Graph Covering Using Bounded Size Subgraphs

Barun Gorain, Shaswati Patra, and Rishi Ranjan Singh(✉)

Indian Institute of Technology Bhilai, Sejbahar, Raipur, India
{barun,shaswatip,rishi}@iitbhilai.ac.in

Abstract. A variant of graph covering problem demands to find a set of sub-graphs when the union of sub-graphs contain all the edges of G. Another variant of graph covering problem requires finding a collection of subgraphs such that the union of the vertices of subgraphs forms a vertex cover. We study the later version of the graph covering problem. The objective of these problems is to minimize the size/cost of the collection of subgraphs. Covering graphs with the help of a set of edges, set of vertices, tree or tour has been studied extensively in the past few decades. In this paper, we study a variant of the graph covering problem using two special subgraphs. The first problem is called *bounded component forest cover* problem. The objective is to find a collection of minimum number of edge-disjoint bounded weight trees such that the vertices of the forest, i.e., collection of edge-disjoint trees, cover the graph. The second problem is called *bounded size walk cover* problem. It asks to minimize the number of bounded size walks which can cover the graph. Walks allow repetition of vertices/edges. Both problems are a generalization of classical vertex cover problem, therefore, are NP-hard. We give 4ρ and 6ρ factor approximation algorithm for bounded component forest cover and bounded size walk cover problems respectively, where ρ is an approximation factor to find a solution to the tree cover problem.

Keywords: Graph Covering · Vertex Cover Problem · Tree Cover Problem · Approximation Algorithm

1 Introduction

A set of vertices are said to cover an edge if at least one end vertex of that edge is present in that set of vertices. Graph covering problems aim to find a subset of graph vertices such that all edges are covered by that subset while minimizing some objective function. The classical vertex cover problem is a graph covering problem which requires finding a minimum size/cost subset of graph vertices that covers all the edges of a given graph. The problem of covering graphs with specific subgraphs is studied by several researchers in the past four decades [1–3,5]. The objective in such problems is to determine an optimal size/cost collection of subgraphs of a graph such that the union of vertices of the subgraphs covers

A. Bagchi and R. Muthu (Eds.): CALDAM 2023, LNCS 13947, pp. 415–426, 2023.
https://doi.org/10.1007/978-3-031-25211-2_32

all edges of the original graph. Based on the topology of subgraphs, several variations of the problem are defined. As most of these variations are NP-hard, the main goal is to design efficient approximation schemes.

In this paper, we aim to study two variants of graph covering with bounded size subgraphs. The problems aim to cover the graph with a minimum number of subgraphs each of whose weight is bounded by a given real number. Formally, let $G = (V, E, w)$ be a weighted graph where $w : E \rightarrow R^+$. A forest cover of G is a collection of disjoint trees $\{T_1, T_2, \cdots, T_j\}$ such that the union of the vertices in all the trees in the collection will be a vertex cover. Note that disjoint trees do not have any common vertices/edges. The cardinality of a forest cover is j, the number of trees in the forest cover. We define a problem named *bounded component forest cover* (BCFC) problem as follows.

Definition 1. *For a given weighted graph $G = (V, E, w)$, and a non-negative real number λ, find a forest cover of minimum cardinality such that the weight of each tree in the forest cover is at most λ.*

Note that when $\lambda = 0$, the problem is reduced to the minimum vertex cover problem. Hence, we have the following result.

Theorem 1. *The BCFC problem over (G, λ) is NP-hard.*

The second problem is motivated by a real-life application problem monitoring a large art gallery. A guard can see the entire corridor from one of its end junction points. The objective is to place a minimum number of mobile guards in such a way that every corridor can be under the scrutiny of at least one guard in t time period, for a given time $t > 0$. If every guard moves with a constant average velocity v, then the movement routes of the guards decompose the graph into subgraphs, each of which has a length at most vt, and the set of vertices covered by the guards must form a vertex cover. In this case, each subgraph is a walk of length at most vt, and the walks in the solution may be intersecting, i.e., may have common edges/vertices.

Formally, let $G = (V, E)$ be a weighted graph with the weight function $w : E \rightarrow R^+$. A walk cover of G is a collection of walks $\{P_1, P_2, \cdots, P_j\}$ which are allowed to intersect, i.e., may have common edges/vertices such that the union of the vertices on all the walks in the collection, forms a vertex cover. The cardinality of a walk cover is j, the number of walks in the walk cover. Analogous to BCFC, we define a problem named *bounded size walk cover* (BSWC) as follows.

Definition 2. *For a given weighted graph $G = (V, E, w)$, and a non-negative real number λ, find a walk cover of minimum cardinality such that the weight of each walk in the walk cover is at most λ.*

If $\lambda = 0$, BSWC problem is reduced to the minimum vertex cover problem. Therefore, the following result holds.

