J Glob Optim (2012) 53:563–572
DOI 10.1007/s10898-011-9726-x

Complexity and approximation of the connected set-cover problem

Wei Zhang · Weili Wu · Wonjun Lee · Ding-Zhu Du

Received: 11 April 2011 / Accepted: 7 May 2011 / Published online: 20 May 2011

Abstract In this paper, we study the computational complexity and approximation complexity of the connected set-cover problem. We derive necessary and sufficient conditions for the connected set-cover problem to have a polynomial-time algorithm. We also present a sufficient condition for the existence of a $(1 + \ln \delta)$-approximation. In addition, one such $(1 + \ln \delta)$-approximation algorithm for this problem is proposed. Furthermore, it is proved that there is no polynomial-time $O(\log^{2-\varepsilon} n)$-approximation for any $\varepsilon > 0$ for the connected set-cover problem on general graphs, unless NP has an quasi-polynomial Las-Vegas algorithm.

Keywords Connected set-cover · Computational complexity · Approximation algorithms

1 Introduction

In this paper, we study the computational complexity and approximation complexity of the connected set-cover (CSC) problem. Firstly, we introduce some related definitions.

W. Zhang (✉)
Department of Applied Mathematics, Xi'an Jiaotong University, Xi'an 710049, Shaanxi, People's Republic of China
e-mail: dc.zhangwei@gmail.com

W. Wu
Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083, USA
e-mail: weiliwu@utdallas.edu

W. Lee
Department of Computer Science and Engineering, Korea University, Seoul 136-713, Republic of Korea
e-mail: wlee@korea.ac.kr

D.-Z. Du
Department of Computer Science, University of Texas at Dallas, USA and WCU FNOT Research Center at Korea University, Seoul, Republic of Korea
e-mail: dzdu@utdallas.edu

Definition 1 *Set-cover*: Given a set system $(V, \mathcal{S})$, where V is a set of elements and $\mathcal{S}$ is a family of n subsets of V such that $\cup_{S\in\mathcal{S}} S = V$, a set-cover C of V is a subfamily of $\mathcal{S}$ such that each element in V is in at least one of the subsets in C.

Let G be a connected graph with $\mathcal{S}$ as the vertex set. Here each node in graph G is labeled with a subset in $\mathcal{S}$, so we use the terminology 'set' and 'vertex' (in G) interchangeably in the rest of the paper. A connected set-cover with respect to $(V, \mathcal{S}, G)$ can be defined as follows.

Definition 2 *Connected set-cover*: We call $C \subseteq \mathcal{S}$ a connected set-cover if C is a set-cover of V and C induces a connected subgraph of G.

The CSC problem is to find the minimum connected set-cover for given $(V, \mathcal{S}, G)$. In [1], Shuai and Hu studied the computational complexity of the problem in several simple graphs such as line graph, ring graph and spider graph. In [2], Gupta et al. studied the CSC problem under the distributed applications, for which they proposed two approximation algorithms. In this paper, we further study the computational complexity and approximation complexity of the CSC problem. Our results are organized as follows. In Sect. 2, we give necessary and sufficient conditions for the CSC problem to have a polynomial-time algorithm. In Sect. 3, we propose a condition for the CSC problem to have a $(1 + \ln \delta)$-approximation algorithm. One such $(1 + \ln \delta)$-approximation algorithm is proposed. In Sect. 4, we show that for the CSC problem in general graphs, there is no polynomial-time $O(\log^{2-\varepsilon} n)$-approximation for any $\varepsilon > 0$ unless NP has a quasi-polynomial Las-Vegas algorithm.

2 Polynomial-time computable condition

The CSC problem $(V, \mathcal{S}, G)$ is solvable in polynomial time in some simple cases. For example in [1], Shuai and Hu have shown that there are polynomial-time algorithms for the CSC problem on line graphs and ring graphs, while on star graphs and spider graphs, the problem is NP-hard. In this section, we further study the computational complexity of the CSC problem. We derive necessary and sufficient conditions for the problem to have a polynomial-time algorithm.

