# A PTAS for Node-Weighted Steiner Tree in Unit Disk Graphs

**Conference Paper** · June 2009

DOI: 10.1007/978-3-642-02026-1_4 · Source: DBLP

**7 authors**, including:

Xianyue Li
Lanzhou University
**25** PUBLICATIONS **452** CITATIONS
SEE PROFILE

Xiaohua Xu
Kennesaw State University
**43** PUBLICATIONS **1,023** CITATIONS
SEE PROFILE

Amy Y-X Wang
Tsinghua University
**102** PUBLICATIONS **666** CITATIONS
SEE PROFILE

# A PTAS for Node-Weighted Steiner Tree in Unit Disk Graphs

Xianyue Li[1], Xiao-Hua Xu[3], Feng Zou[2], Hongwei Du[3], Pengjun Wan[3,⋆], Yuexuan Wang[4], and Weili Wu[2]

[1] School of Mathematics and Statistics, Lanzhou University
Lanzhou, Gansu 730000, China
lixianyue@lzu.edu.cn
[2] Department of Computer Science, University of Texas at Dallas
Richardson, TX 75080, USA
phenix.zou@student.utdallas.edu, weiliwu@utdallas.edu
[3] Department of Computer Science, Illinois Institute of Technology
Chicago, IL 60616, USA
xxu23@iit.edu, hongwei@cs.cityu.edu.hk, wan@cs.iit.edu
[4] Institute of Theoretical Computer Science, Tsinghua University
Beijing 100084, China
wangyuexuan@tsinghua.edu.cn

**Abstract.** The node-weighted Steiner tree problem is a variation of classical Steiner minimum 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. In this paper, we study this problem in unit disk graphs and present a $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$, when the given set of vertices is $c$-local. As an application, we use node-weighted Steiner tree to solve the node-weighted connected dominating set problem in unit disk graphs and obtain a $(5+\varepsilon)$-approximation algorithm.

**Keywords:** Node-weighted Steiner tree, minimum weighted connected dominating set, polynomial-time approximation scheme, approximation algorithm.

## 1 Introduction

Given a graph $G = (V, E)$ and a subset $X \subseteq V$, the classical Steiner tree problem is to find a tree of shortest length in $G$ interconnecting $X$, where the length is the sum of the lengths of all edges in the tree. This problem is known to be NP-hard in graphs, and it is also proved to be NP-hard in most other metrics rather than Euclidean [7]. Lots of effort have been devoted to study the approximation algorithms for this problem [4,10,14,17,20] and some of them have successfully achieved constant ratios [10,17,20]. The best known result among all of them is $\rho = 1 + \frac{\ln 3}{2} \approx 1.55$ by Robins et.al [17] up till now.

⋆ This work was partially supported by NSF under grant CNS-0831831.

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 $C : V \rightarrow R^+$ and a subset $X$ of $V$, which is denoted as the *terminal set*, the node-weighted Steiner tree problem is to find a Steiner tree for the set $X$ such that the total weight of this Steiner tree is minimum. In 1991, Berman [3] proved that NWST problem can not be approximated within a factor of $o(\log k)$. Later, Klein and Ravi [13] presented the first asymptotically optimal solution of approximation ratio $2 \ln k$, by constructing the Steiner tree using "spiders", which is a special kind of tree with at most one node of degree greater than two. Later, this ratio is improved to be $1.35 \ln k$ by Guha and Khuller [8]. They introduce a new concept called "branch-spider" based on "spider".

Recently, researchers are interested in this problem on a special type of graphs called unit disk graphs, 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 in the graph is the center of a unit disk and an edge exists between two vertices $u$ and $v$ if and only if the Euclidean distance between $u$ and $v$ is at most 1. Zou et al. [21] is the first to give a 3.875-approximation algorithm by converting the node-weighted Steiner tree problem to the classical Steiner tree problem.

In this paper, we concern about the same problem from a different perspective. We present a polynomial-time approximation scheme (PTAS) when the given set of vertices is *c-local.* A *polynomial-time approximation scheme* (PTAS) is a family of approximation algorithm with ration $1+\varepsilon$ for any $\varepsilon > 0$ and such a scheme would be the best for a NP-hard problem we can expect.

