Theoretical Computer Science 412 (2011) 198–208

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs

Feng Zou [a,*], Yuexuan Wang [b], Xiao-Hua Xu [c], Xianyue Li [d,**], Hongwei Du [c], Pengjun Wan [c], Weili Wu [a]

[a] Department of Computer Science, University of Texas at Dallas, Richardson, TX 75080, USA
[b] Institute of Theoretical Computer Science, Tsinghua University, Beijing 100084, China
[c] Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA
[d] School of Mathematics and Statistics, Lanzhou University, Lanzhou, Gansu 730000, China

ARTICLE INFO

Keywords:
Minimum-weighted dominating set
Minimum-weighted connected dominating set
Minimum-weighted chromatic disk cover
Node-weighted Steiner tree
Polynomial-time approximation scheme
Approximation algorithm

ABSTRACT

Given a node-weighted graph, the *minimum-weighted dominating set* (**MWDS**) problem is to find a minimum-weighted vertex subset such that, for any vertex, it is contained in this subset or it has a neighbor contained in this set. And the *minimum-weighted connected dominating set* (**MWCDS**) problem is to find a **MWDS** such that the graph induced by this subset is connected. In this paper, we study these two problems on a unit disk graph. A $(4+\varepsilon)$-approximation algorithm for an **MWDS** based on a dynamic programming algorithm for a *Min-Weight Chromatic Disk Cover* is presented. Meanwhile, we also propose a $(1+\varepsilon)$-approximation algorithm for the connecting part by showing a polynomial-time approximation scheme for a *Node-Weighted Steiner Tree* problem when the given terminal set is *c-local* and thus obtain a $(5+\varepsilon)$-approximation algorithm for an **MWCDS**.

1. Introduction

Due to the lack of predefined infrastructure, most routing protocols in wireless networks involve flooding, which usually causes a serious broadcasting storm [12]. The Connected Dominating Set (**CDS**) has become a well known approach for constructing a virtual backbone to alleviate this broadcasting storm in wireless networks. With the help of the **CDS**, the average message burden of the network could be reduced so that routing becomes much easier and can adapt quickly to network topology changes [4]. Furthermore, using a **CDS** as forwarding nodes can efficiently reduce the energy consumption, which is also a critical concern in wireless networks.

Given a graph $G = (V, E)$, a *Dominating Set* (**DS**) is a subset $D \subseteq V$ such that, for every vertex $v \in V$, either $v \in D$, or v has a neighbor in D. If the graph induced from D is connected, then D is called a *Connected Dominating Set* (**CDS**). The *Minimum Dominating Set* (**MDS**) problem is to find a dominating set in G with minimum size and the *Minimum Connected Dominating Set* (**MCDS**) problem is to find a connected dominating set in G with minimum size. Both problems are well-known NP-complete problems [6]. Lichtenstein [11] showed that they are NP-complete problems even though the given graph is a unit disk graph (UDG), which has a wide application in networks. A *unit disk graph* is associated with a set of unit disks in the Euclidean plane. Each vertex is the center of a unit disk. An edge exists between two vertices u and v if and only if the Euclidean distance between u and v is at most 1. The most common methodology researchers use to construct the

* Corresponding author.
** Corresponding author. Tel.: +1 4694503064.
E-mail addresses: phenix.zou@student.utdallas.edu (F. Zou), wangyuexuan@tsinghua.edu.cn (Y. Wang), xxu23@iit.edu (X.-H. Xu), lixianyue@lzu.edu.cn (X. Li), hongwei@cs.cityu.edu.hk (H. Du), wan@cs.iit.edu (P. Wan), weiliwu@utdallas.edu (W. Wu).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.06.022

approximation algorithms for an **MCDS** has the following two steps. First, find a **DS**, and then make this **DS** connected using either a Steiner tree or a spanning tree. Usually, we call this step connecting.

The Minimum-Weighted Connected Dominating Set problem (**MWCDS**) is a generalization of the **MCDS**. Given a graph $G = (V, E)$ with node weight function $C : V \to R^+$, the **MWCDS** problem is to find a **CDS** of G such that its total weight is minimum. Similarly, the Minimum-Weighted Dominating Set problem (**MWDS**) is a generalization of the **MDS**, which is to find a **DS** of G such that its total weight is minimum. For convenience, we normalize the weight function C such that, for any vertex v in G, $C(v) \geq 1$. If the weights on all vertices are the same, the **MWDS** and **MWCDS** problems are equal to the **MDS** and **MCDS** respectively. Hence, the **MWDS** and **MWCDS** problems are also NP-complete problems. Until now, the best known approximation ratio for an **MWCDS** in a general graph is $O(\log n)$ [7].

In this paper, we are concerned about **MWDSs** and **MWCDSs** in unit disk graphs, most of the construction methodologies of which adopted by researchers follow the routine for **MDSs** and **MCDSs**. Ambühl et al. [1] gave a 72-approximation algorithm for an **MWDS**. Meanwhile, by introducing a 17-approximation algorithm for the connecting part, they gave the first constant-factor algorithm for an **MWCDS** with approximation ratio 89. Huang et al. [9] improved the approximation ratio of an **MWCDS** from 89 to $10 + \varepsilon$ with a $(6 + \varepsilon)$-approximation algorithm for the **MWDS** and a 4-approximation algorithm for the connecting part. Recently, Dai and Yu [5] gave a $(5 + \varepsilon)$-approximation algorithm for an **MWDS** and Zou et al. [16] gave a 3.875-approximation algorithm for the connecting part by using a Steiner tree. Therefore, the best known approximation ratio so far for an **MWCDS** in a UDG is $8.875 + \varepsilon$.

In this paper, we first present a $(4 + \varepsilon)$-approximation algorithm for an **MWDS** based on a dynamic programming algorithm for the *Min-Weight Chromatic Disk Cover*. Suppose that we are given a set $\mathcal{D}$ of unit disks with positive weights given by a function c and a set P of nodes in the plane. Also each disk $D \in \mathcal{D}$ is colored either red or blue. In addition, suppose that we are given $m + 1 \geq 2$ distinct horizontal lines $y = y_i$ for $0 \leq i \leq m$ from the top to the bottom, where m is a nonnegative integer *constant*. Thus these $m + 1$ lines separate the plane into $m + 2$ horizontal strips. A node $p \in P$ is *chromatically covered* by a red (resp., blue) disk $D \in \mathcal{D}$ if $p \in D$ and the center of D is above (resp., below) the strip containing p. A subset $\mathcal{D}' \subseteq \mathcal{D}$ is said to be a *chromatic cover* of P if each node in P is chromatically covered by some disk in $\mathcal{D}'$. The *Min-Weight Chromatic Disk Cover* (**MWCDC**) problem seeks a min-weight chromatic cover $\mathcal{D}' \subseteq \mathcal{D}$ of P. We show that there exists a polynomial-time dynamic programming algorithm for the **MWCDC** and prove that given a ρ-approximation algorithm for the **MWCDC** and any fixed ε, there is a polynomial $(4\rho + \varepsilon)$-approximation algorithm for the **MWDS**.