Lemma 1 *Given a graph G and a positive integer k, if every spanning tree of graph G has at most k leaves, then the maximum degree of nodes in G is at most k.*

Proof Assume by contradiction that there is a node $v \in G$ with degree m, where $m > k$.

Consider any spanning tree T of G with v as the root node. If all the neighbors of v in graph G are also directly connected to v in T, then T has m different subtrees rooted at v. So T has at least m leaves, which conflicts with the assumption that every spanning tree of graph G has at most k leaves. Or else some neighbor node of v in G is no longer the neighbor of v in T. Suppose a node u is connected to v in graph G via edge (u, v), and this edge does not appear in T as shown in Fig. 1. Then u must be connected to some other node, denoted by w. Now we add edge (u, v) into T and delete (u, w) from T. Then we get a new tree which is also a spanning tree of graph G. After performing this transformation for all such neighbor nodes of v, we get a new spanning tree T' in which node v has degree m. Then T' has at least m leaves, which leads to a contradiction. □

We call any node that has degree more than two a 3^+ node. We construct a new graph $G^3 = (V^3, E^3)$ as follows. Form V^3 by all the 3^+ nodes and leaf nodes of G. For each pair

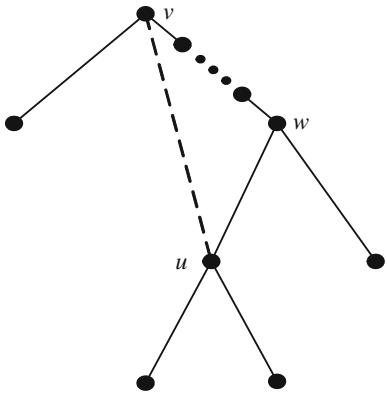

Fig. 1 Spanning tree transformation

of nodes in V^3, if there is an edge between them in G, then there is still an edge between them in G^3; else if there is a path formed by nodes of degree two connecting them, we add an edge to connect them in G^3. The following lemma shows that for a graph G, if the number of the leaves of its spanning trees can be bounded, then the number of 3^+ nodes in it can also be bounded.

Lemma 2 *Given a graph G and a positive integer k, if every spanning tree of graph G has at most k leaves, then there are at most $5k-5$ 3^+ nodes in graph G.*

Proof Suppose among all the spanning trees of G that have k leaves, T is the one that has the minimum total depth. Then via T, we can reveal some basic structural properties of graph G as follows.

1. There are at most $k-2$ 3^+ nodes in tree T. Or else there will be more than k leaves in T, which conflicts with the assumption. All these 3^+ nodes are also 3^+ nodes in graph G.
2. The child nodes of each 3^+ node in T can be 3^+ nodes in graph G. In graph G, these nodes can have edges with each other or some nodes that are 3^+ nodes in T. The number of such nodes equals to the number of paths formed by nodes of degree two in tree T, which is at most $2k-3$.
3. The k leaf nodes of tree T can be 3^+ nodes in graph G. For a leaf node i, suppose i is in level $l(i)$ of T. Due to the assumption that T has the minimum total depth, i can only be connected with nodes of level no smaller than $l(i)-1$. Also notice that i can only be connected to at most one node that has a parent of degree two. Or else we link such two nodes to the leaf. This will change the leaf node into an interior node but will also lead to two new leaves. Then the total number of leaves in the new spanning tree will be $k+1$.
4. A non-leaf node v that has a parent of degree two can not be connected to other interior nodes in graph G. If so, we can change the tree such that the parent of this non-leaf node becomes a new leaf. This will increase the number of the leaves in T. However, v can be connected with some leaf nodes. Because each leaf node can only be connected with one such node as v, there are at most k such nodes as v that can be linked to leaf nodes. These nodes can be 3^+ nodes in graph G.
5. The rest nodes of T have degrees one or two in graph G.

Based on the above analysis, there are at most $5k-5$ 3^+ nodes in graph G. □

For graph G which satisfies the property that all of its spanning trees have no more than k leaves, we propose an algorithm for the CSC problem as follows.

