Polynomial Time Approximation Scheme for Connected Vertex Cover in Unit Disk Graph

Zhao Zhang[1,⋆], Xiaofeng Gao[2,⋆⋆], and Weili Wu[2,⋆⋆]

[1] College of Mathematics and System Sciences, Xingjiang University, Urmuqi, Xinjiang, China
zhzhao@xju.edu.cn

[2] Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083, USA
{xxg05200,weiliwu}@utdallas.edu

Abstract. Connected Vertex Cover Problem (CVC) is an $\mathcal{N}P$-hard problem. The currently best known approximation algorithm for CVC has performance ration 2. This paper gives the first Polynomial Time Approximation Scheme for CVC in Unit Disk Graph.

Keywords: Connected Vertex Cover, Unit Disk Graph.

1 Introduction

Minimum Vertex Cover Problem (MVC) is a classical optimization problem in graph and combinatorial theory. For a undirected graph $G = (V, E)$, a subset $C \subseteq V$ is called a *vertex cover* of G (VC) if for any $(v, w) \in E$, either $v \in C$ or $w \in C$. MVC is to find a vertex cover of G with the minimum number of vertices. This problem has many real-world applications [3], including many in the field of bioinformatics. It can also be used in the construction of phylogenetic trees, in phenotype identification, and in analysis of microarray data. MVC has been studied extensively in the literature [8]. It is known to be $\mathcal{NP}$-hard [9] for a long time. Papadimitriou et al. [13] proved that VC is APX-complete, and Monien et al. [12,3] gave an approximation algorithm for VC with ratio $1 - \frac{\log \log n}{2 \log n}$ (n is the number of vertices).

If furthermore, a vertex cover C induces a connected subgraph $G[C]$, then C is called a *connected vertex cover* (CVC). The Minimum Connected Vertex Cover Problem (MCVC) is to find a CVC with minimum cardinality. MCVC problem is an enforced version of MVC when certain connectivity constraints are needed in some applications. For example, in routing and wavelength assignment (RWA) problem for optical networks, people select a suitable path and wavelength among the many possible choices with the help of CVC. MCVC is

⋆ Support in part by NSFC (60603003) and XJEDU. This work was done while this author visited at University of Texas at Dallas.

⋆⋆ Support in part by National Science Foundation under grants CCF-9208913 and CCF-0728851.

also $\mathcal{NP}$-hard. In fact, Garey and Johnson [7] showed that MCVC is as hard to approximate as MVC. The currently best known approximation algorithms for MCVC have performance ratio 2, which were given by Arkin et al. [2,14].

In this paper, we consider MCVC problem in *Unit Disk Graphs* (UDG). A graph G is an UDG if each vertex of G is associated with the center of a disk with diameter 1 on the plane, and two vertices u, v of G are adjacent if and only if the two disks corresponding to u and v have non-empty intersection. In another word, $(u, v) \in E(G)$ if and only if the Euclidean distance between the centers corresponding to u and v is at most 1. Such a set of unit disks on the plane is called the *geometric representation of* G. When talking about a unit disk graph in this paper, we assume that the geometric representation is given, since it has been proved in [10] that determining whether a graph is a UDG is $\mathcal{NP}$-complete. UDG is widely used in wireless networks, where each vertex represents an idealized multi-hop radio based station, and the corresponding disk is the communication range of the station. For MVC in UDG, there exists Polynomial Time Approximation Scheme (PTAS). That is, for any positive real number ε, there exists a $(1+\varepsilon)$-approximation. In fact, Erlebach et al. [5,11] presented a PTAS for Minimum Weight Vertex Cover (MWVC) in *Disk Graph* (DG), where DG is a generalization of UDG, in which disks have different radiuses.

In this paper, we present the first PTAS for CVC on UDG, using partition technique and shifting strategy. Such an approach was used for Steiner trees in the plane [15]. A more complicated approach was used for connected dominating set [4]. It should be noted that in [4], the technique is heavily based on the property of 2-dimension. In other words, the technique cannot be applied to higher dimensional space, e.g., unit ball graphs, which is also an important model for wireless sensor networks. The technique presented in this paper can be applied to any dimension. Therefore, it is actually proved in this paper that there exists PTAS in unit n-dimensional ball graphs for any n.