Theorem 2. *The BSWC problem over (G, λ) is NP-hard.*

2 Related Work

In this section, we briefly mention works related to BCFC, BSWC and graph covering problems. The tree(tour) cover problem was first defined by Arkin et al. [1] in 1993. The tree(tour) cover problem of an edge weighted graph deals with finding a minimum weight tree(tour) in the graph such that the vertices of the tree(tour) are the vertices of some vertex cover of the graph. These two problems are NP-hard as an instance of vertex cover problem, and traveling salesperson problem [6] can be reduced to an instance of tree and tour cover problem, respectively. Arkin et al. have designed a 3.55 and 5.5-factor approximation algorithm for tree cover and tour cover problems, respectively. In [2,3], researchers have studied the tree cover problem and proposed improved approximation algorithms. Koneman et al. [2] gave a linear programming formulation for the tree cover problem and derived a 3-factor rounding algorithm. Fujito [3] gave a 2-factor approximation algorithm to find a minimum tree cover. Viet Hung Nguyen [4] established a 3.5 approximation factor for the tour cover using a compact linear program which is weaker as compared to 3-factor proposed by Konemann et al. [2]. Researchers have studied a similar problem called edge dominating set problem [5,7,8] that finds a subset of edges E_1 in a graph $G = (V, E)$ such that for each edge not in E_1 has at least one common end vertex with some edges of E_1. It is a minimization problem.. This problem is a special case of BSWC problem when $\lambda = 1$ and the graph is unweighted. Researchers [5,7,8] have proposed various approximation algorithms to solve the edge dominating set problem and the best-known algorithm has an approximation factor 2 [5]. Fujito and Nagamochi [5] and Parekh [9] have proposed 2-factor approximation algorithms to find minimum vertex cover, minimum edge dominating set, and some related problems. Monien and Speckeumeyer [11] have established an approximation factor ≤ 1.8 for finding a minimum vertex cover in all graphs with ≤ 146000 nodes. To find a minimum vertex cover in graphs authors [12,13] have proposed different approximation algorithms whose approximation factors are lesser than 2. In [14], authors have proved that it is NP-hard to establish an approximation factor lesser than 1.36067 for a vertex cover problem in a graph.

The problem of graph covering using walks is related to a well-studied problem of graph exploration by mobile agents. If the mobile agents have to monitor the edges of the network by visiting at least one of its end vertices, they have to visit walks containing all vertices of some vertex cover of the graph. The optimal number of agents required for edge exploration is a related problem to BSWC; therefore, we briefly mention a few results on edge exploration. In a graph exploration problem single or multiple mobile agents have to visit the nodes or edges of a graph. Many research works are concerned with the exploration of the graph by a single mobile agent, as discussed in [15–18,20]. In [21], authors have assumed that in the deterministic exploration of the graph by multiple agents, the movement of the agents are coordinated centrally. In [22], authors have designed different approximation algorithms for the collective exploration of an arbitrary graph by a group of mobile agents. In [19], authors have studied the problem graph exploration where starting from a node, a mobile agent has to

visit all the vertices of an anonymous graph where the nodes do not have ids, but the edges incident on a node are labeled with port numbers. Dhar et al. [23,24] studied the edge exploration of an anonymous graph by a mobile agent.

3 Results

In this section, we present constant factor approximation schemes for both considered problems.

3.1 Constant Factor Approximation Algorithm for BCFC

Recall a tree cover problem in an edge weighted graph deals with finding a minimum weight tree such that the vertices of the tree form a vertex cover of the graph. First, we show that a constant factor approximation algorithm for the tree cover problem can be used to design a constant factor approximation algorithm to BCFC. The general idea is to find a tree cover of a given graph and then split the tree into bounded size components such that all vertices of the tree cover are preserved in the process of splitting. The resulting forest is a solution of BCFC problem on the given graph. The tree cover problem is NP-Hard; therefore, the proposed scheme obtains an approximated tree cover solution using some ρ-factor approximation algorithm. The following lemma from [25] helps us to find a solution of BCFC from a given solution of tree cover problem.

Lemma 1 ([25]). *Let $\beta > 0$ be a positive real number and let T be any tree with vertex set V_T and the edge set E_T. If for each $e \in E_T$, $w(e) \leq \beta$, then T can be split into sub-trees $\zeta_1, \zeta_2, \cdots, \zeta_k$ where $k \leq \max\{\lceil \frac{w(T)}{\beta} \rceil, 1\}$ such that $w(\zeta_i) \leq 2\beta$ for each $1 \leq i \leq k$.*

The procedure of how to split the tree into sub-trees is explained in [26].