Meanwhile, we also propose a $(1 + \varepsilon)$-approximation algorithm for the connecting part by showing a polynomial-time approximation scheme (**PTAS**) for the Node-Weighted Steiner Tree problem when the given terminal set is *c-local*. The *Node-Weighted Steiner Tree* problem (**NWST**) is a variation of the classical Steiner tree problem. Given a graph $G = (V, E)$ with node weight function $C : V \to R^+$ and a subset X of V, the node-weighted Steiner tree problem is to find a Steiner tree for the set X such that its total weight is minimum. We call the set X the *terminal set*. For any Steiner tree T for X and vertex $u \in V(T)$, we call u a *terminal vertex* if $u \in X$; otherwise, we call it a *Steiner vertex*. A *polynomial-time approximation scheme* (PTAS) is a family of approximation algorithms with ratio $1 + \varepsilon$ for any $\varepsilon > 0$. Hence, we obtain a $(5 + \varepsilon)$-approximation algorithm for the **MWCDS**, which is the best approximation ratio so far.

The rest of this paper is organized as follows. In Section 2, we give necessary notations and lemmas. In Section 3, we introduce the dynamic programming algorithm for the **MWCDC** as part of the main algorithm for the **MWDS**. Section 4 shows a PTAS algorithm for the **NWST**, and by using this to connect the **MWDS**, we can obtain a $(5 + \varepsilon)$-approximation algorithm for the **MWCDS**.

2. Preliminaries and fundamental lemmas

In this section, we introduce three variants of disk covers and their relations first as the fundamental lemmas for an **MWDS**. We then introduce some preliminaries for constructing an **NWST** to interconnect the **MWDS**.

2.1. Disk cover problems

Suppose that we are given a set $\mathcal{D}$ of unit disks with positive weights given by a function c and a set P of nodes in the plane. A node $p \in P$ is *covered* by a disk $D \in \mathcal{D}$ if $p \in D$. A subset $\mathcal{D}' \subseteq \mathcal{D}$ is said to be a *cover* of P if each node in P is covered by some disk in $\mathcal{D}'$. The *Min-Weight Disk Cover* (**MWDC**) problem seeks a min-weight cover $\mathcal{D}' \subseteq \mathcal{D}$ of P.

Now, suppose in addition that we are given $m + 1 \geq 2$ distinct horizontal lines $y = y_i$ for $0 \leq i \leq m$ from the top to the bottom, where m is a nonnegative integer *constant*. The set P of nodes lies between the topmost line and the bottommost line. None of the nodes in P and the centers of the disks in $\mathcal{D}$ lies on any of these $m + 1$ horizontal lines. These $m + 1$ lines separate the plane into $m + 2$ horizontal strips. A node $p \in P$ is *strongly covered* by a disk $D \in \mathcal{D}$ if $p \in D$ and the center of D does not lie in the same strip as p. A subset $\mathcal{D}' \subseteq \mathcal{D}$ is said to be a *strong cover* of P if each node in P is strongly covered by some disk in $\mathcal{D}'$. The *Min-Weight Strong Disk Cover* (**MWSDC**) problem seeks a min-weight strong cover $\mathcal{D}' \subseteq \mathcal{D}$ of P.

Next, suppose further that each disk $\mathcal{D}$ is colored either red or blue. A node $p \in P$ is *chromatically covered* by a red (resp., blue) disk $D \in \mathcal{D}$ if $p \in D$ and the center of D is above (resp., below) the strip containing p. A subset $\mathcal{D}' \subseteq \mathcal{D}$ is said to be a *chromatic cover* of P if each node in P is chromatically covered by some disk in $\mathcal{D}'$. The *Min-Weight Chromatic Disk Cover* (**MWCDC**) problem seeks a min-weight chromatic cover $\mathcal{D}' \subseteq \mathcal{D}$ of P.

Using the double partition and shifting techniques developed in [17], we can prove the following relation between the **MWSDC** and the **MWDC**.

Theorem 2.1. *Suppose that there exists a polynomial ρ-approximation algorithm for the **MWSDC**, then for any fixed $\varepsilon > 0$ there is a polynomial $(2\rho + \varepsilon)$-approximation algorithm for the **MWDC**.*

The following theorem further reduces the **MWSDC** to the **MWCDC**.

Theorem 2.2. *Suppose that there exists a polynomial algorithm for the **MWCDC**, then there is polynomial 2-approximation algorithm for the **MWSDC**.*

Proof. Consider an instance $(P, \mathcal{D}, c)$ of the **MWSDC**. We duplicate $\mathcal{D}$ into a red copy $\mathcal{R}$ and a blue copy $\mathcal{B}$. Let $\mathcal{R}' \cup \mathcal{B}'$ be a min-weight chromatic cover of P, and $\mathcal{D}' \subseteq \mathcal{D}$ be the set of disks in $\mathcal{D}$ for which the set of disks in $\mathcal{R}' \cup \mathcal{B}'$ is duplicated. We show that $\mathcal{D}'$ is a 2-approximation. Suppose that $\mathcal{D}^*$ is a min-weight strong cover of P. Let $\mathcal{R}^*$ (resp., $\mathcal{B}^*$) be the red (resp., blue) duplication of $\mathcal{D}^*$. Then, $\mathcal{R}^* \cup \mathcal{B}^*$ is a chromatic cover of P, and consequently,

$$c\left(\mathcal{D}'\right) \leq c\left(\mathcal{R}' \cup \mathcal{B}'\right) \leq c\left(\mathcal{R}^* \cup \mathcal{B}^*\right) = 2c\left(\mathcal{D}^*\right). \quad \blacksquare$$

In the next section, we will establish the following theorem on the **MWCDC**.

Theorem 2.3. *There exists a polynomial algorithm for the **MWCDC**.*

From these three theorems, we can come up with a $(4 + \varepsilon)$-approximation algorithm for the **MWDC**.

2.2. Node-weighted Steiner tree

Given a solution for the **MWDC**, we could consider all the vertices in the **MWDC** as terminals, and consider connecting them together as looking for an **NWST** for all these terminals.

Given a node-weighted graph $G = (V, E)$ with weight function C, and also the terminal set X we are interested in, we first introduce two kinds of distance between any two vertices u and v in the graph, which are called the *e*-distance and the *w*-distance, respectively. In detail, $dist_e(u, v)$ is calculated as the Euclidean distance between the two nodes and $dist_w(u, v)$ is calculated as the minimum weight of all the possible paths connecting u and v in G. The weight of each path here is calculated as the total weight of all intermediate vertices on that path.

Since the graph is node-weighted instead of edge-weighted, the construction of the minimum spanning tree, or saying the minimum node-weighted spanning tree on terminal set X (denoted as $T_s(X)$), is a little bit different here. First, we create an edge-weighted complete graph G' on terminal set X such that, for any edge (u, v) in G' $(u, v \in X)$, its weight is equal to the w-distance between u and v. Then let T_s be a minimum spanning tree of G'. It is easy to see that, for any edge (u, v) in T_s, it corresponds to the minimum-weighted path between u and v in G. In the following, we use $C(T_s)$ to denote the total weight of edges in T_s. Meanwhile, for simplicity, we keep T_s as it is without replacing the weighted edge with the corresponding minimum-weighted path between any two nodes in the node-weighted graph.

A set of vertices X is called *c-local* in a node-weighted graph if, in the minimum node-weighted spanning tree for X, the weight of the longest edge is at most c. This definition could be considered as the node-weighted version of the *c-local* definition given by [14]. In the remainder of the paper, we assume that the terminal set X is c-local for some constant c.

