$\begin{aligned} |SOL| &\leq \sum_{i=1}^{h} \lceil w'(\tau_i) \rceil \\ &\leq \sum_{i=1}^{h} w'(\tau_i) + h \\ &\leq 2\sum_{i=1}^{h} w'(C_i) + h \quad (From\ Equation\ (2)) \\ &\leq 2w'(\Gamma) \quad (From\ Equation\ (1)) \\ &\leq 4opt \quad (From\ Lemma\ 1) \end{aligned}$$

Therefore, the approximation factor of Algorithm 1 is 4. $\square$

2.2 Weak tree cover

First we propose an algorithm to solve weak tree cover problem for a graph with two weight functions. Then we extend the algorithm to solve the problem for a graph with any given number of weight functions. We use the 2-approximation algorithm for k-MST [5] as a subroutine in the proposed algorithm. The definition of k-MST problem is given below.

Definition 5 (k-MST problem [5]) *Let $G = (V, E, w)$ be a weighted graph, where edge weights are positive real numbers and k a given positive integer. The objective is to find a minimum weighted tree of G that spans any k vertices of G.*

Let w_1 and w_2 be two metric weight functions on G. We define two weight functions w'_1 and w'_2 as follows:

$$w'_1(e) = \begin{cases} \frac{w_1(e)}{\lambda_1} & \text{if } w_1(e) \le \lambda_1, \\ 1 & \text{otherwise.} \end{cases}$$

$$w'_2(e) = \begin{cases} \frac{w_2(e)}{\lambda_2} & \text{if } w_2(e) \le \lambda_2, \\ 1 & \text{otherwise.} \end{cases}$$

Algorithm 3 (WeakTreeCover) is proposed to solve the weak tree cover problem on G. The algorithm calls the recursive Algorithm 2 (FindTree) as a subroutine. Algorithm (FindTree(G)) computes $\lceil \frac{n}{2} \rceil$-MST of G, Γ_1 with respect to w'_1 and $\lceil \frac{n}{2} \rceil$-MST of G, Γ_2 with respect to w'_2, respectively. If $w'_1(\Gamma_1) \le w'_2(\Gamma_2)$, then the algorithm returns $(\Gamma_1, 1)$ and recusrively call itself on the graph induced by the remaining vertices. Otherwise, it returns $(\Gamma_2, 2)$ and recusrively call itself on the graph induced by the remaining vertices. Hence, Algorithm 2 returns $O(\log n)$ subtrees, each of which is associated with an integer $i \in \{1, 2\}$. $(\Gamma_1, 1)$ means the subtree Γ_1 is calculated with respect to the weight functions w'_1. Similarly, $(\Gamma_2, 2)$ means the subtree Γ_2 is calculated with respect to the weight functions w'_2. For each pair (Γ', i) returned by FindTree, each edge e with $w_i(e) > \lambda_i$ is deleted from Γ'. This deletion may split Γ' into several connected components. For each component, a tour is computed by doubling the edges and shortcutting. Then paths of weights at most λ_i with respect to w_i are calculated from the tour. Then the set of paths is returned by the algorithm as the tree cover of G.

Algorithm 2: FindTree(G)

```
1: $n' := |V(G)|$.
2: **if** $n' = 1$ **then**
3:     Return $(G, 1)$.
4: **end if**
5: Find a $\left\lceil \frac{n'}{2} \right\rceil$-MST $\Gamma_1$ of $G$ with respect to $w'_1$ using the 2-approximation
   algorithm [5]. Let $G_1$ be the induced subgraph with the remaining
   vertices.
6: Find a $\left\lceil \frac{n'}{2} \right\rceil$-MST $\Gamma_2$ of $G$ with respect to $w'_2$ using the 2-approximation
   algorithm [5]. Let $G_2$ be the induced subgraph with the remaining
   vertices.
7: **if** $w'_1(\Gamma_1) \le w'_2(\Gamma_2)$ **then**
8:     Return $(\Gamma_1, 1) \bigcup$ FindTree($G_1$).
9: **else**
10:    Return $(\Gamma_2, 2) \bigcup$ FindTree($G_2$).
11: **end if**
```

Lemma 2 *According to Algorithm 3, if* $(\Gamma', i) \in S$, *then* $w'_i(\Gamma') \leq 4opt'$, *where* opt' *is the number of trees in the optimal tree cover of* G.