Let $G = (V, E, w)$ be a given weighted graph. We define a weight function $w_{\frac{1}{2}}$ as follows:

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

Let $G' = (V, E, w_{\frac{1}{2}})$ where $G = (V, E, w)$. Let $w_{\frac{1}{2}}(X)$ denote the sum of the weights of the edges in a subgraph X of graph G'. Similarly, $w(X)$ is defined for a subgraph X of graph G. The following lemma establishes a relationship between the optimal tree cover of G' and the number of trees in the optimal solution of BCFC.

Lemma 2. *Let OPT_{BCFC} be the number of sub-trees in the optimal solution of BCFC problem over (G, λ) and let OPT_{TC} be the optimal tree cover of G'. Then $w_{\frac{1}{2}}(OPT_{TC}) \leq 4OPT_{BCFC} - 2$.*

Proof. Let $\xi_1, \xi_2, \cdots, \xi_{OPT_{BCFC}}$ be the trees in an optimal solution of BCFC for (G, λ). Then by the definition of $w_{\frac{1}{2}}$, $w_{\frac{1}{2}}(\xi_i) \leq 2$, for each i, $1 \leq i \leq OPT_{BCFC}$. Construct a graph $H = (V_H, E_H)$ with OPT_{BCFC} many vertices as follows.

For every tree ξ_i, take a vertex u_i in V_H. Add an edge $(u_i, u_j) \in E_H$, if there exists a vertex $v_i \in V_{\xi_i}$ and there exist a vertex $v_j \in V_{\xi_j}$ such that v_i and v_j are connected by a path with at most two edges in G. Assign $w(u_i, u_j) = \min\limits_{\{P_{xy} | x \in V_{\xi_i}, y \in V_{\xi_j}\}} \{w(P_{xy})\}$, where P_{xy} is a path between x and y with at most two edges in G. Since G is connected, and the vertices of OPT_{BCFC} forms a vertex cover, therefore the graph H is also connected. Let τ be the minimum spanning tree of H with respect to w and E_τ be the set of edges in the τ. Note that for every edge in $e \in E_\tau$, $w_{\frac{1}{2}}(e) \leq 2$, as there can be at most two edges in G corresponding to one edge in τ and the weight of an edge in G with respect to $w_{\frac{1}{2}}$ is at most 1. Let $Z = (\bigcup\limits_{i=1}^{OPT_{BCFC}} \xi_i) \bigcup \tau$. Clearly, Z is a tree cover of G and $w_{\frac{1}{2}}(Z) \leq \sum_{i=1}^{OPT_{BCFC}} w_{\frac{1}{2}}(\xi_i) + \sum_{e \in E_\tau} w_{\frac{1}{2}}(e)$. Recall, $w_{\frac{1}{2}}(\xi_i) \leq 2$ and $|E_\tau| = |V_\tau| - 1 = |V_H| - 1 = OPT_{BCFC} - 1$, therefore, we have $w_{\frac{1}{2}}(Z) \leq 2OPT_{BCFC} + 2(OPT_{BCFC} - 1) = 4OPT_{BCFC} - 2$.

Since, OPT_{TC} is an optimal tree cover of G', we have $w_{\frac{1}{2}}(OPT_{TC}) \leq w_{\frac{1}{2}}(Z) \leq 4OPT_{BCFC} - 2$ □

Next, we describe our approach to find a solution for BCFC problem over (G, λ). Let $\mathcal{A}$ be an approximation algorithm with ρ-factor approximation guarantee for the tree cover problem. Let APX_{TC} be the tree cover returned by $\mathcal{A}$ for the input graph $G' = (v, E, w_{\frac{1}{2}})$. First, we obtain an approximated tree cover APX_{TC} of G'. Then each edge e in APX_{TC} for which $w(e) > \frac{\lambda}{2}$ is deleted from APX_{TC}. After deletion of such edges, let APX_{TC} splits into h sub-trees χ_1, χ_2, $\cdots$, χ_h. For $i = 1$ to h, a set of sub-trees S_i is computed from χ_i using the tree splitting strategy proposed in [26] such that weight of each sub-tree in S_i has weight at most 2. Finally, forest cover $APX_{BCFC} = \bigcup_{i=1}^{h} S_i$ is returned as the solution to the BCFC problem.