In [13], Robin et al. showed that the Minimum Spanning Tree Number for a Euclidean metric is 5, and [15] shows that, for any unit disk graph G, there exits a spanning tree T of G such that the maximum degree of T is at most 5. Thus, we can get the following lemma easily.

Lemma 2.4. *The minimum spanning tree of a given terminal set in a node-weighted graph has an approximation ratio of at most 5 for the optimal Steiner tree of the same given terminal set.*

3. Dynamic programming algorithm for the MWCDC

In this section, we present a dynamic programming algorithm for the **MWCDC**. We begin with some terms and notations.

By necessary preprocessing, we can assume that each disk in $\mathcal{D}$ chromatically covers at least one node in P. The $m + 1$ horizontal lines separate (w.r.t. the disk centers) the set $\mathcal{D}$ of disks into $m + 2$ subsets $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_m, \mathcal{D}_{m+1}$ from the top to the bottom. For each $0 \leq i \leq m + 1$, we denote by $\mathcal{R}_i$ (resp., $\mathcal{B}_i$) the set of red (resp., blue) disks in $\mathcal{D}_i$.

For ease of treatment, we also introduce m dummy red disks and m dummy blue disks as follows. For each $0 \leq i \leq m-1$, the half-plane $y \geq y_i$ defines a dummy red disk of zero weight; we denote by $\mathcal{R}_i^+$ the union of $\mathcal{R}_i$ and this red dummy disk. For each $2 \leq i \leq m + 1$, the half-plane $y \leq y_{i-1}$ defines a dummy blue disk of zero weight; we denote by $\mathcal{B}_i^+$ the union of $\mathcal{B}_i$ and this blue dummy disk. None of these $2m$ dummy disks chromatically cover any node in P, but each of them intersects every vertical line as a half-plane.

Consider a disk $D \in \mathcal{R}_i^+$ with $0 \leq i \leq m - 1$ intersecting a vertical line l. A disk $D' \in \mathcal{R}_i^+$ is said to be *dominated* by D at l if one of following cases occurs (see Fig. 1 for an illustration).

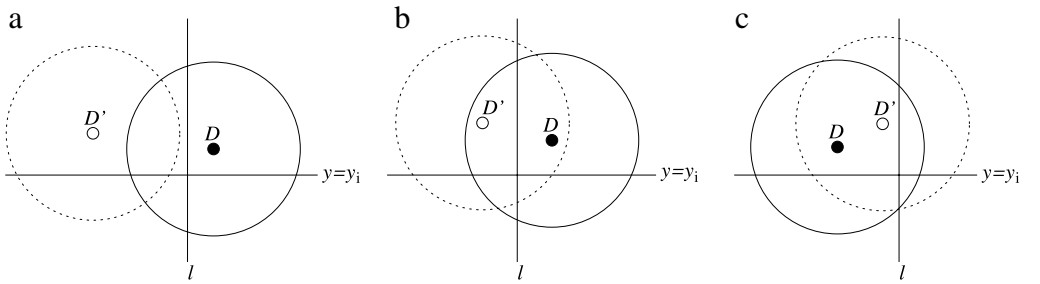

Fig. 1. Three scenarios for a red disk D to dominate another red disk D' at a vertical line l.

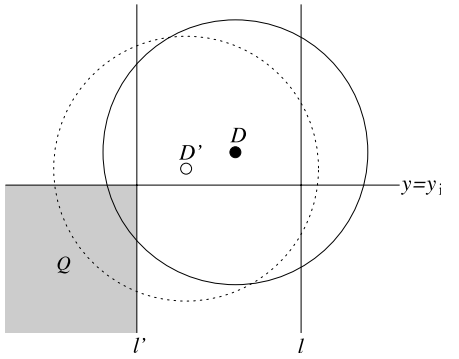

Fig. 2. If D dominates D' at l and D' dominates D at l', then $D \cap Q \subseteq D' \cap Q$.

1. D' does not intersect l;
2. The lower endpoint of $D \cap l$ is below the lower endpoint of $D' \cap l$;
3. $D \cap l$ and $D' \cap l$ have the same lower endpoint, but the center of D lies to the left to the center of D'.

Similarly, consider a disk $D \in \mathcal{B}_i^+$ with $2 \leq i \leq m+1$ intersecting a vertical line l. A disk $D' \in \mathcal{B}_i^+$ is said to be *dominated* by D at l if and only if one of following three conditions is satisfied:

1. D' does not intersect l;
2. The upper endpoint of $D \cap l$ is above the upper endpoint of $D' \cap l$;
3. $D \cap l$ and $D' \cap l$ have the same upper endpoint, but the center of D lies to the left to the center of D'.

It is easy to verify that the domination is *transitive*; i.e.,

- Suppose that D_1, D_2, D_3 are three red disks in $\mathcal{R}_i^+$ for some $0 \leq i \leq m-1$, and l is a vertical line. If D_1 dominates D_2 at l and D_2 dominates D_3 at l, then D_1 dominates D_3 at l.
- Suppose that D_1, D_2, D_3 are three blue disks in $\mathcal{B}_i^+$ for some $2 \leq i \leq m+1$, and l is a vertical line. If D_1 dominates D_2 at l and D_2 dominates D_3 at l, then D_1 dominates D_3 at l.

The following geometric fact was used in both [1,5].

Lemma 3.1. *Let l and l' be two distinct vertical lines with l lying on the right to l'.*

- *Suppose that D and D' are two red disks in $\mathcal{R}_i^+$ for some $0 \leq i \leq m-1$. If D dominates D' at l and D' dominates D at l', then $D \cap Q \subseteq D' \cap Q$, where Q is the (closed) lower-left quarter-plane bounded by the lines $y = y_i$ and l' (see Fig. 2 for an illustration).*
- *Suppose that D and D' are two blue disks in $\mathcal{B}_i^+$ for some $2 \leq i \leq m+1$. If D dominates D' at l and D' dominates D at l', then $D \cap Q \subseteq D' \cap Q$, where Q is the (closed) upper-left quarter-plane bounded by the lines $y = y_i$ and l'.*

Suppose that $\mathcal{D}'$ is a set of disks which includes all dummy disks and l is a vertical line. For each $0 \leq i \leq m-1$, the i-th *red skyline disk* of $\mathcal{D}'$ at l is the disk $D \in \mathcal{D}' \cap \mathcal{R}_i^+$ which dominates all other red disks, if there are any, in $\mathcal{D}' \cap \mathcal{R}_i^+$. For each $2 \leq i \leq m+1$, the i-th *blue skyline disk* of $\mathcal{D}'$ at l is the disk $D \in \mathcal{D}' \cap \mathcal{B}_i^+$ which dominates all other blue disks, if there are any, in $\mathcal{D}' \cap \mathcal{B}_i^+$. The *skyline* of $\mathcal{D}'$ at l is defined to be sequence of m red skyline disks at l followed by the sequence of m blue skyline disks at l.