Proof: Since (Γ', i) is returned by FindTree, there exists a subgraph G' of G such that Γ' is a subtree spanning $\left\lceil \frac{|V(G')|}{2} \right\rceil$ vertices of G'. Let $opt_{G'}$ be the number of trees in the optimal tree cover of G'. Let $T_1, T_2, \cdots, T_{opt_1}, T_{opt_1+1}, T_{opt_1+2}, \cdots T_{opt_{G'}}$ be the trees in the optimal tree cover of G', where $w_1(T_i) \leq \lambda_1$ for $i = 1, 2, \cdots, opt_1$ and $w_2(T_i) \leq \lambda_2$ for $i = opt_1+1, opt_1+2, \cdots, opt_{G'}$. Let $v_i \in V(T_i)$. Construct two subgraphs H_1 and H_2 as follows:

$$\begin{aligned} H_1 &= \left(\bigcup_{i=1}^{opt_1} T_i\right) \bigcup \{(v_i, v_{i+1}) | i = 1, 2, \cdots, opt_1 - 1\} \\ H_2 &= \left(\bigcup_{i=opt_1+1}^{opt_{G'}} T_i\right) \bigcup \{(v_i, v_{i+1}) | i = opt_1 + 1, opt_1 + 2, \cdots, opt_{G'} - 1\} \end{aligned}$$

Now,

$$\begin{aligned} w'_1(H_1) &= \sum_{i=1}^{opt_1} w'_1(T_i) + \sum_{i=1}^{opt_1-1} w'_1(v_i, v_{i+1}) \\ &\leq 2opt_1 - 1 \end{aligned}$$

Therefore, $\lceil w'_1(H_1) \rceil \leq 2opt_1 \leq 2opt_{G'}$. Similarly, $\lceil w'_2(H_2) \rceil \leq 2opt_{G'}$.

Since $\{T_1, T_2, \cdots, T_{opt_1}, T_{opt_1+1}, T_{opt_1+2}, \cdots, T_{opt_{G'}}\}$ is a tree cover of G', either $|V(H_1)| \geq \left\lceil \frac{|V(G')|}{2} \right\rceil$ or $|V(H_2)| \geq \left\lceil \frac{|V(G')|}{2} \right\rceil$.

Without loss of generality let $|V(H_1)| \geq \left\lceil \frac{|V(G')|}{2} \right\rceil$. For $j = 1, 2$, let Γ_j^{opt} be the optimal $\left\lceil \frac{|V(G')|}{2} \right\rceil$-MST of G' with respect to w'_j. Since H_1 is a spanning tree that spans at least $\left\lceil \frac{|V(G')|}{2} \right\rceil$ vertices of G', therefore,

$$\lceil w'_1(\Gamma_1^{opt}) \rceil \leq \lceil w'_1(H_1) \rceil \leq 2opt_{G'}$$

Let Γ_1 and Γ_2 be the $\left\lceil \frac{|V(G')|}{2} \right\rceil$-MST of G' with respect to w'_1 and w'_2, respectively computed using 2-approximation algorithm [5] (Ref. step 5 and step 6 of FindTree). Then, $w'_1(\Gamma_1) \leq 2w'_1(\Gamma_1^{opt})$.