Theorem 3. *Let $|APX_{BCFC}|$ be the number of trees in the forest cover APX_{BCFC}, then $|APX_{BCFC}| \leq 4 \cdot \rho \cdot OPT_{BCFC}$, when we have a ρ-factor approximation algorithm for the tree cover problem.*

Proof. Let T be a sub-tree in a set of sub-trees $S_i \subseteq APX_{BCFC}$. Then, $w_{\frac{1}{2}}(T) \leq 2$. Furthermore, for each edge $e \in T$, $w(e) \leq \frac{\lambda}{2}$ and $w(T) \leq \lambda$. According to Lemma 1, the number of trees in the set of sub-trees S_i, $|S_i| \leq \max\{\lceil w_{\frac{1}{2}}(\chi_i) \rceil, 1\} = \max\{\lceil \frac{2w(\chi_i)}{\lambda} \rceil, 1\} \leq w_{\frac{1}{2}}(\chi_i) + 1$. Recall, $APX_{BCFC} = \bigcup_{i=1}^{h} S_i$, therefore, $|APX_{BCFC}| = \sum_{i=1}^{h} |S_i| \leq \sum_{i=1}^{h} (w_{\frac{1}{2}}(\chi_i) + 1) = \sum_{i=1}^{h} w_{\frac{1}{2}}(\chi_i) + h$. Note that, $w_{\frac{1}{2}}(APX_{TC}) = \sum_{i=1}^{h} w_{\frac{1}{2}}(\chi_i) + h - 1$, therefore, we have $APX_{BCFC} \leq w_{\frac{1}{2}}(APX_{TC}) + 1 \leq \rho \cdot w_{\frac{1}{2}}(OPT_{TC}) + 1$. Using Lemma 2, we have $APX_{BCFC} \leq 4\rho \cdot OPT_{BCFC}$. □

Theorem 4 ([3]). *There exist a* 2 *factor approximation algorithm for tree cover problem.*

In view of Theorem 3 and Theorem 4 we have the final result in this subsection.

Theorem 5. *There exists an 8-factor approximation algorithm for BCFC.*

3.2 Constant Factor Approximation Algorithm for BSWC

Recall that a solution to BCFC can be easily converted to a solution of BSWC by doubling the edges in each component of the forest cover, breaking the tour into two bounded size walks, which may intersect. Let $|APX_{BCFC}|$ be the number of trees in the forest cover APX_{BCFC}, which is a solution of BCFC given by the approximation algorithm given in the above section. Then, after doubling the trees in APX_{BCFC} and cutting the formed tour due to doubling into two walks, we would have $2|APX_{BCFC}|$ walks. Therefore, the number of walks would be less than or equal to $8 \cdot \rho \cdot OPT_{BCFC}$ when we have a ρ factor approximation algorithm for the tree cover problem.

In this section, we show that an approximation scheme for the tree cover problem can be used to design a constant factor approximation scheme for BSWC. The general idea is to find a tree cover of a given graph and then delete high-cost edges. This process may result in a forest. The edges in the resulting forest are doubled to form tours over vertices in all respective components of the forest. Splitting these tours into bounded-size walks results in a collection of walks which is a feasible solution for BSWC. We prove that this approximation approach guarantees to give $6 \cdot \rho \cdot OPT_{BSWC}$ solution for BSWC problem.

Let $G = (V, E, w)$ be a weighted undirected graph, where every edge $e \in E$ has a positive real weight. We define a weight function w' on the graph G such that for each edge $e \in G$, $w'(e) = \frac{w(e)}{\lambda}$ if $w(e) \leq \lambda$ else $w'(e) = 1$. Let $G' = (V, E, w')$ where $G = (V, E, w)$. Let λ be a non-negative real number. Note that two walks may intersect and may have common vertices/edges. A set of walks $\{P_1, P_2, \cdots, P_j\}$, such that each walk is of weight at most λ, is called bounded size walk cover if union of vertices in all the walks forms a vertex cover of G. For any real $\lambda \geq 0$, the objective of BSWC problems is to find the minimum cardinality walk cover of G such that the weight of each walk in the walk cover is at most λ.

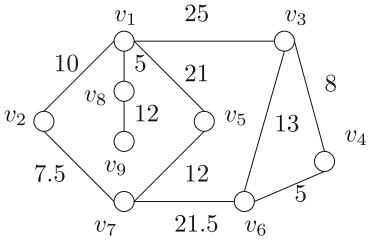
The above problem is NP-hard. To solve this problem, we design an approximation algorithm that finds a tree cover of the graph and splits the tree cover into sub-trees of smaller size by deleting high-weight edges, but all the vertices of the tree cover must be present in the sub-trees. Deletion of high-weight edges is a classical mechanism to break a tree in problems that have bounded size constraints [26]. The proposed algorithm finds walk cover from a tree cover following the idea of constructing sub-trees from a minimum spanning tree given in the Algorithm 1 in the paper [27]. We have modified Algorithm 1 from [27] according to our requirement to result walks which may intersect. For the sake