As an application, we use our algorithm to solve minimum weighted connected dominating set problem in unit disk graph. The Minimum Weighted Connected Dominating Set problem (MWCDS) is a generalization of the minimum connected dominating set problem. Given a graph $G = (V, E)$ with node weight function $C : V \rightarrow R^+$, the *minimum weighted connected dominating set* problem is to find a connected dominating set of $G$ such that its total weight is minimized. Up till now, the best known approximation ratio for MWCDS in general graphs is $O(\log n)$ [8].

In unit disk graphs, researchers usually construct an approximation algorithm for MWCDS with the following two steps. The first step is to find a DS, and the second step is to interconnect DS. Ambühl et al. [1] gave the first constant-factor algorithm for MWCDS with an approximation ratio of 89 with a 72-approximation algorithm for MWDS. Huang et al. [11] improved the approximation ratio from 89 to $10+\varepsilon$ with a $(6+\varepsilon)$-approximation algorithm for the first step. Recently, Dai and Yu [6] gave a $(5+\varepsilon)$-approximation algorithm for MWDS. Therefore, the best known approximation ratio for MWCDS in UDG is $8.875+\varepsilon$. In this paper, we first give a $(4+\varepsilon)$-approximation algorithm for MWDS and then obtain a $(5+\varepsilon)$-approximation algorithm by using Steiner tree to interconnect such DS.

The rest of this paper is organized as follows. In Section 2, we give some useful notations and lemmas. In Section 3, we introduce our main strategy, which is the partition and shifting strategy. Based on the partition, we present our algorithm in Section 4. In Section 5, we first show that our algorithm is a PTAS for NWST in unit disk graphs when the given set of vertices is *c-local*. Then, as an application, we obtain a (5+$\varepsilon$)-approximation algorithm for MWCDS in unit disk graphs.

## 2 Preliminaries and Fundamental Lemmas

Given a node-weighted unit disk graph $G = (V, E)$ with weighted function $C$, and the terminal set $X$. For convenience, we normalize the weight function $C$ such that for any vertex $v$ in $G$, $C(v) \geq 1$. And we denote a vertex $u$ in the Steiner tree $T$ for $X$ as *terminal vertex* if $u \in X$; otherwise, we call it *Steiner vertex*.

We introduce two kinds of distance between any two vertices $u$ and $v$ in the graph, which are called $e$-distance and $w$-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. Firstly, 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 $G'$. Easy to see, 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 longest edge is at most $c$. This definition could be considered as the node-weighted version of the *c-local* definition given by [18]. In the following of paper, we assume that the terminal set $X$ is $c$-local for some constant $c$.

In [16], Robin et.al showed that the Minimum Spanning Tree Number for Euclidean metric is 5 and [19] shows that for any unit disk graph $G$, there exits a spanning tree $T$ of $G$ such that the maximum degree of $G$ is at most 5. Thus, we can get the following lemma easily. (Proof is omitted due to lack of space.)

**Lemma 1.** *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.*

Following are some of the preliminaries for MWCDS problem, which we will talk about as an application after introducing our own algorithm. 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). *Minimum Connected Dominating Set* (MCDS) problem is to find a connected dominating set in $G$ with minimum size, which is a well-known NP-complete problem[7] and has been further shown to be NP-complete even though the given graph is a unit disk graph (UDG)[15]. The MWCDS is a generalization of MCDS, which is obviously NP-hard problem in unit disk graphs.

## 3 Partition and Shifting

One of the key strategies adopted in our algorithm is partition and shifting. Considered as a special way to make restriction and derandomize the probabilistic result to get a deterministic one, researchers has started to use partition and shifting strategy [2,9] in approximation algorithms from 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