Therefore,

$$\lceil w'_i(\Gamma') \rceil = \min\{\lceil w_1(\Gamma_1) \rceil, \lceil w_2(\Gamma_2) \rceil\} \leq 2 \left\lceil w'_1(\Gamma_1^{opt}) \right\rceil \leq 4opt_{G'}$$

Since $opt_{G'} \leq opt'$, therefore $\lceil w'_i(\Gamma') \rceil \leq 4opt'$. □

Theorem 6 *The approximation factor of Algorithm 3 is* $O(\log n)$, *where* n *is the number of vertices of* G.

Algorithm 3: WeakTreeCover(G, w_1, w_2)

```
1: S=FindTree($G$).
2: $SOL'$={}.
3: for each $(\Gamma', p) \in S$ do
4:     Delete every edge $e$ from $\Gamma'$ for which $w_p(e) > \lambda_p$.
5:     Let $C_1$, $C_2$, $\cdots$ $C_h$ be the connected components after deletion of the
       edges.
6:     for $i = 1$ to $h$ do
7:         Find a tour $\tau_i$ from $C_i$ after doubling the edges and shortcutting.
8:         Let $P_i$ be the path after deleting any edge from $\tau_i$.
9:         while $|V(P_i)| > 0$ do
10:            $P_i := v_i^1 v_i^2 \cdots v_i^{|V(P_i)|}$.
11:            Let $v_i^j$ be the first vertex on $P_i$ such that $w_p(v_i^1 v_i^2 \cdots v_i^{j+1}) > \lambda_p$.
12:            $SOL' = SOL' \bigcup (v_i^1 v_i^2 \cdots v_i^j)$ and
               $P_i = \Big(P_i \setminus (v_i^1 v_i^2 \cdots v_i^j)\Big) \setminus (v_i^j, v_i^{j+1})$.
13:        end while
14:    end for
15: end for
16: Return $SOL'$.
```

Proof: Let $(\Gamma', i) \in S$. By Lemma 2, $\lceil w_i'(\Gamma') \rceil \leq 4opt'$. In step 5 through step 14 of Algorithm 3, a set of trees are computed from Γ'. Let the total number of trees computed from Γ' be N_1. If no edge from Γ' is deleted in step 5 of Algorithm 3, i.e., $h = 1$ then $N_1 \leq 2\left\lceil \frac{w_i(\Gamma')}{\lambda_i} \right\rceil = \lceil w_i'(\Gamma') \rceil \leq 4opt'$.

Consider the case when at least one edge is deleted from Γ', i.e., $h \geq 2$. After the deletion of $h-1$ edges, Γ' splits into h connected components $C_1, C_2, \cdots, C_h$. The corresponding tours $\tau_1, \tau_2, \cdots, \tau_h$ are computed after doubling the edges of each component and shortcutting.

Therefore, $N_1 \leq \sum_{j=1}^{h} \lceil w_i'(\tau_j) \rceil \leq \sum_{j=1}^{h} 2w_i'(C_j) + h$.

Also, $w_i'(\Gamma') = \sum_{j=1}^{h} w_i'(C_j) + h - 1$. Therefore,

$$\begin{aligned} N_1 &\leq \sum_{j=1}^{h} 2w_i'(C_j) + h \\ &\leq 2w_i'(\Gamma') \\ &\leq 8opt' \end{aligned}$$

Since the total number of trees returned by FindTree is $O(\log n)$, therefore $|SOL'| \leq O(\log n) \cdot 8opt'$.

Hence, the approximation factor of Algorithm 3 is $O(\log n)$.