Algorithm 1 Polynomial-time algorithm for the CSC problem

Input: A instance of the CSC problem $(V, \mathcal{S}, G)$.
Output: The minimum connected set-cover.
1: **for** each maximal path (i, j) formed by nodes of degree two in G **do**
2: Compute the minimum connected set-cover on (i, j) through exhaustive search;
3: **end for**
4: **for** each possible set M of 3^+ nodes **do**
5: Compute the minimum connected set-cover formed by nodes in M and additional nodes of degree two by exhaustive search;
6: **end for**
7: Return the minimum of the above outcomes;

Recall that each edge in G^3 corresponds to either a simple edge or a path formed by nodes of degree two and two 3^+ nodes in G. In the rest of this paper, we treat each edge of G^3 as a path. Intuitively, there are two cases for the minimum connected set-cover.

The first case is that the minimum connected set-cover is formed only by nodes of degree two, then it lies on only one edge of G^3. For this case, the algorithm checks all the edges of G^3. Notice that any connected set-cover on an edge must form a line segment. So every possible line segment on each edge is examined. Finally, the algorithm selects the minimum one.

The other case, in which the minimum connected set-cover lies on at least two edges of G^3, is basically the same. In this case, the minimum connected set-cover contains one or more 3^+ nodes. Without loss of generality, we assume that the 3^+ nodes in the minimum connected set-cover form set M. Clearly for those edges of G^3 that have no endpoints in M, the nodes on them can not appear in the minimum connected set-cover. Otherwise, the connectivity can not be maintained. If there is a node v of degree two which is in the minimum connected set-cover, then v must be connected to some 3^+ node $p \in M$ through a path formed by nodes of degree two lying between v and p. For one such edge i, there are two cases as shown in Fig. 2. If only one end point p is in M, then we consider all the possible paths with v and p be the endpoints. Else if both endpoints p and q of edge i are in M, the solution on edge i will be formed by two pieces, each of which is connected with one of the endpoints p and q. During the second round of Algorithm 1, all such possibilities are checked.

Lemma 3 *For a connected set-cover problem $(V, \mathcal{S}, G)$, if every spanning tree of graph G has at most k leaves, then Algorithm 1 finds the minimum connected set-cover in polynomial time.*

Proof Clearly in both cases, our exhaustive search-based algorithm finds the optimal solution. So here we only need to analysis the time complexity of the algorithm in these two cases.

Case 1 The minimum connected set-cover of $(V, \mathcal{S}, G)$ consists of only nodes of degree two.

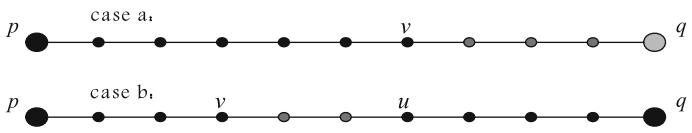

Fig. 2 Two cases when at least one 3^+ node is in the minimum connected set-cover

Suppose there are totally n_i nodes on edge i of G^3, then these n_i nodes form a line segment. If a connected set-cover exists on such a line segment, it must also form a line segment to maintain connectivity. Based on the selections of the endpoints, there are $\binom{n_i}{2} + n_i$ different such segments on edge i (Here n_i is for the case that the minimum connected set-cover is a single node). Hence during the exhaustive search, we only need to check these $\binom{n_i}{2} + n_i$ possibilities. This procedure takes time of $O(n_i^2)$. By Lemma 1, each node of G has degree at most k. Then there are at most $k(5k-5)$ edges in G^3. Hence this case takes $O(k^2n^2)$ time.

Case 2 The minimum connected set-cover of $(V, \mathcal{S}, G)$ lies on two or more edges of G^3.

Notice that there are $2^{5k-5} - 1$ possible nonempty sets of 3^+ nodes. Each of them is checked by our algorithm. Consider such a set M, the degree of each 3^+ node in M is at most k by Lemma 1. Hence there are at most $k|M|$ edges which have endpoints in M. Let edge i formed by n_i nodes be one of such edges. In case a as shown in Fig. 2, there are $n_i - 1$ choices of v, so is the number of choices of the path with p and v as endpoints. While in case b, there are $\binom{n_i}{2} - k$ different selections of $\{v, u\}$. Hence for M, we need to check $O(k|M|n_i^2)$ possibilities. Thereby, the time needed to perform the exhaustive search for case 2 is $O(2^{5k}kn^2)$.