Let $l_1, l_2, \ldots, l_n$ be the set of vertical lines through P from left to right. Let $P_0 = \emptyset$. For each $1 \leq k \leq n$, we use P_k to denote the set of nodes in P lying on the (closed) left half-plane separated by l_k, and use $\mathcal{R}_{i,k}^+$ (resp., $\mathcal{B}_{i,k}^+$) to denote the set of red disks in $\mathcal{R}_i^+$ (resp., $\mathcal{B}_i^+$) intersecting the line l_k. Denote

$$\Gamma_k = \left(\prod_{i=0}^{m-1} \mathcal{R}_{i,k}^+\right) \times \left(\prod_{i=2}^{m} \mathcal{B}_{i,k}^+\right).$$

For each $\mathbf{D} \in \Gamma_k$, $\mathcal{C}_k(\mathbf{D})$ denotes the collection of chromatic disk covers $\mathcal{D}'$ of P_k with $\mathbf{D}$ as the skyline at l_k and containing all the $2m$ dummy disks. If $\mathcal{C}_k(\mathbf{D})$ is non-empty, let $C_k(\mathbf{D})$ be a min-weight cover in $\mathcal{C}_k(\mathbf{D})$, and $c_k(\mathbf{D})$ be the weight of $C_k(\mathbf{D})$; otherwise, set $C_k(\mathbf{D})$ to *null*, and set $c_k(\mathbf{D})$ to ∞. Clearly,

- if $\mathbf{D}$ is a chromatic cover of P_k, then $C_k(\mathbf{D}) = \mathbf{D}$ and $c_k(\mathbf{D}) = c(\mathbf{D})$;
- if $\mathbf{D}$ is not a chromatic cover of $P_k \setminus P_{k-1}$, then $C_k(\mathbf{D}) = null$ and $c_k(\mathbf{D}) = \infty$.

Suppose that $2 \leq k \leq n$. For any $\mathbf{D} \in \Gamma_k$, denote by $\Gamma_{k-1}(\mathbf{D})$ the set of $\mathbf{D}' \in \Gamma_{k-1}$ satisfying that each component disk in $\mathbf{D}'$ is dominated by the corresponding component disk of $\mathbf{D}$ at l_k. Then, we have the following recursive relation.

Lemma 3.2. *Suppose that $2 \leq k \leq n$. For any $\mathbf{D} \in \Gamma_k$ which is a chromatic cover of $P_k \setminus P_{k-1}$,*

$$c_k(\mathbf{D}) = \min_{\mathbf{D}' \in \Gamma_{k-1}(\mathbf{D})} c_{k-1}(\mathbf{D}') + c(\mathbf{D} \setminus \mathbf{D}').$$

Proof. First, we show that

$$c_k(\mathbf{D}) \geq \min_{\mathbf{D}' \in \Gamma_{k-1}(\mathbf{D})} c_{k-1}(\mathbf{D}') + c(\mathbf{D} \setminus \mathbf{D}').$$

This holds trivially if $c_k(\mathbf{D}) = \infty$. So we assume that $c_k(\mathbf{D}) < \infty$ is finite. Let $\mathbf{D}'$ be the skyline of $C_k(\mathbf{D})$ at the line l_{k-1}. Then, $\mathbf{D}' \in \Gamma_{k-1}(\mathbf{D})$. Set

$$\mathcal{D}' = C_k(\mathbf{D}) \setminus (\mathbf{D} \setminus \mathbf{D}').$$

Then

$$c(\mathcal{D}') = c_k(\mathbf{D}) - c(\mathbf{D} \setminus \mathbf{D}')$$

and $\mathbf{D}'$ is the skyline of $\mathcal{D}'$ at the line l_{k-1}. We claim that $\mathcal{D}'$ is a chromatic cover of P_{k-1}. Assume to the contrary that some point $p \in P_{k-1}$ is not chromatically covered by $\mathcal{D}'$. Then, p is chromatically covered by some disk $D \in \mathbf{D} \setminus \mathbf{D}'$. By symmetry, we assume that D is a red disk. Suppose that $D \in \mathcal{R}_{i,k}$. Let D' be the component disk of $\mathbf{D}'$ in $\mathcal{R}_{i,k-1}^+$. Then D and D' are distinct, D dominates D' at l_k, and D' dominates D at l_{k-1}. By Lemma 3.1, p is also chromatically covered by D', which is a contradiction. Therefore, our claim holds. Hence $\mathcal{D}' \in \mathcal{C}_{k-1}(\mathbf{D}')$. So

$$c_{k-1}(\mathbf{D}') \leq c(\mathcal{D}') = c_k(\mathbf{D}) - c(\mathbf{D} \setminus \mathbf{D}'),$$

which implies

$$c_k(\mathbf{D}) \geq c_{k-1}(\mathbf{D}') + c(\mathbf{D} \setminus \mathbf{D}').$$

Secondly, we prove that

$$c_k(\mathbf{D}) \leq \min_{\mathbf{D}' \in \Gamma_{k-1}(\mathbf{D})} c_{k-1}(\mathbf{D}') + c(\mathbf{D} \setminus \mathbf{D}').$$

This inequality holds trivially if the right-hand side is ∞. So we assume that the right-hand side is finite. Suppose that the right-hand side achieves its minimum at $\mathbf{D}' \in \Gamma_{k-1}(\mathbf{D})$. Let

$$\mathcal{D}' = C_{k-1}(\mathbf{D}') \cup \mathbf{D}.$$

Since $\mathbf{D}$ is a chromatic cover of $P_k \setminus P_{k-1}$, $\mathcal{D}'$ is a chromatic cover of P_k. We prove the following two claims:
Claim 1: For any component disk D of $\mathbf{D}$ which is not a dummy, $D \notin C_{k-1}(\mathbf{D}') \setminus \mathbf{D}'$.
Claim 2: $\mathbf{D}$ is the skyline of $\mathcal{D}'$ at l_k.
Claim 1 implies that

$$c(\mathcal{D}') = c_{k-1}(\mathbf{D}') + c(\mathbf{D} \setminus \mathbf{D}').$$

Claim 2 implies that $\mathcal{D}' \in \mathcal{C}_k(\mathbf{D})$. Consequently,

$$c_k(\mathbf{D}) \leq c(\mathcal{D}') = c_{k-1}(\mathbf{D}') + c(\mathbf{D} \setminus \mathbf{D}').$$

We prove **Claim 1** by contradiction. Assume to the contrary that **Claim 1** does not hold. By symmetry, we assume that for some red component disk D of $\mathbf{D}$ which is not a dummy, $D \in C_{k-1}(\mathbf{D}') \setminus \mathbf{D}'$. Suppose that $D \in \mathcal{R}_{i,k}$. Let D' be the component disk of $\mathbf{D}'$ in $\mathcal{R}_{i,k-1}^+$. Then D and D' are distinct. So, D dominates D' at l_k and D' dominates D at l_{k-1}. By Lemma 3.1, any node in

P_{k-1} chromatically covered by D is also chromatically covered by D'. Hence $C_{k-1}\left(\mathbf{D}'\right) \setminus \{D\}$ would still be a chromatic cover of P_{k-1} in $\mathcal{C}_{k-1}\left(\mathbf{D}'\right)$, which contradicts the minimality of $C_{k-1}\left(\mathbf{D}'\right)$ in $\mathcal{C}_{k-1}\left(\mathbf{D}'\right)$. Thus, **Claim 1** holds.