□

Algorithm 3 can be extended to work for a graph with arbitrary number of weight functions. Let $G = (V, E, w_1, w_2, \cdots, w_l)$ be the given graph with bounds λ_1, λ_2,$\cdots$, λ_l. For each i, $1 \leq i \leq l$, we define w'_i as follows:

$$w'_i(e) = \begin{cases} \frac{w_i(e)}{\lambda_i} & \text{if } w_i(e) \leq \lambda_i, \\ 1 & \text{otherwise.} \end{cases}$$

Following modification on FindTree subroutine is made for the extended version of Algorithm 3. The spanning trees Γ_1, Γ_2, $\cdots$, Γ_l are computed with respect to w'_1, w'_2, $\cdots$, w'_l, respectively such that each Γ_i spans any $\left\lceil \frac{|V(G)|}{l} \right\rceil$ vertices of G. FindTree returns $(\Gamma_j, j) \bigcup$ FindTree(G_j) where j $(1 \leq j \leq l)$ is the index such that $w'_j(\Gamma_j) = \min\{w'_i(\Gamma_i) | i = 1, 2, \cdots, l\}$.

Since in each call of FindTree, the number of vertices of the graph is reduced by a fixed fraction of $\frac{1}{l}$, FindTree returns $O(\log n)$ number of subtrees in S. Using arguments similar to those used in Theorem 6, one can show that the approximation factor of this modified Algorithm 3 is $O(\log n)$.

2.3 Path cover problem

Similar problem like strong and weak tree cover can be defined to cover a graph with multiple weight functions by paths with bounded weights. As the proposed algorithms for both strong and weak tree cover problems return a set of paths as the tree covers, the same algorithms can be applied to get a solution for corresponding path cover problems. Also, it can be proved that Algorithm 1 and Algorithm 3 yield approximation factors 4 and $O(\log n)$, respectively. The reason for getting the same approximation factor is that Lemma 1 holds for the strong path cover problem. The above discussion can be summarized by the following theorem.

Theorem 7 *There is a 4-approximation algorithm for strong path cover problem and a $O(\log n)$-approximation algorithm for weak path cover problem.*

With a small modification on Algorithm 1, the approximation factor for strong path cover problem can be improved. In step 2 of Algorithm 1, a TSP tour L using Christofides $\frac{3}{2}$-approximation algorithm [2] is computed instead of an MST Γ. Then each edge e with $w'(e) \geq 1$ is deleted from L and L may be decomposed into a set of paths $P_1, P_2, \cdots P_h$ (say) after deletion of such edges. The rest of the steps are same as Algorithm 1 which compute a set of paths from $P_1, P_2, \cdots P_h$ and return the set of paths as the set of trees in the tree cover.

The above technique for the strong path cover problem gives an approximation factor 3. As Christofides algorithm [2] is used to compute L, therefore, $w'(L) \leq \frac{3}{2} w'(L_{opt})$, where L_{opt} is the optimal TSP tour on G with respect to w'. With the same arguments as given in Lemma 1, it can be proved that $w'(L_{opt}) \leq 2opt$. Therefore, $w'(L) \leq 3opt$. With the similar steps of calculation as Theorem 4, we have,

$$\begin{aligned}|SOL| &\leq \sum_{i=1}^{h} \lceil w'(P_i) \rceil \\ &\leq \sum_{i=1}^{h} w'(P_i) + h \\ &\leq w'(L) \\ &\leq 3opt\end{aligned}$$

Therefore, the above modified technique gives an approximation factor 3 for the strong path cover problem.

3 Tree cover with different bounds

In some practical applications it may be required that the trees of the tree cover are bounded by different limits. For example, consider a service provider having a set of vehicles with different speed limits. Vehicle V_i can travel maximum distance D_i in time t. With these vehicles the service provider can provide service to a set of customers who are located at different locations. Suppose there are multiple service requests from different customers at a time instance for providing service within time t. Here the goal of the service provider is to schedule minimum number of vehicles such that all the customers will be served within that time. The problem can be formulated as BTC with different bounds. The definition of tree cover problem with different bounds (TCDB) is given below.