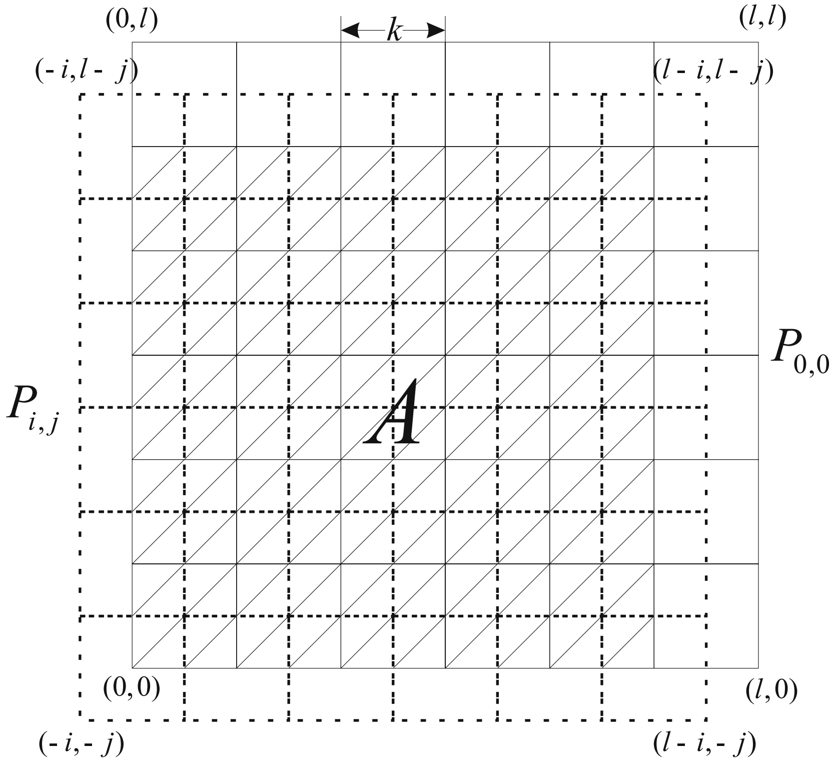

**Fig. 1.** Two partitions $P_{0,0}$ and $P_{i,j}$ with shadow area is $A$. The black is $P_{0,0}$ and the dashed is $P_{i,j}$.

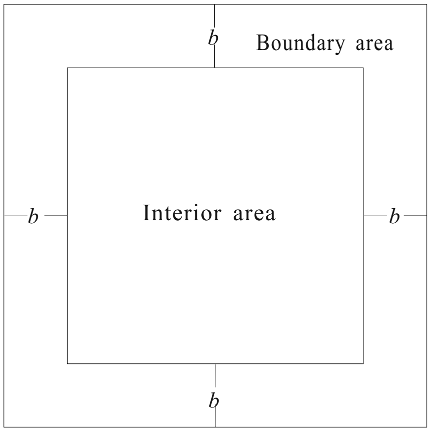

**Fig. 2.** The interior area and boundary area with boundary width $b = (1 + 1.5 \log k)c$

square $A$ as 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: interior area and boundary area with boundary width $b = (1 + 1.5 \log k)c$ (as in Fig 2.). We call this partition as $P_{0,0}$. 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 all $k^2$ possible partitions and any partition contains the area $A$.

Our intention of making use of 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.

## 4 NWST Approximation Algorithm

In this section, we present our approximation algorithm for this problem based on the partition and shifting strategy introduced in last section. 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 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. Firstly, for each cell, we construct a local optimal Steiner forest on terminal vertices in the interior area of this cell. Then, union 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 resulted 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.

Following, we describe in detail the construction of the local optimal Steiner forest and the final Steiner tree in our scheme.

## 4.1 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 terminals 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 derives 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 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 vertices in the resulted Steiner forest to be the merging-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 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 2.** *$\hat{F}_p$ 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, $G$ can be cover 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 $m$ the number of groups in the same cell. Since there is a minimum Steiner tree containing all of edges of induced subgraph

$G[Y]$, we shrink every component of $G[Y]$ to a new vertex and set $Y'$ as the set of these new vertices. Easy to see that $m < |Y'|$. 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 local optimal Steiner forest is $O(2^m M(|Y'|))$ [18], where $M(|Y'|)$ is the time to compute an optimal Steiner tree on terminal set $Y'$ and $M(|Y'|)$ is exponential in $|Y'|$ [5,12]. In order to show $|Y'|$ is bounded by a constant value, 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 5, we will show that $k$ is only related with $c$ and $\varepsilon$. Hence, we can compute a local optimal Steiner forest in polynomial times.

## 4.2 Constructing 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 corresponding path in $G$. Denote $G_p$ as the resulting graph, we have

**Lemma 3.** *$G_p$ 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 $E_p$, 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 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 derives 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.