Algorithm 1: Bounded Size Walk Cover Algorithm

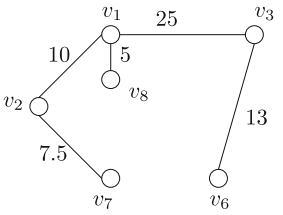
Find an approximate tree cover APX_{TC} in G' using ρ-approximation tree cover algorithm .
From the tree cover APX_{TC} delete each edge e with cost $w(e) \geq \lambda$. Let k be the number of edges deleted from APX_{TC}. It splits APX_{TC} into $k+1$ sub-trees denoted as $T_0, T_1, \cdots, T_k$.
for $i = 0$ *to* k **do**
    Find a tour ET_i on T_i by doubling the edges.
    Delete an arbitrary edge from ET_i, to get a path C_i.
end
Define $APX_{BSWC} = \emptyset$.
for $i = 0$ *to* k **do**
    while $w(C_i) > \lambda$ **do**
        Let $C_i = u_i^1 u_i^2 \cdots u_i^{|V(C_i)|}\}$.
        Let u_i^j be the first vertex on C_i such that $w(u_i^1 \cdots u_i^{j+1}) > \lambda$.
        $APX_{BSWC} = APX_{BSWC} \bigcup (u_i^1 \cdots u_i^j)$, $C_i = C_i \setminus (u_i^1 \cdots u_i^j u_i^{j+1})$.
        Delete all edges of the path $(u_i^1 \cdots u_i^j u_i^{j+1})$ from C_i. To delete the path $(u_i^1 \cdots u_i^j u_i^{j+1})$, we delete the vertices $\{u_i^1, \cdots, u_i^j\}$ and the edges $\{(u_i^1, u_i^2), \cdots, (u_i^{j-1}, u_i^j)\}$ from C_i.
    end
    $APX_{BSWC} = APX_{BSWC} \bigcup C_i$
end
Return APX_{BSWC}.

of completeness, the modified algorithm is summarized as Algorithm 1 in this paper and its working procedure is explained as follows. The algorithm finds an approximated tree cover APX_{TC} in the graph G using some ρ-factor approximation algorithm. It deletes all heavy edges with weight more than λ from APX_{TC}. Let k number of edges are deleted which splits APX_{TC} into $k+1$ different components $T_0, \cdots, T_k$. In each T_i, $i = 0, \cdots, k$, the algorithm finds a tour ET_i by doubling edges of T_i and at the end, it deletes one arbitrary edge from ET_i to get a walk C_i. Note that an edge may appear more than one time in such tour. For $i = 1, \cdots, k$, each walk C_i, is split into sub-walks of weight less than or equal to λ and those sub-walks are added to the solution APX_{BSWC}. Let the walk C_i is represented as a sequence of vertices $(u_i^1 u_i^2 \cdots u_i^{|V(C_i)|})$. In the walk C_i let u_i^j be the first vertex such that $w(u_i^1 \cdots u_i^{j+1}) > \lambda$, then it adds the walk $(u_i^1 \cdots u_i^j)$ to the set APX_{BSWC} and deletes the walk $(u_i^1 \cdots u_i^j u_i^{j+1})$ from C_i. To delete any walk $(u_i^1 \cdots u_i^j u_i^{j+1})$ from C_i , the algorithm deletes the vertices $\{u_i^1 \cdots u_i^j u_i^j\}$ and the edges $\{(u_i^1, u_i^2) \cdots (u_i^j, u_i^{j+1})\}$. It continues this process until $w(C_i) \leq \lambda$. Finally, we add the truncated walk C_i to the set APX_{BSWC}.

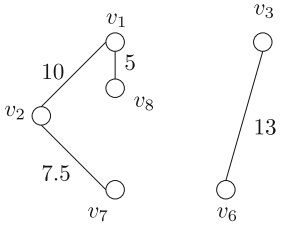
The execution steps of the Algorithm 1 are depicted with the help of an example, as shown in Fig. 1. Let $G = (V, E, w)$ be a positive edge-weighted graph as shown in Fig. 1a and let the bound on the weight of each walk be $\lambda = 20$. The algorithm first computes an approximate tree cover APX_{TC} in the graph G using



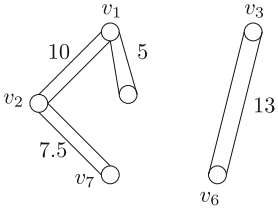
(a) A weighted Graph G



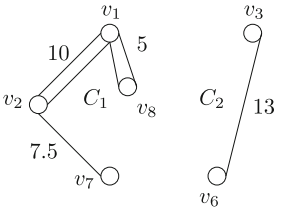
(b) A Tour cover APX_{TC} of G



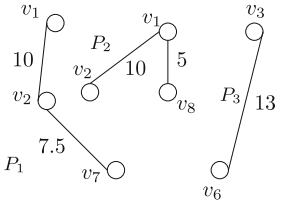
(c) Sub-trees T_1 and T_2 of APX_{TC}



(d) Tours ET_1 and ET_2



(e) Paths C_1 and C_2



(f) Paths P_1, P_2, P_3 of length at most 20.