Definition 8 (TCDB) *Let $G = (V, E, w)$ be a weighted graph and $\lambda_1, \lambda_2, \cdots, \lambda_n \geq 0$ given real numbers. The objective is to find a tree cover $\{T_{i_1}, T_{i_2}, \cdots, T_{i_p}\}$ with minimum number of trees such that $w(T_{i_1}) \leq \lambda_{i_1}$, $w(T_{i_2}) \leq \lambda_{i_2}$, $\cdots$, $w(T_{i_p}) \leq \lambda_{i_p}$, where $i_j \in \{1, 2, \cdots, n\}$ and $i_j \neq i_q$ for $j \neq q$ for $j, q = 1, 2, \cdots, p$.*

We prove that there does not exist any constant factor approximation algorithm for TCDB unless P=NP. Arkin et al. [1] proved the hardness of the following decision version of BTC.

Definition 9 (k-BTC [1]) *Given a graph $G = (V, E, w)$, a real number $\lambda \geq 0$ and a positive integer k, whether there is a tree cover of G with k trees such that the weight of each tree is at most λ.*

Arkin et al. [1] proved that k-BTC is NP-hard for $k = \frac{|V|}{3}$. Guttmann-Beck et al. [6] proved hardness of the following decision version of the Min-Max tree partitioning problem.

Definition 10 (Min-Max tree partitioning problem [6]) *Given a graph $G = (V, E, w)$ and a real number $\lambda \geq 0$, whether G can be partitioned into two trees T_1 and T_2 such that $V(T_1) = V(T_2) = \frac{|V|}{2}$ and $w(T_1) \leq \lambda$, $w(T_2) \leq \lambda$.*

It can be shown that k-BTC is NP-hard even for $k = 2$ using the same reduction technique which is used to prove the hardness of the Min-Max tree partitioning problem in [6].

Theorem 11 *There does not exist any constant factor approximation algorithm for TCDB, unless P=NP.*

Proof: We propose a polynomial time reduction from an instance of 2-BTC to an instance of TCDB. We show that if there exist a constant factor approximation algorithm for TCDB then 2-BTC can be decided in polynomial time. If possible, suppose there exists a z-approximation algorithm $\mathcal{A}$ for TCDB. Without loss of generality we assume that z is a positive integer. We consider an instance (G_1, λ) of 2-BTC , where $G_1 = (V_1, E_1, w_1)$ is a complete weighted graph with $n > 2z$ vertices. The weight w_1 of each edge is integer and satisfies triangle inequality. We construct a complete graph $G_2 = (V_2, E_2, w_2)$ with n^2 vertices from G_1 as follows. For each vertex $v_i \in V_1$, we consider n vertices $v_i^1, v_i^2, \cdots v_i^n$ in V_2. Weight w_2 of an edge $(v_i^l, v_j^k) \in E_2$ is defined as $w_2(v_i^l, v_j^k) = w_1(v_i, v_j)$ for $i \neq j$ and $w_2(v_i^l, v_i^k) = \frac{1}{2n(n-1)}$.

We show that G_1 has a tree cover with two trees having weights at most λ iff G_2 has a tree cover with two trees having weights at most $\lambda + \frac{1}{2}$. Let $\{T_1, T_2\}$ be a tree cover of G_1 with $w_1(T_1) \leq \lambda$ and $w_1(T_2) \leq \lambda$. Let $v_1, v_2, \cdots, v_q$ be the vertices of $V(T_1)$. We compute a subtree of T_1' of G_2 as follows. For every edge $(v_x, v_y) \in E(T_1)$, one edge (v_x^1, v_y^1) is added in T_1'. Then the edges $\{v_f^i, v_f^{i+1} \mid i = 1 \text{ to } n-1,\ v_f \in V(T_1)\}$ are added in T_1'. Similarly, T_2' can be computed from T_2. It can be easily verified that $\{T_1', T_2'\}$ is a tree cover of G_2 with $w_2(T_1') \leq \lambda + \frac{1}{2}$ and $w_2(T_2') \leq \lambda + \frac{1}{2}$.