## 5 Theoretical Analysis

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

### 5.1 $C(\hat{F}_P) \leq C(T_{OPT})$

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 $C(\hat{F}_p) \leq C(F_p)$. 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 3 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 the figure 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

$$1 + 2 + \cdots + 2^{(1.5 \log k)-1} = 2^{1.5 \log k} - 1 = k^{1.5} - 1$$

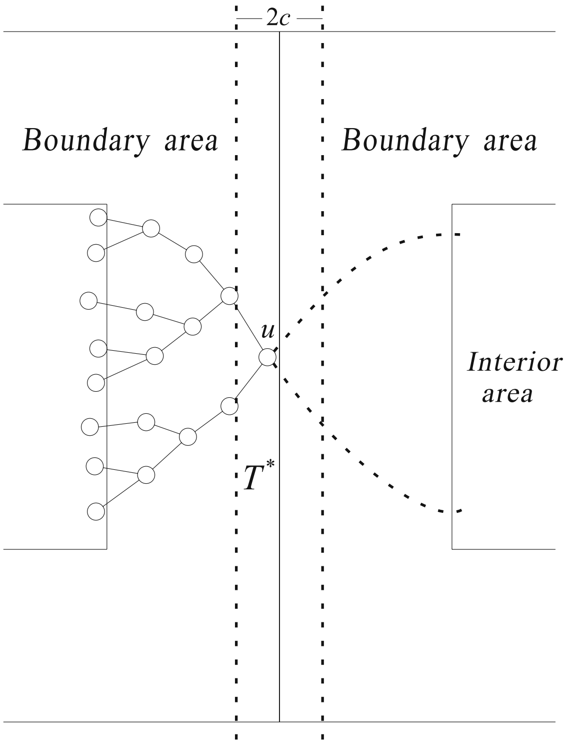

**Fig. 3.** The tree $T^*$ and the vertex $u$

real Steiner vertices in the tree $T^*$(see figure 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 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 be a new tree. As there are at most $k$ trees, the whole weight will increase at most $c(k - 1)$. Meanwhile, as we delete at least $c(k^{1.5} - 1)$ vertices, which means the whole weight decreases at least $c(k^{1.5} - 1)$. Hence, the weight of the new $F'_p$ is decreased by doing so. Repeating this step until there are no trees connecting different cells, denoted 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 Steiner minimum tree on its terminal vertices in the cell. Let $F_p$ be the resulting graph afterwards, we can obtain the following lemmas.

**Lemma 4.** *$F_p$ is a Steriner 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 5.** *$C(\hat{F}_p) \leq C(F_p) \leq C(T_{OPT})$.*

*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 $C(\hat{F}_p) \leq C(F_p)$. □

## 5.2 Performance Analysis

Based on Lemma 5 and the construction of $T_{out}$, we obtain the main theorem in this paper.

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

*Proof.* Recall that $T_{out}$ is consisted of two parts, local optimal $\hat{F}_p$ and $E_{p_{i,j}}$. To bound 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 $1 \times 1$ squares, for different partitions, the same terminal vertex must lie in different square according to the shifting strategy we used. Since there are at most $4ck\lceil 1 + 1.5 \log k\rceil$ squares in 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

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

Therefore, we have

$$C\big(T_{out}\big) \leq \left( \sum_{0 \leq i,j \leq k-1} C(G_{p_{i,j}}) \right) / k^2 \leq \Big(1 + 40c\lceil 1 + 1.5 \log k\rceil / k\Big)\, C\big(T_{OPT}\big).$$

The proof is then finished. □

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

### 5.3 Application on MWCDS

As an application, we apply NWST algorithm into MWCDS problem. Recall that the problem of MWCDS is to construct the connected dominating set in a node-weighted graph with the minimum total weight. Normally, researchers start with calculating dominating set for the graph first and then interconnecting them. Obviously, the node-weighted Steriner tree can be used in the MWCDS problem to interconnect all nodes of the DS to get better approximation algorithm. Therefore, we can obtain the following results.

**Theorem 2.** *There is a (4+$\varepsilon$)-approximation algorithm for minimum weighted dominating set problem. (Proof is omitted due to lack of space.)*

**Corollary 2.** *There is a (5+$\varepsilon$)-approximation algorithm for MWCDS by using node-weighted Steriner tree to interconnect all nodes of the DS.*

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

By Theorem 2, 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

$$\begin{aligned}
C(T) &\le \left(1+\frac{\varepsilon}{7}\right) C(T_{OPT}) \\
&\le \left(1+\frac{\varepsilon}{7}\right) (\, C(D) + C(OPT_{CDS}) \,) \\
&\le \left(1+\frac{\varepsilon}{7}\right)\left(4+\frac{\varepsilon}{7}\right) C(OPT_{CDS}) + \left(1+\frac{\varepsilon}{7}\right) C(OPT_{CDS}) \\
&\le \left(4+6\frac{\varepsilon}{7}\right) C(OPT_{CDS}) + \left(1+\frac{\varepsilon}{7}\right) C(OPT_{CDS}) \\
&\le (5+\varepsilon)\, C(OPT_{CDS}).
\end{aligned}$$