Fig. 1. Example: Construction of bounded size walk cover.

some existing tree cover algorithm. Let Fig. 1b be the approximated tree cover APX_{TC}. Then, the algorithm deletes the edge (v_1, v_3) whose weight is greater than $\lambda = 20$ from APX_{TC}. Deletion of (v_1, v_3) splits APX_{TC} into two different sub-trees T_1 and T_2 as shown in Fig. 1c. Next, the algorithm doubles the edges of sub-trees T_1 and T_2 and finds tours ET_1 and ET_2, respectively, as depicted in Fig. 1d. From each tour ET_1 and ET_2, the algorithm deletes an arbitrary edge and gets open walks C_1 and C_2 as shown in Fig. 1e. In this example (v_2, v_7) and (v_3, v_6) are deleted from ET_1 and ET_2 respectively. Note that an edge may occur twice in these walks due to doubling. In the walk C_1, the algorithm starts from node v_7 and visits up to the node v_1. Since the walk $P_1 = (v_7, v_2, v_1)$ is the largest visited walk with weight at most 20, it adds the sub-walk $P_1 = (v_7, v_2, v_1)$ into the solution and deletes the walk P_1 along with the edge (v_1, v_8) from C_1. Then it finds the sub-walk $P_2 = (v_8, v_1, v_2)$ from the remaining walk of P_1. Similarly,

it finds sub-walk P_3 from the walk C_2. All the walks have a weight at most 20 as shown in Fig. 1f. Note that P_1 and P_2 are intersecting and have a common edge (v_1, v_2).

The following Lemmas and Theorem give correctness and derive the approximation factor of Algorithm 1. Recall APX_{BSWC} is the output of Algorithm 1, which is a set of walks.

Lemma 3. *APX_{BSWC} is a bounded size walk cover of graph G.*

Proof. The walks in APX_{BSWC} are constructed by deleting some edges of a tree cover of the graph G. So the walks in APX_{BSWC} include all the vertices of the tree cover of G. Therefore, the vertices of all the walks in APX_{BSWC} still form a vertex cover of graph G. According to Algorithm 1, the weight of each walk in APX_{BSWC} are bounded to be less or equal to λ. Hence, APX_{BSWC} is a feasible solution to BSWC problem in the graph G. □

To establish the approximation factor of Algorithm 1, we define certain variables. Let OPT_{BSWC} be the minimum number of bounded size walks, which forms the optimal solution of $BSWC$ problem over a graph G. Let OPT_{TC} be the optimal tree cover of the graph G'. We establish a relation between OPT_{BSWC} and $w'(OPT_{TC})$.

Lemma 4. $w'(OPT_{TC}) \leq 3.OPT_{BSWC} - 2$

Proof. Let $\{Q_1, Q_2, \cdots, Q_{OPT_{BSWC}}\}$ be the set of walks which forms the optimal solution of $BSWC$ as shown in Fig. 2. Weight of each Q_i is less than or equal to λ, i.e. $w(Q_i) \leq \lambda$ and $w'(Q_i) \leq 1$. We construct a graph $H = (V_H, E_H)$, similarly to how we constructed a graph H in the proof of Lemma 2. The graph H contains all walks $Q_1, \cdots, Q_{OPT_{BSWC}}$ as a subgraph and contains a few extra edges/vertices from G to connect these walks into a single connected component. In the graph G, if two walks Q_i and Q_j have common vertices or edges, then join them into a single component by taking the union of those walks so that each edge/vertex appears exactly once. We start with graph G, and then, we contract each component from the previous step, which is a subgraph of G, into a single vertex by contracting all the edges and respective vertices to obtain a graph $G_c = (V_c, E_c)$. Afterward, we find a minimum spanning tree MST T on the contracted graph G_c. Graph H is constructed from MST T by reversing the contraction of the components. The graph H is a sub-graph of G containing all the vertices of walks in the set $\{Q_1, Q_2, \cdots, Q_{OPT_{BSWC}}\}$. The sub-graph H is a tree cover of G as depicted in Fig. 3.

Fig. 2. $Q_1, \cdots, Q_{OPT_{BSWC}}$ be the walks in the optimal solution of $BSWC$