Based on the above analysis for the two cases, we can claim that Algorithm 1 finds the minimum connected set-cover in polynomial time. □

Theorem 1 *Given a graph G and a positive integer k, if every spanning tree of graph G has at most k leaves, then there is a polynomial-time algorithm solving the CSC problem on graph G.*

Theorem 2 *If $NP \neq P$, then the CSC problem is NP-hard if and only if G has a spanning tree with the number of the leaves growing to infinity as the size of G grows to infinity.*

Proof a. The only if part can be explained by the polynomial-time algorithm.

b. Given an instance of the set-cover problem $(V, \mathcal{S})$, we can construct a connected set-cover problem $(V', \mathcal{S}', G)$ in which G satisfies the spanning tree constraint as follows. Create a new element v_0 and a related set $s_0 = \{v_0\}$. Then $V' = V \cup \{v_0\}$ and $\mathcal{S}' = \mathcal{S} \cup \{\{v_0\}\}$. Let G be a star graph with the center $\{v_0\}$ connecting to all the sets in $\mathcal{S}$. Clearly, there is only one spanning tree for graph G, which is itself. The number of leaves of G grows to infinity as the size of G goes to infinity. It is easy to verify that $(V, \mathcal{S})$ has a set-cover C if and only if $(V', \mathcal{S}', G)$ has a connected set-cover $C \cup \{\{v_0\}\}$. □

Corollary 1 *Given a graph G and two positive integers k and k', if G has no more than k 3^+ nodes and the maximum degree of nodes in G is no larger than k', then there is a polynomial-time algorithm solving the CSC problem on graph G.*

3 $(1 + \ln \delta)$-approximatable condition

Theorem 3 *If G has a spanning tree with the number of leaves growing to infinity as the size of G grows to infinity, then there is no polynomial-time $(\rho \ln \delta)$-approximation for $0 < \rho < 1$ for the CSC problem $(V, \mathcal{S}, G)$ unless $NP \subset DTIME(n^{O(\log \log n)})$, here δ is the maximum size of the sets in $\mathcal{S}$.*

Proof Feige [3] has proved that for any $0 < \rho < 1$, there is no approximation algorithm with performance ratio $\rho \ln \delta$ for the set-cover problem, unless $NP \subset DTIME(n^{\log n \log n})$.

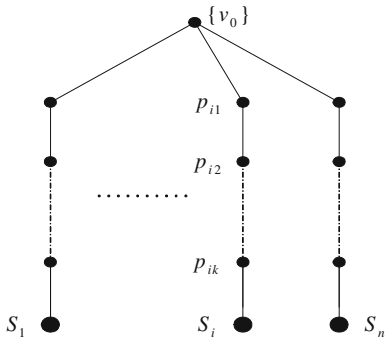

Fig. 3 Construction of the connected set-cover problem

By contradiction, we assume that there is a polynomial-time $(\rho' \ln \delta)$-approximation for the CSC problem with $0 < \rho' < 1$.

Given an instance $(V, \mathcal{S})$ of the set-cover problem, we construct a graph G for the CSC problem as shown in Fig. 3. Create a new element v_0 and a set $\{v_0\}$. Connect $\{v_0\}$ with each set $S_i \in \mathcal{S}$ via a path $P_i = p_{i1}p_{i2}, \ldots, p_{ik}$. Each p_{ij} is a singleton which contains an element v^*. Then we get an instance of the CSC problem $(V', \mathcal{S}', G)$. Here $V' = V \cup \{v_0, v^*\}$, $\mathcal{S}' = \mathcal{S} \cup \{\{v_0\}\} \cup \{p_{ij} | 1 \leq i \leq n, 1 \leq j \leq k\}$ where k is an integer which satisfies that $k > (\rho')/(1 - \rho')$. Clearly, G is a spider graph with the number of leaves being $(n - 1)/k$ which grows to infinity as the size of G goes to infinity.