The idea of the algorithm is: First, we take an area containing all vertices of the graph, and partition it into small squares. For each small square, define the inner area and the boundary area, such that the inner area and the boundary area of a same small square has an overlap. For each component of the inner area, compute a minimum CVC. To cover edges not in the inner area, use a constant-approximation algorithm to compute a connected vertex cover C_0 of G, and union those vertices of C_0 which belong to the 'boundary area' of the partition into the above CVC's. The overlap of inner area and boundary area ensures the connectivity of the output. The shifting strategy is used to select a partition such that the number of vertices of C_0 falling into the boundary area of this partition is small enough relative to ε.

The rest of this paper is organized as follows. In Section 2, we introduce some terminologies used to describe the algorithm. In Section 3, the algorithm is presented. In Section 4 we show the correctedness of our algorithm, analyze the time complexity, and prove that it is a PTAS. A conclusion is given in Section 5.

2 Preliminaries

In this section we introduce the symbols and definitions used for algorithm description.

For a given UDG $G = (V, E)$, where $|V| = n$, we assume that all the disks are located in a square plane $Q = \{(x, y)|0 \leq x \leq q, 0 \leq y \leq q\}$, where q is related to n. Using partition strategy, we divide Q into squares each with side length $m \times m$. We set $m = \lceil \frac{48\rho}{\varepsilon} \rceil$, where ρ is a constant which is the approximation ratio of an APX for CVC (for example, ρ can be taken as 2 if we use the 2-approximation algorithm in [6]), and ε is an arbitrary positive number. Let $p = \lfloor \frac{q}{m} \rfloor + 1$. Since we shall use shifting policy, we widen Q into a bigger region $\widetilde{Q} = \{(x, y)| - m \leq x \leq pm, -m \leq y \leq pm\}$ (see Fig. 1).

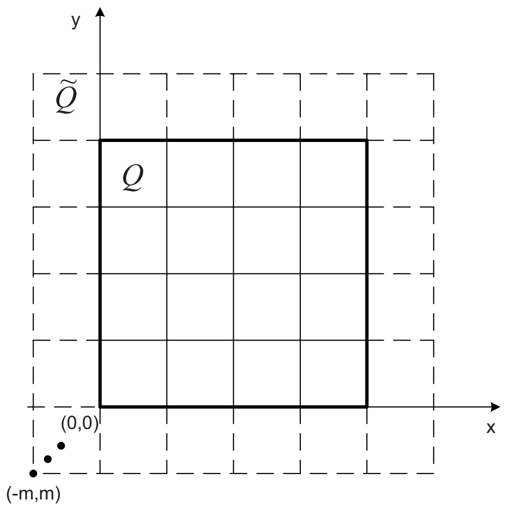

Fig. 1. Partition for Graph G

Name this partition as $P(0)$, and denote by $P(a)$ the partition obtained from $P(0)$ by shifting it such that the left-bottom corner of $P(a)$ is at $(a - m, a - m)$, for $a = 0, 1, \cdots, m - 1$.

For each square e, we define the *Inner* area I_e and *Boundary* area B_e (see Fig. 2). If $e = \{(x, y)|im \leq x \leq (i + 1)m, jm \leq y \leq (j + 1)m\}$,

$$I_e = \{(x, y)|im + 1 \leq x \leq (i + 1)m - 1, jm + 1 \leq y \leq (j + 1)m - 1\},$$

$$B_e = e - \{(x, y)|im + 2 \leq x \leq (i + 1)m - 2, jm + 2 \leq y \leq (j + 1)m - 2\}.$$

Note that there is an overlap of I_e and B_e.

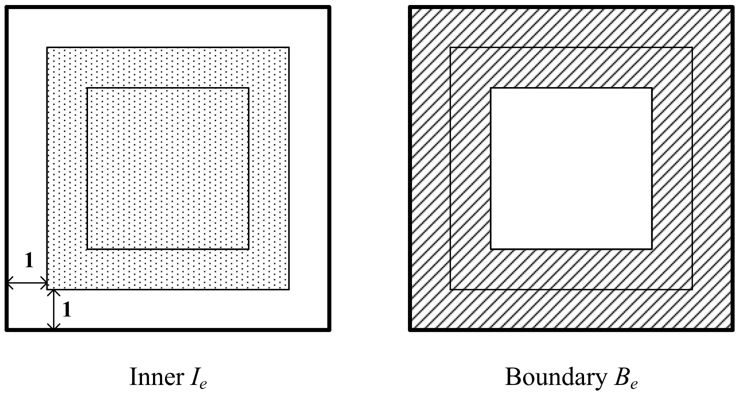

Fig. 2. Inner Region and Boundary Region for a Square

3 Algorithm Overview

For a partition $P(a)$, denote by $B(P(a)) = \bigcup_{e\in P(a)} B_e$. The algorithm is executed in two phases.