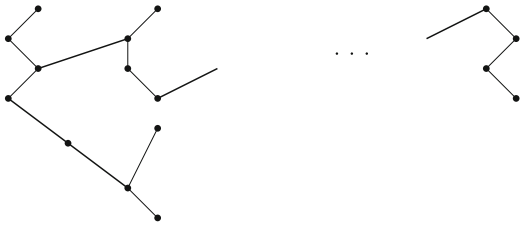


Fig. 3. Sub-graph H of G formed by edges of Q_i and MST T on G_c

In the graph G_c, consider two nodes u_x and u_y that represent two components that may be formed by union of few walks from the set $\{Q_1, Q_2, \cdots, Q_{OPT_{BSWC}}\}$. Let u_x and u_y are connected in G_c through a path p_{xy}, which is also present in MST T. Then, the number of edges on path p_{xy} is at most two. Otherwise, if three edges are present in p_{xy}, then any end vertex of middle edge can not be covered by any vertex of Q_i, for $1 \leq i \leq OPT_{BSWC}$, and hence the covering of all edges is not guaranteed. Therefore, weight of each such path p_{xy} in G', $w'(p_{xy}) \leq 2$. Note that the vertices H form a vertex cover of G. As per the construction, the graph H contains all walks from the set $\{Q_1, Q_2, \cdots, Q_{OPT_{BSWC}}\}$ and at most $OPT_{BSWC} - 1$ many paths (of p_{xy} type) to connect all walks. The weight of each walk Q_i in G' is $w'(Q_i) \leq 1$. Hence, the weight of sub-graph H in G' is given as $w'(H) \leq OPT_{BSWC} + 2.(OPT_{BSWC} - 1) \leq 3OPT_{BSWC} - 2$. Since H is also a tree cover of the graph G, weight of the optimal tree cover in G', $w'(OPT_{TC}) \leq w'(H) \leq 3OPT_{BSWC} - 2$. □

Theorem 6. *Let $Y = |APX_{BSWC}|$ be the number of walks of a bounded weight in the set APX_{BSWC} resulted by Algorithm 1. Then, $Y \leq 6 \cdot \rho \cdot OPT_{BSWC}$, when we have a ρ-factor approximation algorithm for the tree cover problem.*

Proof. The proposed algorithm obtains an approximate tree cover APX_{TC} of G' using some ρ-factor approximation algorithm, i.e. weight of APX_{TC} in G' is $w'(APX_{TC}) \leq \rho.w'(OPT_{TC})$, where OPT_{TC} is the optimal tree cover of G'. The algorithm deletes the edges from APX_{TC} whose weight is greater than λ in G. After deletion of heavy edges, let APX_{TC} be split into $\{T_1, T_2, ..., T_m\}$ sub-trees. The algorithm doubles all edges in each sub-tree for a set of sub-tours $\{ET_1, ET_2, ..., ET_m\}$. After the deletion of an arbitrary edge from each sub-tour, the algorithm finds open walks $\{C_1, C_2, ..., C_m\}$. Each walk C_i is then splitted into bounded size sub-walks which are kept in the solution as walks that may intersect. Note that $Y \leq \sum_{i=1}^{m} \lceil w'(C_i) \rceil \leq \sum_{i=1}^{m} \frac{w(C_i)}{\lambda} + m$. We have $w'(APX_{TC}) = \sum_{i=1}^{m} w'(T_i) + m - 1$ which can be rewritten as $2w'(APX_{TC}) \geq \sum_{i=1}^{m} \frac{w(C_i)}{\lambda} + 2m - 2$. Hence $Y \leq 2w'(APX_{TC}) - m + 2 \leq 2\rho \cdot w'(OPT_{TC}) - m + 2$. Using Lemma 4, we have $Y \leq 6 \cdot \rho \cdot OPT_{BSWC}$, as $\rho > 1$. □

In view of Theorem 6 and Theorem 4, we have the final result in this sub-section.

Theorem 7. *There exists a 12-factor approximation algorithm for BSWC.*

4 Conclusion and Future Work

In this paper, we have studied two graph covering problems: bounded component forest cover (BCFC) problem and bounded size walk cover (BSWC) problem. The problems are NP-hard due to a trivial reduction to the classical vertex cover problem when the bound on weight is 0. We designed $4.\rho$ factor approximation algorithm for the bounded component forest cover problem, where ρ is the approximation factor for finding a solution of tree cover problem. We further give a $6.\rho$ factor approximation algorithm for bounded size walk cover problem. Using 2-factor approximation algorithm given by Fujito [3] for tree cover problem, we have 8-factor and 12-factor approximation algorithm for BCFC and BSWC respectively.

Reducing these approximation factors is the first obvious direction to work on. One possible such improvement may be due to starting of with a subgraph other than a solution to the tree cover problem, which may bring down the final approximations factors. Studying bounded size path cover and bounded size intersecting sub-tour cover are other alternatives that we plan to look in future. Another future direction is to study the graph covering problem using other types of bounded size subgraphs.

Acknowledgements. The authors would like to thank the referees for the helpful comments. Barun Gorain and Rishi Ranjan Singh acknowledges the support of the Research Initiation Grant awarded by IIT Bhilai, India. Barun Gorain acknowledges the support of the Science and Engineering Research Board (SERB), Department of Science and Technology, Govt. of India (Grant Number: CRG/2020/-005964 and Grant Number: MTR/2021/000118).