Suppose the optimal solutions of the set-cover problem and the CSC problem have sizes opt and opt^* respectively. Then according to our earlier assumption, we have a polynomial-time algorithm that produces a solution A^* of size no larger than $\rho' \ln \delta \cdot opt^*$ for the CSC problem. Then we can easily get a solution A for the set-cover problem that satisfies

$$|A| = (|A^*| - 1)/k.$$

Hence we have

$$\begin{aligned} |A| &= (|A^*| - 1)/k \\ &\leq (\rho' \ln \delta \cdot opt^* - 1)/k \\ &= (\rho' \ln \delta (opt \cdot k + 1) - 1)/(k) \\ &\leq \rho' \left(1 + \frac{1}{k}\right) \ln \delta \cdot opt \\ &< \ln \delta \cdot opt, \end{aligned}$$

which means that we have a polynomial-time ρ-approximation algorithm for the set-cover problem with $0 < \rho < 1$. □

We propose an algorithm for the CSC problem as shown by Algorithm 2. Here $\bar{S}$ is constructed as follows. After a spanning tree T is selected, the nodes of M are all connected. Here the nodes of T are all 3^+ nodes and the edges of T are either simple edges or paths formed by nodes of degree two. If T has already covered all the elements in V, then T is already a connected set-cover. Or else if some element of V is not covered yet, then we need to form a connected set-cover via additional nodes of degree two. Notice that if some node v of degree two is selected, then the path formed by nodes of degree two between v and some 3^+ node must also be selected. For each node $p \in M$, let $P = p_1 p_2, \ldots, p_i$ be the path connected with p by p_1 (P has no intersection with T), here each node of P has degree two. Suppose

Algorithm 2 Approximation algorithm for the CSC problem

Input: An instance of the CSC problem $(V, \mathcal{S}, G)$.
Output: A connected set-cover within $(1 + ln\delta)$ of the optimal solution.
1: **for** each maximal path (i, j) formed by nodes of degree two in G **do**
2: Perform an exhaustive search on (i, j);
3: **end for**
4: **for** each possible set of 3^+ nodes M **do**
5: **for** each possible spanning tree T of nodes in M **do**
6: Let the the set of elements covered by T be V_T
7: Find a set-cover S_0 from $\bar{S}$ for $V_0 = V - V_T$ via the greedy algorithm in [4];
8: Form a CSC by combining T and S_0;
9: **end for**
10: **end for**
11: Return the minimum among all the outcomes;

P is on edge a with n_a nodes. Form $n_a - 1$ sets with the form of $S_j = (p_1, p_2, \ldots, p_j)$ as shown in Fig. 4, where $1 \leq j \leq n_a - 1$. Then $\bar{S}$ is the set that contains all such sets as S_j for each $p \in M$.

Theorem 4 *If G is a tree with at most k nodes of degree more than two, then the CSC problem has a polynomial-time* $(1 + \ln \delta)$*-approximation algorithm.*

Proof We prove this theorem by proving that if G is a tree with at most k 3^+ nodes, then Algorithm 2 is a polynomial-time $(1 + \ln \delta)$-approximation algorithm for the CSC problem $(V, \mathcal{S}, G)$.

Let OPT with $|OPT| = opt$ be the minimum connected set-cover. Then similar with the analysis in the last section, there are two cases for OPT.

Case 1 If OPT lies only on one edge of G^3, then by exhaustive search on each edge of G^3, Algorithm 2 finds OPT.

Case 2 If OPT lies on two or more edges of G^3, then at least one 3^+ node is in the OPT. Let $M = V^3 \cap OPT$ be the set of 3^+ nodes in OPT. Then there is only one spanning tree for nodes of M in G^3, which is the minimum connected subgraph T of G that contains all the nodes of M. T must be in OPT, or else OPT can not be connected. Suppose the set of elements get covered by T is V_T and $V_0 = V - V_T$. Clearly, V_0 is the set of elements that are not covered by T. Then a set-cover S_0 for V_0 is generated by the greedy algorithm. Let the optimal set-cover for elements in V_0 be S^*. As $OPT - T$ is also a set-cover for V_0, so