Phase I. Use a ρ-approximation to compute a CVC C_0 for graph G. Let $C_0(a) = C_0 \cap B(P(a))$ be the set of vertices of C_0 lying in the boundary area of partition $P(a)$. Choose a^* such that $|C_0(a^*)| = \min |C_0(a)|$.

Phase II. For any square $e \in P(a^*)$, denote by G_e the subgraph of G induced by the vertices in I_e, and $Comp(G_e)$ the set of connected components in G_e. For each square e and each component $H \in Comp(G_e)$, use exhaust search to find a minimum CVC C_H of H. Set $C_e = \bigcup_{H\in Comp(G_e)} C_H$.

Final Result Output $C = C_0(a^*) \cup (\bigcup\limits_{e\in P(a^*)} C_e)$.

4 Analysis of the Algorithm

In this section, we firstly prove the correctness of our algorithm, and then discuss the overall time complexity, that is, we prove that our algorithm runs in polynomial time. Finally, we give the performance ratio of the algorithm, which is $(1 + \varepsilon)$.

4.1 Correctness

To prove that the output C of our algorithm is a CVC for graph G, we firstly prove that C is a vertex cover for G, then prove that the induced subgraph $G[C]$ is connected.

Lemma 1. *C is a vertex cover for $G = (V, E)$.*

Proof. For each square e, the inner area I_e and the boundary area B_e have an overlap with width 1. Since for any edge (v, w), the Euclidean distance between v and w is less than or equal to 1, we see that both v and w belong to the inner area I_e for some square e, or belong to the boundary area $B(P(a^*))$. In the former case, the edge (v, w) is in a component H of G_e. By Phase II of the algorithm, either $v \in C_H$ or $w \in C_H$, meaning that(v, w) can be covered by $C_e \subseteq C$. In the second case, by Phase I of the algorithm, C_0 is a CVC of G. Therefore either $v \in C_0(a^*)$ or $w \in C_0(a^*)$, meaning that (v, w) can be covered by $C_0(a^*) \subseteq C$. Thus we have proved that any edge in G is covered by C. So C is a vertex cover of G.

Lemma 2. *The induced subgraph $G[C]$ is connected.*

Proof. We prove this lemma by two steps. In step 1, we show that distinct connected components in $G[C_0(a^*)]$ (if exist) can be connected through vertices in $\bigcup_{e \in P(a^*)} C_e$. In step 2, we show that there is no other components of $G[C]$ left after step 1.

Step 1. Let H_1 and H_2 be two components in $G[C_0(a^*)]$ which are 'closest' in $G[C_0]$ with each other. Then, there is a path $P = (v_1, v_2, \cdots, v_t)$ of $G[C_0]$ connecting H_1 and H_2 through the inner area of 'one' square e. Without loss of generalization, we may assume that $v_1 \in V(H_1)$, $v_t \in V(H_2)$ and $\{v_2, \cdots, v_{t-1}\} \subseteq I_e$. Fig. (see 3 for illustration).

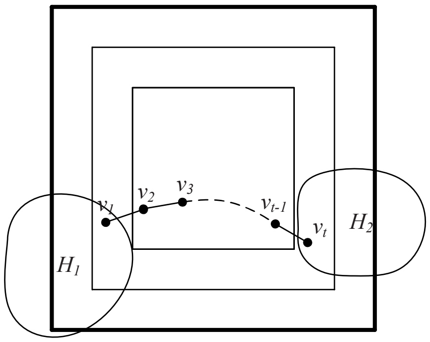

Fig. 3. An Illustration for H_1 and H_2

It is easy to see that v_1 and v_t belong to $I_e \cap B_e$, so P is in a connected component H of G_e. Based on Phase II of our algorithm, P is covered by C_H. It follows that [either $v_1 \in C_H$ or $v_2 \in C_H$], and [either $v_{t-1} \in C_H$ or $v_t \in C_H$]. Since $G[C_H]$ is connected, we see that H_1 and H_2 are connected through $G[C_H]$.