References

1. Arkin, E.M., Halldorsson, M.M., Hassin, R.: Approximating the tree and tour covers of a graph. Inf. Process. Lett. **47**(6), 275–282 (1993)
2. Könemann, J., Konjevod, G., Parekh, O., Sinha, A.: Improved approximations for tour and tree covers. Algorithmica **38**(3), 441–449 (2004)
3. Fujito, T.: How to trim a MST: a 2-approximation algorithm for minimum cost-tree cover. ACM Trans. Algorithms (TALG) **8**(2), 1–11 (2012)
4. Nguyen, V.H.: Approximating the minimum tour cover with a compact linear program. In: van Do, T., Thi, H.A.L., Nguyen, N.T. (eds.) Advanced Computational Methods for Knowledge Engineering. AISC, vol. 282, pp. 99–104. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06569-4_7
5. Fujito, T., Nagamochi, H.: A 2-approximation algorithm for the minimum weight edge dominating set problem. Discrete Appl. Math. **118**(3), 199–207 (2002)
6. Christofides, N.: Worst-case analysis of a new heuristic for the traveling salesman problem. Technical report, GSIA, Carnegie Mellon University (1976)
7. Carr, R., Fujito, T., Konjevod, G., Parekh, O.: A 2 1/10-approximation algorithm for a generalization of the weighted edge-dominating set problem. In: Paterson, M.S. (ed.) ESA 2000. LNCS, vol. 1879, pp. 132–142. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45253-2_13
8. Fujito, T.: On approximability of the independent/connected edge dominating set problems. Inf. Process. Lett. **79**(6), 261–266 (2001)

9. Parekh, O.: Edge dominating and hypomatchable sets. In: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 287–291 (2002)
10. Edmonds, J., Johnson, E.L.: Matching: a well-solved class of integer linear programs. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) Combinatorial Optimization — Eureka, You Shrink! LNCS, vol. 2570, pp. 27–30. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36478-1_3
11. Monien, B., Speckenmeyer, E.: Ramsey numbers and an approximation algorithm for the vertex cover problem. Acta Informatica **22**(1), 115–123 (1985)
12. Halperin, E.: Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. SIAM J. Comput. **31**(5), 1608–1623 (2002)
13. Karakostas, G.: A better approximation ratio for the vertex cover problem. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1043–1050. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_84
14. Dinur, I., Safra, S.: The importance of being biased. In: Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, pp. 33–42 (2002)
15. Bender, M. A., Fernández, A., Ron, D., Sahai, A., Vadhan, S.: The power of a pebble: exploring and mapping directed graphs. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 269–278 (1998)
16. Fleischer, R., Trippen, G.: Exploring an unknown graph efficiently. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 11–22. Springer, Heidelberg (2005). https://doi.org/10.1007/11561071_4
17. Panaite, P., Pelc, A.: Exploring unknown undirected graphs. J. Algorithms **33**(2), 281–295 (1999)
18. Duncan, C.A., Kobourov, S.G., Kumar, V.A.: Optimal constrained graph exploration. ACM Trans. Algorithms (TALG) **2**(3), 380–402 (2006)
19. Gorain, B., Pelc, A.: Deterministic Graph Exploration with Advice, ACM Trans. Algorithms **15**, 8:1–8:17 (2018)
20. Diks, K., Fraigniaud, P., Kranakis, E., Pelc, A.: Tree exploration with little memory. J. Algorithms **51**(1), 38–63 (2004)
21. Bender, M.A., Slonim, D.K.: The power of team exploration: two robots can learn unlabeled directed graphs. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 75–85. IEEE (1994)
22. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: 17th Annual Symposium on Foundations of Computer Science (SFCS 1976), pp. 216–227. IEEE (1976)
23. Dhar, A.K., Gorain, B., Mondal, K., Patra, S., Singh, R.R.: Edge exploration of a graph by mobile agent. In: Li, Y., Cardei, M., Huang, Y. (eds.) COCOA 2019. LNCS, vol. 11949, pp. 142–154. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36412-0_12
24. Dhar, A.K., Gorain, B., Mondal, K., Patra, S., Singh, R.R.: Edge exploration of anonymous graph by mobile agent with external help. Computing (2022). https://doi.org/10.1007/s00607-022-01136-8
25. Khani, M.R., Salavatipour, M.R.: Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. Algorithmica **69**(2), 443–460 (2014)
26. Even, G., Garg, N., Könemann, J., Ravi, R., Sinha, A.: Min-max tree covers of graphs. Oper. Res. Lett. **32**(4), 309–315 (2004)
27. Gorain, B., Mandal, P.S., Mukhopadhyaya, K.: Generalized bounded tree cover of a graph. J. Graph Algorithms Appl. **21**(3), 265–280 (2017)