Conversely, let $\{T_1', T_2'\}$ be a tree cover of G_2 with $w_2(T_1') \leq \lambda + \frac{1}{2}$ and $w_2(T_2') \leq \lambda + \frac{1}{2}$. A subtree T_1 of G_1 from T_1' is constructed as follows. First we remove all the edges of type (v_i^k, v_i^l) from T_1'. The remaining edges of T_1' are of the form (v_i^k, v_j^l) for $i \neq j$. For each and every such edge (v_i^k, v_j^l) of T_1' we consider the corresponding edge (v_i, v_j) in G_1 and construct a graph $\mathcal{G}$. Then the minimum spanning tree T_1 of $\mathcal{G}$ is computed. Note that $w_1(T_1) \leq \lambda + \frac{1}{2}$. Since the edge weights of G_1 are integer, $w_1(T_1) \leq \lambda$. Similarly, T_2 can be constructed from T_2'.

We consider an instance of TCDB as G_2 with n^2 bounds $\{\lambda + \frac{1}{2}, \lambda + \frac{1}{2}, 0 \cdots, 0\}$. Let y be the number of trees in the tree cover returned by algorithm $\mathcal{A}$ on G_2.

Case 1: $(y \leq 2z)$ We show that there is a tree cover of G_1 with two trees having weights at most λ. In this case, among these y trees, $y - 2$ are singleton vertices. The other two trees of weight $\lambda + \frac{1}{2}$ cover at least $n^2 - 2z + 2$ vertices of G_2. Since $n^2 - 2z + 2 > n^2 - n$, at least one vertex from each of the set $\{v_i^1, v_i^2, \cdots, v_i^n\}$ of G_2 must be covered by at least one of the trees having weights at most $\lambda + \frac{1}{2}$. From these two trees, two subtrees of G_1 with weight at most λ can be computed in the similar way as explained in the previous paragraph. Hence, G_1 has a tree cover with two trees having weights at most λ.

Case 2: $(y > 2z)$ As $\mathcal{A}$ is a z-approximation algorithm, we can say that there is no tree cover of G_2 with two trees having weight at most $\lambda + \frac{1}{2}$. This implies that there doest not exist any tree cover of G_1 with two trees having weight at most λ.

Hence, 2-BTC can be decided in polynomial time by applying $\mathcal{A}$ on G_2, which is a contradiction unless P=NP. Hence the statement of the theorem follows. □

3.1 Constant factor approximation algorithm for a special case of TCDB

To prove the inapproximability of TCDB, we reduce an instance of 2-BTC to a particular instance of TCDB, where ratio of the maximum bound and the minimum bound is unbounded. We show that if the ratio of the maximum and minimum bound is bounded by a constant, then a constant factor approximation algorithm for TCDB can be designed.

Let $G = (V, E, w)$ be a weighted graph and $\lambda_1, \lambda_2, \cdots, \lambda_n \geq 0$ be the given bounds for TCDB such that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n$. We consider the special case of TCDB where ratio of the λ_1 and λ_n is bounded by a constant ρ, i.e., $\frac{\lambda_1}{\lambda_n} \leq \rho$. We propose Algorithm 4 (SplTCDB(G)) which uses the algorithm *Path_Min_k* [1] for bounded tree cover as subroutine. First *Path_Min_k* [1] is applied to find a tree cover S of G with bound λ_1. If $\rho|S| \geq n$, then Algorithm 4 returns all the singleton vertices of G as trees of the tree cover, otherwise, it splits each subtree of S into trees with weights at least λ_n.