We also prove **Claim 2** by contradiction. Assume to the contrary that **Claim 2** does not hold. By symmetry, we assume that some red component disk D of $\mathbf{D}$ is not a skyline disk of $\mathcal{D}'$ at l_k. Then there is a disk $D'' \in \mathcal{R}_i \cap \mathcal{D}'$ which dominates D at l_k. Let D' the component disk of $\mathbf{D}'$ in $\mathcal{R}^+_{i,k-1}$. Then, D' and D'' are distinct. Note that either $D = D'$ or D dominates D' at l_k. In either case, D'' dominates D' at l_k. On the other hand, D' dominates D'' at l_{k-1}. By Lemma 3.1, any node in P_{k-1} chromatically covered by D'' is also chromatically covered by D'. Hence, $C_{k-1}\left(\mathbf{D}'\right) \setminus \left\{D''\right\}$ would still be a chromatic cover of P_{k-1} in $\mathcal{C}_{k-1}\left(\mathbf{D}'\right)$, which contradicts the minimality of $C_{k-1}\left(\mathbf{D}'\right)$ in $\mathcal{C}_{k-1}\left(\mathbf{D}'\right)$. Thus, **Claim 2** holds. ■

Now, we are ready to present the dynamic programming for the **MWCDC**. For each $1 \leq k \leq n$, we build a table of $|\Gamma_k|$ entries, with each entry corresponding to an element of Γ_k. The first table is constructed as follows. For each $\mathbf{D} \in \Gamma_1$, if $\mathbf{D}$ is a chromatic cover of P_1, set

$$C_1(\mathbf{D}) = \mathbf{D}, \qquad c_1(\mathbf{D}) = c(\mathbf{D});$$

otherwise, set

$$C_1(\mathbf{D}) = null, \qquad c_1(\mathbf{D}) = \infty.$$

Now suppose that the first $k-1$ tables have been constructed for some $2 \leq k \leq n$. We construct the k-th table as follows. For each $\mathbf{D} \in \Gamma_k$,

- if $\mathbf{D}$ is a chromatic cover of P_k, set

  $$C_k(\mathbf{D}) = \mathbf{D},\ c_k(\mathbf{D}) = c(\mathbf{D});$$

- if $\mathbf{D}$ is not a chromatic cover of $P_k \setminus P_{k-1}$, set

  $$C_k(\mathbf{D}) = null,\ c_k(\mathbf{D}) = \infty;$$

- otherwise, compute a $\mathbf{D}' \in \Gamma_{k-1}(\mathbf{D})$ minimizing $c_{k-1}\left(\mathbf{D}'\right) + c\left(\mathbf{D} \setminus \mathbf{D}'\right)$. If $c_{k-1}\left(\mathbf{D}'\right) = \infty$, set

  $$C_k(\mathbf{D}) = null,\ c_k(\mathbf{D}) = \infty;$$

  otherwise, set

  $$C_k(\mathbf{D}) = C_{k-1}\left(\mathbf{D}'\right) \cup \mathbf{D},\ c_k(\mathbf{D}) = c_{k-1}\left(\mathbf{D}'\right) + c\left(\mathbf{D} \setminus \mathbf{D}'\right).$$

Next, we compute a $\mathbf{D} \in \Gamma_n$ minimizing $c_n(\mathbf{D})$. Removing all dummy disks from $C_n(\mathbf{D})$, the remaining set of disks is a min-weight chromatic cover of P.

Thus, with this algorithm, we could obtain the following theorem.

Theorem 3.3. *There is a polynomial-time $(4+\varepsilon)$-approximation algorithm for the minimum-weighted dominating set problem.*

Proof. We observe that the time complexity of the above algorithm for the **MWCDC** depends on m and number of disks in $\mathcal{D}$. According to the techniques used in [17], m depends on ε; thus this algorithm is polynomial-time computable according to the number of disks in $\mathcal{D}$. According to Theorems 2.1 and 2.3, we can conclude that there exists a polynomial-time $(4+\varepsilon)$-approximation algorithm for the minimum-weighted dominating set problem. ■

4. PTAS for the NWST

Recall that the **MWCDS** problem is to construct the connected dominating set in a node-weighted graph with the minimum total weight. Normally, researchers start by calculating the dominating sets for the graph first and then interconnecting them. Obviously, the node-weighted Steiner tree can be used in the **MWCDS** problem to interconnect all nodes of the **DS** to get a better approximation algorithm. We propose a **PTAS** for the **NWST** in this section. We present our approximation algorithm based on the partition and shifting strategy. Before introducing this algorithm, we first give some useful notations.

Recall that T_s is a minimum spanning tree of terminal set X in G. For a fixed partition P, we call an edge uv a *crossing edge* if at least one of the end nodes u or v is contained in the boundary area of P. We use X_p to denote the set of terminal vertices contained in the interior area of P. Note that we study this problem under a fixed partition P in this section.

The algorithm has two steps, as follows. First, for each cell, we construct a local optimal Steiner forest on terminal vertices in the interior area of this cell. Then, combine all these forests to obtain a local optimal Steiner forest $\hat{F}_p$ on X_p. In the second step, we add all the crossing edges in T_s into $\hat{F}_p$ to get a Steiner tree on terminal set X. We call the resulting graph G_p for a specific partition P. In order to approximate the minimum node-weighted Steiner tree, we calculate all $G_{p_{i,j}}$ for $0 \leq i, j \leq k-1$ and choose the minimum one among all of them as the output of our algorithm.

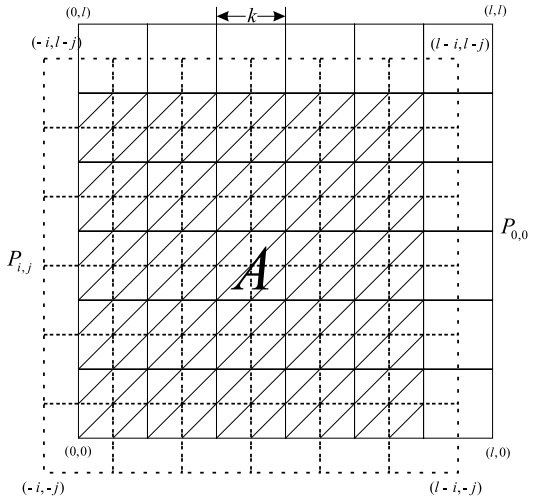

Fig. 1. Two partitions $P_{0,0}$ and $P_{i,j}$ with the shadow area A. The solid lines represent partition $P_{0,0}$ and the dashed lines represent partition $P_{i,j}$.

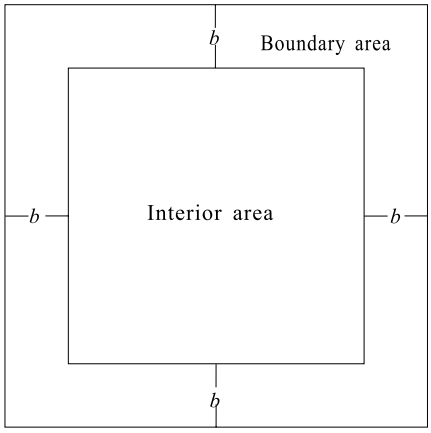

Fig. 2. The interior area and boundary area with boundary width $b = (1 + 1.5 \log k)c$.