Step 2. Let $\widetilde{G}$ be the component of $G[C]$ containing all vertices of $C_0(a^*)$. Such $\widetilde{G}$ exists because of step 1. Suppose $\widetilde{G} \neq G[C]$, then there exists a square e and a connected component H of G_e such that

(i) $C_H \cap C_0(a^*) = \emptyset$ and

(ii) no vertex of C_H is adjacent with any vertex in $C_0(a^*)$.

Let x be a vertex in C_H. Then either $x \in C_0$ or x is adjacent with a vertex $y \in C_0$. We firstly assume that $x \in C_0$. From (i), we know that $x \notin C_0(a^*)$, so $x \in e \backslash B_e$. Since $G[C_0]$ is connected, there is a path P in $G[C_0]$ connecting x to the other parts of G outside of e. Suppose $P = (v_0, v_1, ..., v_t)$, where $v_0 = x$, $v_t \notin e$, and $\{v_1, ..., v_{t-1}\} \subseteq e$. Let i be the index such that v_i is the first vertex on P with $v_i \in B_e$. Then

(iii) $v_i \in C_0(a^*)$;

(iv) $v_i \in I_e$ and thus v_i and x belong to a same component of G_e, which is H;

(v) both v_{i-1} and v_i are in I_e, and hence the edge (v_{i-1}, v_i) are in H (note that $i \geq 1$ since $v_0 = x \notin B_e$).

By (v) and Phase II of the algorithm, either $v_i \in C_H$ or $v_{i-1} \in C_H$. But this contradicts (i) (ii) and (iii).

The case that $x \notin C_0$ but is adjacent with a vertex $y \in C_0$ can be proved similarly.

Therefore, we have proved that $\widetilde{G} = G[C]$. □

Based on the conclusions from Lemma 1 and Lemma 2, we obtain the following theorem showing the correctness of our algorithm.

Theorem 1. *The output C of our algorithm is a connected vertex cover for G.*

4.2 Time Complexity

In this subsection we consider the time complexity of our algorithm. Phase I of the algorithm uses a polynomial time ρ-approximation to compute C_0. Phase II uses exhaust search which is the most time consuming part. We shall prove that this part can also be executed in polynomial time, by implementing the relation between vertex cover and independent set.

Lemma 3. *The number of independent unit disks in an $m \times m$ square is at most $\lfloor \frac{(m+2)^2}{\pi} \rfloor$.*

Proof. Enlarge the $m \times m$ square to an $(m+2) \times (m+2)$ square by adding a boundary with width one. Then all the disks whose centers are in the $m \times m$ square lie completely in the $(m+2) \times (m+2)$ square. Since each unit disk occupies area π, the result follows from the independence assumption.

With the help of Lemma 3, we have the following theorem.

Theorem 2. *The running time of our algorithm is $n^{O(1/\varepsilon^2)}$, where n is the number of vertices in the graph.*

Proof. It is well known that a vertex set S is a vertex cover of a graph if and only if its complement is an independent set. Thus by Lemma 3, each $V(H) \backslash C_H$ contains at most $\lfloor \frac{(m+2)^2}{\pi} \rfloor$ independent vertices, and therefore the exhaust search for C_H (which can be done by considering the complement of each independent set in H) takes time at most $\sum_{k=0}^{\lfloor\frac{(m+2)^2}{\pi}\rfloor}\binom{n_H}{k} = n_H^{O(m^2)}$, where n_H is the number of vertices in H, and the total running time for phase II is at most $\sum_{e,H} n_H^{O(m^2)} = \big(\sum_{e,H} n_H\big)^{O(m^2)} = n^{O(m^2)} = n^{O(1/\varepsilon^2)}$.

4.3 Performance

Here we prove that our algorithm is a $(1+\varepsilon)$-approximation.

Definition 1. For two subgraphs G_1, G_2 of G, the *distance between* G_1 *and* G_2 is the length of a shortest path of G connecting G_1 and G_2 (where 'length' means the number of edges on the path), denoted by $dist(G_1, G_2)$.

In another word, if $dist(G_1, G_2) = k$, then G_1 and G_2 can be connected through $k-1$ vertices. If a vertex cover of a connected graph is not a connected vertex cover, the distance between connected components of the subgraph induced by the vertex cover is not far, as can be seen from the following lemma.