Algorithm 4: SplTCDB(G)

```
1: Apply Path_Min_k [1] to find a tree cover $S$ of $G$ with bound $\lambda_1$.
2: if $\rho|S| \geq n$ then
3:   return $\{\{v\} \mid v \in V(G)\}$.
4: else
5:   Let $S = \{P_1, P_2, \cdots, P_a\}$.
6:   $p = 2$.
7:   for $i = 2$ to $a$ do
8:     while $|V(P_i)| > 0$ do
9:       $P_i := v_i^1 v_i^2 \cdots v_i^{|V(P_i)|}$.
10:      Let $v_i^j$ be the first vertex on $P_i$ such that $w(v_i^1 v_i^2 \cdots v_i^{j+1}) > \lambda_p$.
11:      $SOL' = SOL' \bigcup (v_i^1 v_i^2 \cdots v_i^j)$ and
         $P_i = \left(P_i \setminus (v_i^1 v_i^2 \cdots v_i^j)\right) \setminus (v_i^j, v_i^{j+1})$. $p = p + 1$.
12:    end while
13:  end for
14: end if
15: return $SOL$.
```

Theorem 12 *Algorithm 4 is a $3(\rho + 1)$-approximation algorithm.*

Proof: Let opt be the number of trees in the optimal solution of TCDB, and opt_1 be the number of trees in the optimal solution for the bounded tree cover problem with bound λ_1. Then $opt_1 \leq opt$. Since *Path_Min_k* [1] is a 3-approximation algorithm, therefore, $|S| \leq 3opt_1$. Each subtree of S is split into at most $\lceil \frac{\lambda_1}{\lambda_n} \rceil$ trees of weight at least λ_n.
Therefore, $|SOL'| \leq \lceil \frac{\lambda_1}{\lambda_n} \rceil |S| \leq 3(\rho + 1) \cdot opt_1 \leq 3(\rho + 1) \cdot opt$. □

4 Tree Cover for General Weighted Graph

The existing works in literature [1, 7] discussed BTC on a graph with metric weight function. In this section, we consider the tree cover problem for a weighted graph with arbitrary positive weight function. Let $G = (V, E, w)$ be a weighted graph where $w : E \to \mathbb{R}^+$ is an arbitrary weight function. Let λ be the upper bound of the weights of the trees in the tree cover. The following lemma [7] proposed by Kani et al. is useful in our context.

Lemma 3 *([7]) Let T be any tree with weight $w(T)$ such that for each edge e of $E(T)$, $w(e) \leq \beta$. Then T can be decomposed into subtrees T_1, T_2, $\cdots$, T_k where $k \leq \max\{\lceil \frac{w(T)}{\beta} \rceil, 1\}$ such that $w(T_i) \leq 2\beta$ for each $1 \leq i \leq k$.*

The idea of splitting a tree into subtrees is explained by Evan et al. in [3]. Note that Lemma 3 holds for any graph with a weight function which is not necessarily a metric. We define a weight function $w_{\frac{1}{2}}$ as follows:

$$w_{\frac{1}{2}}(e) = \begin{cases} \frac{2w(e)}{\lambda} & \text{if } w(e) \leq \frac{\lambda}{2}, \\ 1 & \text{otherwise.} \end{cases}$$

Lemma 4 *Let opt be the number of trees in the optimal tree cover of G and Γ be the minimum spanning tree of G with respect to $w_{\frac{1}{2}}(e)$. Then $w_{\frac{1}{2}}(\Gamma) \leq 3opt - 1$.*

Proof: Let T_1, T_2, $\cdots$, T_{opt} be the trees in the optimal tree cover of G. Then $w_{\frac{1}{2}}(T_i) \leq 2$, for each i, $1 \leq i \leq opt$. Let v_i be a vertex of T_i. Let $H = (\bigcup_{i=1}^{opt} T_i) \bigcup (\bigcup_{i=1}^{opt-1} \{v_i, v_{i+1}\})$. Clearly, H is a spanning tree of G and $w_{\frac{1}{2}}(H) \leq \sum_{i=1}^{opt} w_{\frac{1}{2}}(T_i) + \sum_{i=1}^{opt-1} w_{\frac{1}{2}}(v_i, v_{i+1}) \leq 2opt + opt - 1 = 3opt - 1$. □

Now we are going to describe our algorithm for finding a tree cover of G. First, we find an MST Γ of G with respect to $w_{\frac{1}{2}}$. Then each edge $e \in \Gamma$ for which $w_{\frac{1}{2}}(e) > 1$ are deleted from Γ. After deletion of such edges, let Γ splits into h subtrees Γ_1, Γ_2, $\cdots$, Γ_h. For $i = 1$ to h, a set of subtrees S_i are computed from Γ_i with weight at most 2 using the splitting strategy proposed in [3]. Finally, $SOL = \bigcup_{i=1}^{h} S_i$ is returned as the tree cover of G.