4.1. Partition and shifting

One of the key strategies adopted in our algorithm is partition and shifting. Considered as a special way to make a restriction and derandomize the probabilistic result to get a deterministic one, researchers started to use a partition and shifting strategy [2,8] in approximation algorithms from the early 1980s.

Specifically, in our algorithm, we partition the graph according to the following strategy. Let A be the smallest square containing all vertices of G with size $q \times q$. For a given integer k, let $l = (\lfloor q/k \rfloor + 2)k$ and make the lower left corner of the square A the center of the coordinate system. We extend the area A to A' of size $l \times l$ and divide it into small cells such that the size of each cell is $k \times k$ (see Fig. 1). Furthermore, for each cell, we divide it into two parts: the interior area and the boundary area with boundary width $b = (1 + 1.5 \log k)c$ (as in Fig. 2). We call this partition $P_{0,0}$. Then we shift the extended area A' to make its lower left corner positioned at point $(-i, -j)$ $(0 \leq i, j \leq k - 1)$ in the coordinate system, to get another partition $P_{i,j}$. Clearly, there are k^2 possible partitions and any partition contains the area A.

Our intention of making use of the partition and shifting strategy is that, for any fixed partition, we first construct the local optimal solution for each cell. Then, we further modify the union of the local optimal solution of all cells to make it a feasible solution. In order to achieve the best solution, we use shifting to obtain a set of solutions on different partitions and choose the best solution among all these feasible solutions. With this strategy, we could better bound the approximation ratio of our algorithm.

We now describe in detail the construction of the local optimal Steiner forest and the final Steiner tree in our scheme.

4.1.1. Local optimal Steiner forest $\hat{F}_p$

Our target in this part is to construct a local optimal Steiner forest $\hat{F}_p$ on X_p. In order to achieve this goal, we first group the terminal vertices in the interior area of every cell satisfying that the w-distance between any two groups is greater than c. The grouping is achieved by first constructing the minimum spanning tree on the terminal vertices in the interior area of

each cell and then deleting all edges with weight greater than c. Obviously, by doing so, terminal vertices will be divided into different connected components. We consider all terminal vertices in the same connected component to be in the same group. Clearly, the w-distance between any two groups is greater than c. Otherwise, there will be another spanning tree with weight less than our minimum spanning tree, which creates a contradiction.

For a fixed cell, let $Y_1, \ldots, Y_m$ be the different groups of all terminal vertices after grouping. In order to get desired solution, we merge $Y_1, \ldots, Y_m$ into new groups, construct the Steiner minimum tree for each new group in this cell, and then combine them to form a Steiner forest. If we calculate the total weight of the edges (the w-distance between two end points) in the resulting Steiner forest to be the cost of this specific merging, with different possible merging choices, we choose the merging with the minimum merging cost among all of them. The corresponding Steiner forest is the local optimal Steiner forest that we are after for this cell in partition P.

For a fixed partition P, we denote $\hat{F}_p$ the local optimal Steiner forest on the terminal vertices X_p in graph G. It is calculated as the union of the local optimal Steiner forest in each cell. From the method for $\hat{F}_p$ construction described above, we can obtain the following lemma easily.

Lemma 4.1. *$\hat{F}_p$ is a Steiner forest on X_p with the following properties:*

(1) *Each tree in the forest $\hat{F}_p$ is completely included in some cell.*

(2) *The w-distance between any two terminal vertices in different trees of $\hat{F}_p$ is greater than c.*

In the following, we will discuss the running time for computing a local optimal Steiner forest. Let n be the number of vertices of G. Since G is a unit disk graph, we can see that G can be covered by a square with size $n \times n$. Recall that the size of every cell is $k \times k$; there are at most $O(n/k)^2$ cells. Then, we will discuss the time for computing a local optimal Steiner forest in a cell. Let Y be the set of terminal vertices in the interior area in this cell and m be the number of groups in the same cell. In the subgraph $G[Y]$ induced by Y, we shrink each component to be a new vertex and set Y' as the set of these new vertices. It is easy to see that $m \leq |Y'|$. If we find a minimum Steiner tree on Y', and replace every vertex in Y' by the corresponding component, we obtain a minimum Steiner tree on Y. Hence, the time complexity to compute the local optimal Steiner forest is $O(2^m M(|Y'|))$ [14], where $M(|Y'|)$ is the time to compute an optimal Steiner tree on terminal set Y' and $M(|Y'|)$ is exponential in $|Y'|$ [3,10]. If we divide the cell into some squares such that the size of each square is $\frac{\sqrt{2}}{2} \times \frac{\sqrt{2}}{2}$, then the terminal vertices in each square must belong to the same component of $G[Y]$. Hence, there are at most $2k^2$ components in $G[Y]$; i.e., $|Y'| \leq 2k^2$. In Section 4.2, we will show that k is only related with c and ε. Hence, we can compute a local optimal Steiner forest in polynomial time.

4.1.2. Constructing the NWST T_{out} from $\hat{F}_p$

Recall that in the above subsection, we get a local optimal Steiner forest $\hat{F}_p$ on X_p and the w-distance between any two terminal vertices in different trees of $\hat{F}_p$ is greater than c.

Let E_p^s be the set of all crossing edges in T_s under a partition P. In order to interconnect the disconnected components in the Steiner forest $\hat{F}_p$, we add all edges in E_p^s into $\hat{F}_p$ and then replace every crossing edge by the corresponding path in G. Denoting G_p as the resulting graph, we have

Lemma 4.2. *G_p contains a Steiner tree interconnecting X.*

Proof. In order to prove this statement, it is sufficient to show (1) $X \subseteq V(G_p)$; (2) G_p is connected.

Obviously, vertices in the interior area of a partition $X_p \subseteq V(G_p)$. If a vertex is in the boundary area of a partition, it must be on one of the crossing edges included in the set E_p^s, which has already been added into $V(G_p)$. So $X \subseteq V(G_p)$. It is sufficient to show G_p is connected. For convenience, we keep G_p as it is without replacing every crossing edge by the corresponding path; i.e., G_p is obtained from $\hat{F}_p$ by adding all edges in E_p^s. Clearly, if this G_p is connected, after replacing every crossing edge, the resulting graph G_p is also connected.

Now, suppose to the contrary that G_p is disconnected. Then, G_p can be divided into two disjoint subgraphs G_p^1 and G_p^2 such that there are no edges connecting G_p^1 and G_p^2 in G_p. Since G_p is obtained from $\hat{F}_p$ by adding all edges in E_p^s, there are some terminal vertices contained in G_p^1 and G_p^2. Since T_s is a spanning tree of terminal set X in G, there is an edge L in T_s connecting G_p^1 and G_p^2. Because all crossing edges are added in G_p, the edge L must be a non-crossing edge. Therefore, L is contained in some cell. Denote u and v as the endpoints of this edge. Also let T_u and T_v be the two trees containing u and v in the cell, respectively. Since c is the maximum edge weight among all edges in T_s, we have $dist_w(u, v) \leq c$. On the other hand, from Lemma 2, we have $dist_w(u, v) > c$. This creates a contradiction. Hence, G_p is connected. ■

Recall that there are all together k^2 different partitions, and for every partition $P_{i,j}$, we could obtain a graph $G_{P_{i,j}}$. Among all k^2 graphs, we choose the minimum-weight graph and prune it into a Steiner tree on X. This final tree, denoted as T_{out}, is the output of our algorithm.