Lemma 4. *Suppose H is a connected graph, and C is a vertex cover of H. If $H[C]$ is not connected, then there exist two components R_1, R_2 of $H[C]$ such that $dist(R_1, R_2) = 2$.*

Proof. Let R_1, R_2 be two 'closest' connected components of $G[C]$, and $P = (v_0, v_1, ..., v_t)$ be a shortest path of H connecting R_1 and R_2, $v_0 \in V(R_1)$ and $v_t \in V(R_2)$. If $t \geq 3$, consider the edge (v_1, v_2). Since C covers H, we have either $v_1 \in C$ or $v_2 \in C$. Suppose, without loss of generality, that $v_1 \in C$. Let R_3 be the component of $G[C]$ containing v_1. Then $R_3 \neq R_1$ and R_2, and $dist(R_3, R_2) < dist(R_1, R_2)$, contradicting our choice of R_1 and R_2.

The following lemma is well known in unit disk graph.

Lemma 5. *Let G be a unit disk graph and u be a vertex in $V(G)$. The there are at most 5 independent vertices in $N(u)$, where $N(u)$ is the set of vertices adjacent with u in G.*

The following theorem shows that our algorithm is a PTAS.

Theorem 3. *Let C^* be an optimal CVC for G, and C be the output of our algorithm. Then $|C| \leq (1+\varepsilon)|C^*|$.*

Proof. Firstly, we prove that

$$|C_0(a^*)| \leq \frac{\varepsilon}{6}|C^*|. \tag{1}$$

When the partition shifts, a vertex of C_0 belongs to at most 8 boundary areas of $B(P(a))$'s. Therefore, we have,

$$|C_0(0)| + |C_0(1)| + \cdots + |C_0(m-1)| \leq 8|C_0|,$$

and thus

$$|C_0(a^*)| \leq \frac{8\rho|C^*|}{m} \leq \frac{\varepsilon}{6}|C^*|.$$

Next, we shall add some vertices to C^* such that the resulting set $\widetilde{C}$ satisfies: for each square e and

$$\text{for each component } H \in Comp(G_e),\ \widetilde{C} \cap V(H) \text{ is a CVC of } H. \qquad (2)$$

For a square e, let $\widetilde{C}_e = C^* \cap I_e$. It is easy to see that for each component $H \in Comp(G_e)$, $\widetilde{C}_e \cap V(H)$ covers H. Suppose there exists a component $H \in Cpomp(G_e)$ such that requirement (2) is not satisfied. By Lemma 4, there are two components R_1, R_2 of $G[\widetilde{C}_e \cap V(H)]$ such that R_1 and R_2 can be connected through one vertex in $V(H)\backslash\widetilde{C}_e$. Add this vertex to $\widetilde{C}_e$. If the new $\widetilde{C}_e$ still does not satisfy requirement (2), continue as above to add vertices to merge components. Suppose this is done k times before $\widetilde{C}_e$ satisfies (2), then

$$|\widetilde{C}_e| \leq |C^* \cap e| + k. \qquad (3)$$

On the other hand, we can show that

$$|C_0(a^*) \cap e| \geq \frac{k}{5}. \qquad (4)$$

For this purpose, we suppose that the components merged are in the order that R_1 with R_2, R_3 with R_4, $\cdots$, R_{2k-1} with R_{2k}. For simplicity of presenting the idea, we firstly assume that all the above R_j's are distinct components of $G[C^* \cap I_e]$. For each $i = 1, 2, \cdots, k$, let x_i be a vertex in $V(R_{2i-1}) \cap B_e \cap I_e$, such that x_i is adjacent with a vertex $y_i \in B_e \setminus I_e$. Such x_i exists since R_{2i-1} is connected to the outer parts of e through C^*. Then either $x_i \in C_0$ or $y_i \in C_0$. Set $z_i = x_i$ if $x_i \in C_0$ and $z_i = y_i$ otherwise. Note that both $x_i, y_i \in B_e$. Hence $z_i \in C_0(a^*) \cap e$. A vertex may serve more than once as z_i's. For example, it is possible that there are two indices $i \neq j$ such that the vertex of C_0 covering edges (x_i, y_i) and (x_j, y_j) is the same $y_i = y_j \in C_0$. In this case, we see that x_i and x_j are independent since they belong to different components of $G[C^* \cap I_e]$. Then by Lemma 5, such vertex serves at most 5 times as z_i's, and inequality (4) follows. Next, suppose the R_j's are not all distinct. For example, suppose R_3 is the component obtained by merging R_1 and R_2. Then x_3 can be chosen such that $x_3 \in V(H_2) \cap B_e \cap I_e$, which is independent with x_1. In general we can find k independent vertices $x_1, x_3, ..., x_{2k-1}$ and thus (4) also holds in this case.