Theorem 13 *The above technique is a 3-approximation algorithm.*

Proof: Let $T' \in S_i$. Then $w_{\frac{1}{2}}(T') \leq 2$ and this implies that $w(T) \leq \lambda$. According to Lemma 3, $|S_i| \leq \max\{\lceil \frac{2w(\Gamma_i)}{\lambda} \rceil, 1\} \leq \max\{\lceil w_{\frac{1}{2}}(\Gamma_i) \rceil, 1\} \leq w_{\frac{1}{2}}(\Gamma_i) + 1$. Also, $w_{\frac{1}{2}}(\Gamma) = \sum_{i=1}^{h} w_{\frac{1}{2}}(\Gamma_i) + h - 1$.

Therefore, total number of trees in the tree cover is given by

$$\begin{aligned}
|SOL| &\leq \sum_{i=1}^{h} (w_{\frac{1}{2}}(\Gamma_i) + 1) \\
&\leq \sum_{i=1}^{h} w_{\frac{1}{2}}(\Gamma_i) + h \\
&\leq w_{\frac{1}{2}}(\Gamma) + 1 \\
&\leq 3opt
\end{aligned}$$

Therefore, the proposed technique is a 3-approximation algorithm. □

5 Conclusion

Some variations of bounded tree cover problem have been discussed in this paper. We have introduced strong tree cover and weak tree cover for graphs with multiple metric weight functions. A 4-approximation algorithm is proposed for strong tree cover and an $O(\log n)$-approximation algorithm is proposed for weak tree cover. An inapproximability result is established for bounded tree cover problem where the trees in the tree cover are bounded by a set of given bounds which are not necessarily same. We have proposed a 3-approximation algorithm for BTC for a graph with general weight function.

Designing constant factor approximation algorithm for the weak tree cover may be an interesting open problem to be investigated in future.

References

[1] E. M. Arkin, R. Hassin, and A. Levin. Approximations for minimum and min-max vehicle routing problems. *J. Algorithms*, 59(1):1–18, 2006. doi:10.1016/j.jalgor.2005.01.007.

[2] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[3] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Min-max tree covers of graphs. *Oper. Res. Lett.*, 32(4):309–315, 2004. doi:10.1016/j.orl.2003.11.010.

[4] G. N. Frederickson, M. S. Hecht, and C. E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978. doi:10.1137/0207017.

[5] N. Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In H. N. Gabow and R. Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 396–402. ACM, 2005. doi:10.1145/1060590.1060650.

[6] N. Guttmann-Beck and R. Hassin. Approximation algorithms for min-max tree partition. *J. Algorithms*, 24(2):266–286, 1997. doi:10.1006/jagm.1996.0848.

[7] M. R. Khani and M. R. Salavatipour. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. *Algorithmica*, 69(2):443–460, 2014. doi:10.1007/s00453-012-9740-5.

[8] H. Nagamochi. Approximating the minmax rooted-subtree cover problem. *IEICE Transactions*, 88-A(5):1335–1338, 2005. doi:10.1093/ietfec/e88-a.5.1335.

[9] V. Nagarajan and R. Ravi. Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214, 2012. doi:10.1002/net.20435.

[10] M. Rocklin and A. Pinar. On clustering on graphs with multiple edge types. *Internet Mathematics*, 9(1):82–112, 2013. doi:10.1080/15427951.2012.678191.