4.2. Theoretical analysis

In this section, we study the approximation ratio of our algorithm and show that, for any $\varepsilon > 0$, choosing the appropriate integer k, the approximation ratio is $1 + \varepsilon$.

There are two steps in our proof. In the first step, we show that, for any partition P, $C(\hat{F}_P) \leq C(T_{OPT})$, where T_{OPT} is the optimal solution for node-weighted Steiner tree on terminal set X. In the second step, we show that our algorithm has a performance ratio of $1 + \varepsilon$.

4.2.1. $C(\hat{F}_P) \leq C(T_{OPT})$

Let T_p be the minimum Steiner tree in G on X_p. Since T_{OPT} is also a Steiner tree on X_p, clearly $C(T_p) \leq C(T_{OPT})$. In order to prove $C(\hat{F}_P) \leq C(T_{OPT})$, we construct a new Steiner forest F_p on X_p, which is modified from T_p satisfying that each tree in F_p is completely included in a cell and also $C(\hat{F}_p) \leq C(F_p)$. The following gives some useful notations for further proof.

For any Steiner tree, we call a Steiner vertex a *real Steiner vertex* if its degree is at least 3. Besides, a path between two vertices in the Steiner tree is a *Stem* if its endpoints are either a terminal vertex or a real Steiner vertex and also all the other vertices are 2-degree Steiner vertices. We modify T_p to be the desired forest F_p with the following three steps.

In the first step, we delete all stems with weight greater than c in T_p, and denote the resulting forest by F'_p. After this, the w-distance between any two trees in F'_p is greater than c because of the optimality of T_p. Also we have $C(F'_p) \leq C(T_p)$.

In the second step, we further modify F'_p to guarantee that each tree in it is interconnecting terminal vertices in the same cell. If there is a tree T^* in F'_p connecting terminal vertices in different cells, T^* must have some Steiner vertices between the boundary areas of two adjacent cells since the e-distance between any two vertices is at most 1. By Steiner vertices, we mean those vertices not belong to the X_p. If we draw two vertical lines, the distance between which is $2c$ as illustrated in Fig. 3, there must exist a vertex u in the tree T^* within these two lines since the e-distance between any two vertices is at most 1. Therefore, the e-distance between u and any boundaries of the interior area is more than $1.5c \log k$. Since the weight of any stem in F'_p is at most c and the weight of any vertex is at least 1, the e-distance between any two adjacent real Steiner vertices is at most c, where two real Steiner vertices are *adjacent* if they can be connected without any other real Steiner vertices. Now, we count the number of real Steiner vertices to connect the vertex u and any boundary of the interior area. Clearly, it must use at least $1 + 2 + \cdots + 2^{(1.5 \log k)-1} = 2^{1.5 \log k} - 1 = k^{1.5} - 1$ real Steiner vertices in the tree T^* (see Fig. 3). Hence, there are at least $k^{1.5} - 1 + (k^{1.5} - 1)(c - 1) = c(k^{1.5} - 1)$ Steiner vertices in the tree between u and the boundary of the interior area. By deleting all of these vertices, at most k more trees will be created along this boundary. If the w-distance of any two trees is at most c, connect them to create a new tree. As there are at most k trees, the whole weight will increase by at most $c(k - 1)$. Meanwhile, as we delete at least $c(k^{1.5} - 1)$ vertices, which means the whole weight decreases by at least $c(k^{1.5} - 1)$. Hence, the weight of the new F'_p is decreased by doing so. Repeat this step until there are no trees connecting different cells, denoting the resulting forest by F''_p. We can see that the w-distance between any two trees in F''_p is also greater than c and $C(F''_p) \leq C(F'_p)$.

In the last step, if any tree of F''_p is completely included in a cell, we do nothing. Otherwise, there must exist a tree such that all its terminal vertices are in a same cell, but at least one Steiner vertex is in a different cell. In this case, we modify F''_p using the same method described in Step 2. Clearly, any tree in the new F''_p is completely in one cell. Finally, for any tree, we reconstruct the Steiner minimum tree on its terminal vertices in the cell. Letting F_p be the resulting graph afterwards, we can obtain the following lemmas.

Lemma 4.3. *F_p is a Steiner forest on T_p with the following properties:*
(1) *Each tree of F_p is completely included in some cell.*
(2) *The w-distance between any two trees in F_p is greater than c. Furthermore, the w-distance between any two terminal vertices in different trees of F_p is greater than c.*
(3) $C(F_p) \leq C(T_p) \leq C(T_{OPT})$.

Lemma 4.4. $C(\hat{F}_p) \leq C(F_p) \leq C(T_{OPT})$.

Proof. It is only necessary to show $C(\hat{F}_p) \leq C(F_p)$. Recalling the constructions of $\hat{F}_p$ and F_p, for a fixed cell, every Y_i is completely contained in one tree of F_p. Hence, F_p will be one of possible merging solutions as well. Since $\hat{F}_p$ is the minimum solution of all possible merging choices, we have $C(\hat{F}_p) \leq C(F_p)$. ■

4.2.2. Performance analysis

Based on Lemma 4.4 and the construction of T_{out}, we obtain the main theorem in this paper.

Theorem 4.5. *The approximation ratio for the **NWST** problem used in our algorithm to interconnect the terminal set is* $1 + 40c\lceil 1 + 1.5 \log k \rceil / k$.

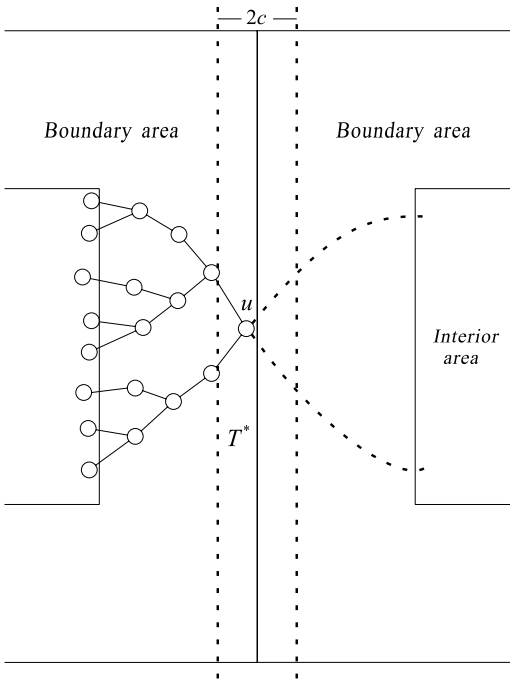

Fig. 3. The tree T^* and the vertex u.

Proof. Recall that T_{out} consists of two parts, the local optimal $\hat{F}_p$ and $E_{p_{i,j}}$. To bound the total weight of the $E_{p_{i,j}}$, we consider the number of times each vertex in the terminal set appears in the boundary area in all k^2 partitions.