$$|OPT - T| \geq |S^*|.$$

On the other side,

$$|S_0| \leq (1 + \ln \delta)|S^*|.$$

Hence we have

$$|S_0| \leq (1 + \ln \delta)|OPT - T|.$$

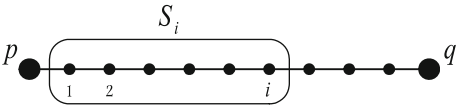

Fig. 4 Construction of $\bar{S}$

Let A be the final solution generated by our algorithm and B be the solution generated during the algorithm that contains T. Then $B = T \cup S_0$ and we have

$$\begin{aligned} |A| \le |B| &= |T| + |S_0| \\ &\le |T| + (1 + \ln \delta)|OPT - T| \\ &\le (1 + \ln \delta)|OPT|. \end{aligned}$$

This means that Algorithm 2 generates a solution of size less than $(1 + \ln \delta)$ of the optimal solution. In the following we show that the algorithm runs in polynomial time.

In Case 1, the exhaustive search on an edge i with n_i nodes takes time $O(n_i^2)$. Because G is a tree, the number of edges in G^3 is at most $n - 1$. So it takes $O(n^3)$ time for Case 1. In Case 2, there are totally $2^k - 1$ possible non-empty set of 3^+ nodes. For each of these 3^+ node sets, a greedy procedure with time complexity $O(n^2)$ is performed. so the total time cost for Case 2 is $O(2^k n^2)$. The total time complexity of Algorithm 2 is $O(2^k n^3)$.

Based on the above analysis, Algorithm 2 is a polynomial-time $(1 + \ln \delta)$-approximation algorithm. □

Theorem 5 *For graph G and a given positive integer k, if there are at most k 3^+ nodes in graph G, then there is a polynomial-time $(1 + \ln \delta)$-approximation algorithm for the CSC problem on graph G.*

Proof We prove that if G is a graph with at most k 3^+ nodes, then Algorithm 2 is a polynomial-time $(1 + \ln \delta)$-approximation algorithm for the CSC problem $(V, \mathcal{S}, G)$. The proof is basically the same as that of Theorem 4. The difference is that for each nonempty set M of 3^+ nodes, there might be several spanning trees connecting the nodes of M. So here the algorithm considers all such possible spanning trees. For each of these trees, a connected set-cover is generated.

The number of simple edges between nodes of G^3 is at most $k(k-1)/2$ and the number of paths which are formed by nodes of degree two and connecting nodes of G^3 is at most $n - k$. So there are at most $n + (k^2 - 3k)/2$ different edges between nodes of M in G^3. It takes $|M| - 1$ edges to form a spanning tree for nodes in M. So there are no more than $\binom{n+k^2}{|M|-1}$ possible spanning trees for M. This will add a factor of $O((n + k^2)^{(k-1)})$ to the time complexity comparing with that in Theorem 4. So the time complexity of Algorithm 2 for graph G is $O(2^k (n + k^2)^{k+2})$. □

4 $O(\log^{2-\varepsilon} n)$-inapproximability for $\varepsilon > 0$

In this section, we show that for general graphs, there is no polynomial-time $O(\log^{2-\varepsilon} n)$-approximation for the CSC problem for any $\varepsilon > 0$. Firstly, we introduce the Group Steiner Tree problem, which is related closely with the CSC problem.

Definition 3 Group Steiner Tree problem: Given an edge-weighted graph $G = (V, E)$, a root vertex $r \in V$ and k nonempty subsets of vertices, $g_1, g_2, \ldots, g_k$, find the minimum total weight tree containing r and at least one vertex from each subset g_i.

The following fact has been proved in [5].

Lemma 4 *There is no polynomial-time $O(\log^{2-\varepsilon} n)$-approximation for the Group Steiner Tree problem for any $\varepsilon > 0$ unless NP has quasi-polynomial Las-Vegas algorithms.*

Now, for the CSC problem, we have a similar result.

Theorem 6 *There is no polynomial-time $O(\log^{2-\varepsilon} n)$-approximation for the CSC problem for any $\varepsilon > 0$ unless NP has quasi-polynomial Las-Vegas algorithms.*

Proof We construct a reduction from the Group Steiner Tree problem to the CSC problem as follows.