Combining inequalities (3) and (4), we have

$$|\widetilde{C}_e| \leq |C^* \cap e| + 5|C_0(a^*) \cap e|. \qquad (5)$$

Since in Phase II of the algorithm, C_e is a 'minimum' vertex set satisfying requirement (2) for each square e, we have $|C_e| \leq |\widetilde{C}_e|$. Combining this with (1) and (5), we have

$$\begin{aligned}|\bigcup_{e\in P(a^*)} C_e| = \bigcup_{e\in P(a^*)} |C_e| \leq \sum_{e\in P(a^*)} |\widetilde{C}_e| \\ \leq \sum_{e\in P(a^*)} \big(|C^* \cap e| + 5|C_0(a^*) \cap e|\big) \\ = |C^*| + 5|C_0(a^*)| \leq (1 + \frac{5\varepsilon}{6})|C^*|.\end{aligned}$$

Hence

$$|C| \leq |C_0(a^*)| + |\bigcup_{e\in P(a^*)} C_e| \leq (1+\varepsilon)|C^*|.$$

5 Conclusion

In this paper, we presented the first polynomial time approximation scheme to compute a connected vertex cover of a graph. The method used in this paper can be applied to CVC problems in n-dimensional ball graphs. In a *unit ball graph*, each vertex corresponds to the center of a unit ball in the n-dimensional space, and two vertices are adjacent if and only if the Euclidean distance between them is at most 1.

References

1. Abu-Khzam, F.N., et al.: Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments. In: Proc.6th Workshop on Algorithm Engineering and Experiments & 1st Workshop on Analytic Algorithmics and Combinatorics, pp. 62–69 (2004)
2. Arkin, E.M., Halldorsson, M.M., Hassin, R.: Approximating the Tree and Tour Covers of a Graph. Inform. Process. Lett. 47, 275–282 (1993)
3. Bar-Yehuda, R., Even, S.: A local-ration Theorem for Approximating the Weighted Vertex Cover Problem, Analysis and Design of Algorithms for Combinatorial Problems. Annals of Discrete Mathematics 25, 27–46 (1985)
4. Cheng, X., Huang, X., Li, D., Wu, W., Du, D.: A polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks. Networks 42, 202–208 (2003)
5. Erlebach, T., Jansen, K., Seidel, E.: Polynomial-Time Approximation Schemes for Geometric Intersection Graphs. SIAM J. Comput. 34(6), 1302–1323 (2005)
6. Fujito, T., Doi, T.: A 2-Approximation NC Algorithm for Connected Vertex Cover and Tree Cover. Inform. Process. Lett. 90, 59–63 (2004)
7. Garey, M.R., Johnson, D.S.: The Rectilinear Steiner-Tree Problem is $\mathcal{NP}$-Complete. SIAM J. Appl. Math. 32, 826–834 (1977)
8. Hochbaum, D.H.: Approximation Algorithm for $\mathcal{NP}$-hard Problems, PWS, Boston, MA (1996)

9. Karp, R.M.: Reducibility among Combinatorial Problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)
10. Kratochvil, J.: Intersection graphs of noncrossing arc-connected sets in the plane. In: North, S.C. (ed.) GD 1996. LNCS, vol. 1190, pp. 257–270. Springer, Heidelberg (1997)
11. Li, X.Y., Wang, Y.: Simple Approximation Algorithms and PTASs for Various Problems in Wireless Ad-Hoc Networks. Journal of Parallel and Distributed Computing 66, 515–530 (2006)
12. Monien, B., Speckenmeyer, E.: Ramsey Numbers and an Approximation Algorithm for the Vertex Cover Problem. Acta Informatica 22, 115–123 (1985)
13. Papadimitriou, C.H., Yannakakis, M.: Optimization, Approximation, and Complexity Classes. J. Computer and System Sciences 43, 425–440 (1991)
14. Savage, C.: Depth-First Search and the Vertex Cover Problem. Inform. Process. Lett. 14, 233–235 (1982)
15. Wang, L.S., Jiang, T.: An approximation scheme for some Steiner tree problems in the plane. Networks 28(4), 187–193 (1996)