If we divide every cell into 1×1 squares, for different partitions, the same terminal vertex must lie in different squares according to the shifting strategy we used. Since there are at most $4ck\lceil 1+1.5\log k\rceil$ squares in the boundary area, a terminal vertex will appear in the boundary area at most $4ck\lceil 1+1.5\log k\rceil$ times. For an edge in T_s, since both of its endpoints are terminal vertices, it will be considered as a crossing edge at most $2\times 4ck\lceil 1+1.5\log k\rceil$ times in all k^2 partitions. Hence, we have

$$\begin{aligned}\sum_{0\le i,j\le k-1} C(G_{p_{i,j}}) &\le \sum_p \hat{F}_p + \sum_{0\le i,j\le k-1} C(E_{p_{i,j}})\\ &\le k^2 C(T_{OPT}) + \sum_{0\le i,j\le k-1} C(E_{p_{i,j}})\\ &\le k^2 C(T_{OPT}) + 8ck\lceil 1+1.5\log k\rceil C(T_s)\\ &\le k^2 C(T_{OPT}) + 8ck\lceil 1+1.5\log k\rceil \times 5C(T_{OPT})\\ &\le k^2 C(T_{OPT}) + 40ck\lceil 1+1.5\log k\rceil C(T_{OPT}).\end{aligned}$$

Therefore, $C(T_{out}) \le (\sum_{0\le i,j\le k-1} C(G_{p_{i,j}}))/k^2 \le (1+40c\lceil 1+1.5\log k\rceil/k)\ C(T_{OPT})$. ■

Corollary 4.6. *For any given $\varepsilon > 0$, let $k > \lceil (41\,c/\varepsilon)^2\rceil$. Then $C(T_{out}) \le (1+\varepsilon)\ C(T_{OPT})$.*

*4.3. $(5+\varepsilon)$-approximation for the **MWCDS***

We apply the **NWST** algorithm into the **MWCDS** problem to interconnect all nodes of the **MWDS** to get a better approximation algorithm. Therefore, we can obtain the following results for the **MWCDS**.

Corollary 4.7. *There is a $(5+\varepsilon)$-approximation algorithm for the **MWCDS** by using a node-weighted Steiner tree to interconnect all nodes of the **MWDS**.*

Proof. For any node-weighted graph G and a given Dominating Set **MWDS** of G, denote by OPT_{CDS} and T_{OPT} the optimal **MWCDS** of G and the optimal Steiner tree of G on the given **MWDS**, respectively. Since the induced graph $G[DS \bigcup OPT_{CDS}]$ is connected, this graph contains a Steiner tree of G on the **MWDS**. Hence, we have $C(T_{OPT}) \le C(DS) + C(OPT_{CDS})$.

By Theorem 3.3, for any $\varepsilon > 0$, we can obtain a dominating set D of G with $C(D) \le (4+\varepsilon/7)\ C(OPT_{CDS})$. Then, using our algorithm for D, we can get a Steiner tree T interconnecting D with $C(T) \le (1+\varepsilon/7)\ C(T_{OPT})$. Since D is a dominating set,

clearly, $V(T)$ is a connected dominating set of G and

$$\begin{aligned} C(T) &\leq (1+\varepsilon/7)\ C(T_{OPT}) \\ &\leq (1+\varepsilon/7)\ (\ C(D)+C(OPT_{CDS})\ ) \\ &\leq (1+\varepsilon/7)(4+\varepsilon/7)\ C(OPT_{CDS})+(1+\varepsilon/7)C(OPT_{CDS}) \\ &\leq (4+6\varepsilon/7)\ C(OPT_{CDS})+(1+\varepsilon/7)C(OPT_{CDS}) \\ &\leq (5+\varepsilon)\ C(OPT_{CDS}). \quad \blacksquare \end{aligned}$$

5. Conclusion and discussion

In this paper, we first propose a $(4+\varepsilon)$-approximation algorithm for the **MWDS** based on a polynomial-time dynamic programming algorithm for the **MWCDC**. Adopting the strategy of partition and shifting, we also propose a $(1+\varepsilon)$-approximation algorithm for the **NWST** problem in unit disk graphs, which is the best solution for this problem we could ever have without proving $P = NP$. Based on it, we give a $(5+\varepsilon)$-approximation solution for the **MWCDS** problem in unit disk graphs when the given terminal set is *c-local* afterwards, by interconnecting the MWDS constructed, which better bounds the performance of the **MWCDS** compared with existing algorithms.

Acknowledgements

Dr. Yuexuan Wang was supported in part by the National Basic Research Program of China Grants 2007CB807900 and 2007CB807901, the NSFC under Grant 60604033, and the Hi-Tech Research Development Program of China Grant 2006AA10Z216. Dr. Pengjun Wan was supported in part by the NSF under grant CNS-0831831.

References

[1] C. Ambühl, T. Erlebach, M. Mihalák, M. Nunkesser, Constant-factor approximation for minimum-weight (connect) dominating sets in unit disk graphs, Lecture Notes in Computer Science 4110 (2006) 3–14.
[2] B.S. Baker, Approximation algorithm for NP-complete problems on planar graphs, Journal of the ACM 41 (1994) 153–180.
[3] D. Chen, D.Z. Du, X.D. Hu, G.H. Lin, L. Wang, G. Xue, Approximation for Steiner trees with minimum number of Steiner points, Theoretical Computer Science 262 (2001) 83–99.
[4] B. Das, V. Bharghavan, Routing in ad hoc networks using minimum connected dominating sets, in: International Conference on Communications, 1997.
[5] D. Dai, C. Yu, A $(5+\varepsilon)$-approximation algorithm for minimum weighted dominating set in unit disk graph, Theoretical Computer Science (2009).
[6] M.R. Garey, D.S. Johnson, Computers and Intractability. A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[7] S. Guha, S. Khuller, Improved methods of approximating node weighted Steiner tree and connected dominating sets, Information and Computation 150 (1999) 57–74.
[8] D.S. Hochbaum, W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI, Journal of the ACM 32 (1985) 130–136.
[9] Y. Huang, X. Gao, Z. Zhang, W. Wu, A better constant-factor approximation for weighted dominating set in unit disk graph, Journal of Combinatorial Optimization (2008).
[10] F.K. Hwang, D.S. Richards, Steiner tree problems, Networks 22 (1992) 55–89.
[11] D. Lichtenstein, Planar formulae and their uses, SIAM Journal on Computing 11 (1982) 329–343.
[12] S. Ni, Y. Tseng, Y. Chen, J. Sheu, The broadcast storm problem in a mobile ad hoc network, in: MOBICOM'99, Washington, USA, August 1999, pp. 152–162.
[13] G. Robins, J.S. Salowe, On the maximum degree of minimum spanning trees, in: Proceedings of the ACM Symposium on Computational Geometry, 1994, pp. 250–258.
[14] L. Wang, T. Jiang, An approximation scheme for some Steiner tree problem in the plane, Networks 28 (1996) 187–193.
[15] W. Wu, H. Du, X. Jia, Y. Li, S.C.H. Huang, Minimum connected dominating sets and maximal independent sets in unit disk graphs, Theoretical Computer Science 352 (2006) 1–7.
[16] F. Zou, X. Li, D. Kim, W. Wu, Node-weighted Steiner tree approximation in unit disk graphs, Journal of Combination Optimization (2009).
[17] Z. Zhang, Y. Huang, X. Gao, W. Wu, A better constant-factor approximation for weighted dominating set in unit disk graphs, Journal of Combinatorial Optimization (2008) 1573–2886.