Consider an input of the Group Steiner Tree problem, a graph $G = (V, E)$ with edge weight $w : E \to Z_+$, a root vertex $r \in V$ and k nonempty subsets of vertices, $g_1, g_2, \ldots, g_k$. Define $X = \{g_0, g_1, \ldots, g_k\}$ where $g_0 = \{r\}$. For each node $u \in V$, define $S_u = \{g_i \mid u \in g_i\}$. For each edge $(u, v) \in E$, construct a path connecting S_u and S_v with $k \cdot w(u, v)$ intermediate nodes. Let the obtained graph on $\mathcal{V} = \{S_v \mid v \in V\}$ and these intermediate nodes be H. Suppose there is a polynomial-time $O(\log^{2-\varepsilon} n)$-approximation for the CSC problem. Let D be such an approximation solution on instance $(X, \mathcal{V}, H)$ and let opt_{CSC} be the number of nodes in an optimal solution, i.e., the objective function value of the CSC problem. Then we have

$$|D| \leq O(\log^{2-\varepsilon} n) opt_{CSC}.$$

Clearly, D is the node set of a tree T in H, which induces a tree T' in G. Denote by $w(T')$ the total edge weight of T'. Then

$$w(T') \leq \frac{|D|}{k}.$$

Now, let T^* be a minimum Group Steiner Tree, which induces a tree T^{**} in H. Then the number of nodes in T^{**} is at most $w(T^*)(k + 2)$ and is at least opt_{CSC}. Therefore,

$$w(T') \leq O(\log^{2-\varepsilon} n) \cdot \frac{k+2}{k} \cdot w(T^*) = O(\log^{2-\varepsilon} n) \cdot w(T^*),$$

that is, T' is a polynomial-time $O(\log^{2-\varepsilon} n)$-approximation for the Group Steiner Tree problem. By Lemma 4, NP has quasi-polynomial Las-Vegas algorithms.

In the above, we treat $(X, \mathcal{V}, H)$ as an instance of the CSC problem. The reader may suspect this treatment because

(a) it is unclear how to represent each intermediate node as a subset of X, and
(b) it is possible that $S_u = S_v$, but in the definition of the CSC problem, $\mathcal{V}$ is not a multiple subset collection.

We remark that (a) and (b) can be fixed easily. In fact, for each intermediate node x, we can introduce a new element e_x and let $S_x = \{e_x\}$ represent node x. For each S_v, $v \in V$, we can also introduce a new element e_v and add e_v into S_v. Finally, put all new elements into S_r and X. Note that g_0 is contained only in S_r. Therefore, any feasible solution of the CSC problem must contain S_r. This means that those subsets representing intermediate nodes are useless for covering elements and they are useful only in the establishment of connectivity. Moreover, we can easily see that this modification does not effect the size of any solution of the CSC problem. □

Acknowledgments This work was supported in part by the National Scientific Foundation of USA under Grant No. CNS-0524429, CCF-0627233, and CCF-0514796 and also was jointly sponsored by MEST, Korea under WCU (R33-2008-000-10044-0), a NRF Grant under (KRF-2008-314-D00354), and MKE, Korea under ITRC NIPA-2010-(C1090-1021-0008).

References

1. Shuai, T., Hu, X.: Connected set-cover problem and its applications. In: Proceeding of AAIM, pp. 243–254 (2006)
2. Gupta, H., Das, S., Gu, Q.: Connected sensor cover: self-organization of sensor networks for efficient query execution. In: Proceeding of ACM MOBIHOC (2003). doi:10.1145/778415.778438
3. Feige, U.: A threshold of ln n for approximating set-cover. In: Proceeding of ACM STOC (1998). doi:10.1145/285055.285059
4. Chvatal, V.: A greedy heuristic for the set-covering problem. Math. Oper. Res. (1979). doi:10.2307/3689577
5. Halperin, E., Krauthgamer, R.: Polylogarithmic inapproximability. In: Proceeding of ACM STOC (2003). doi:10.1145/780542.780628