The proof is then finished. □

## 6 Conclusion and Discussion

In this paper, adopting the strategy of partition and shifting, we propose a $(1+\varepsilon)$-approximation algorithm for NWST problem in unit disk graphs, which is the best solution for this problem we could ever have without proving P=NP. As an application, we give a (5+$\varepsilon$)-approximation solution for MWCDS problem in unit disk graphs afterwards, by interconnecting the DS constructed by the $(4 + \varepsilon)$-approximation algorithm using our PTAS solution for NWST problem, which better bounds the performance of the MWCDS compared with existing algorithms.

## References

1. Ambühl, C., Erlebach, T., Mihalák, M., Nunkesser, M.: Constant-factor Approximation for Minimum-weight (Connect) Dominating Sets in Unit Disk Graphs. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX 2006 and RANDOM 2006. LNCS, vol. 4110, pp. 3–14. Springer, Heidelberg (2006)
2. Baker, B.S.: Approximation Algorithm for NP-Complete Problems on Planar Graphs. Journal of the ACM 41, 153–180 (1994)
3. Berman, P.: Personal Communciation (1991)
4. Berman, P., Ramaiyer, V.: Improved Approximations for the Steiner Tree Problem. Journal of Algorithms 17, 381–408 (1994)
5. Chen, D., Du, D.Z., Hu, X.D., Lin, G.H., Wang, L., Xue, G.: Approximation for Steienr Trees with Minimum Number of Steiner Points. Theoretical Computer Science 262, 83–99 (2001)
6. Dai, D., Yu, C.: A 5-Approximation Algorithm for Minimum Weighted Dominating Set in Unit Disk Graph. Theoretical Computer Science (to appear)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-completeness. Freeman, New York (1979)
8. Guha, S., Khuller, S.: Improved Methods of Approximating Node Weighted Steiner Tree and Connected Dominating Sets. Information and Computation 150, 57–74 (1999)
9. Hochbaum, D.S., Maass, W.: Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. Journal of the ACM 32, 130–136 (1985)
10. Hougardy, S., Prömel, H.J.: A 1.598-Approximation Algorithm for the Steiner Problem in Graphs. In: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 448–453 (1999)
11. Huang, Y., Gao, X., Zhang, Z., Wu, W.: A Better Constant-Factor Approximation for Weighted Dominating Set in Unit Disk Graph. Jounral of Combibatorial Optimization (to appear)
12. Hwang, F.K., Richards, D.S.: Steiner tree problems. Networks 22, 55–89 (1992)
13. Klein, P., Ravi, R.: A Nearly Best-Possible Approximation Algorithm for Node-Weighted Steiner Trees. Journal of Algorithms 19, 104–115 (1995)
14. Kou, L.T., Markowsky, G., Berman, L.: A Fast Algorithm for Steiner Trees. Acta Informatica 15(2), 141–145 (1981)
15. Lichtenstein, D.: Planar Formulae and their Uses. SIAM Journal on Computing 11, 329–343 (1982)
16. Robins, G., Salowe, J.S.: On the Maximum Degree of Minimum Spanning Trees. In: Proceedings of the ACM Symposium on Computational Geometry, pp. 250–258 (1994)
17. Robins, G., Zelikovski, A.: Improved Steiner Tree Approximation in Graphs. In: Proc. of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 770–779 (2000)

18. Wang, L., Jiang, T.: An Approximation Scheme for Some Steiner Tree Problem in the Plane. Networks 28, 187–193 (1996)
19. Wu, W., Du, H., Jia, X., Li, Y., Huang, S.C.H.: Minimum Connected Dominating Sets and Maximal Independent Sets in Unit Disk Graphs. Theor. Comput. Sci. 352, 1–7 (2006)
20. Zelikovsky, A.: An 11/6 Approximation Algorithm for the Network Steiner Problem. Algorithmica 9, 463–470 (1993)
21. Zou, F., Li, X., Kim, D., Wu, W.: Two Constant Approximation Algorithms for Node-Weighted Steiner Tree in Unit Disk Graphs. In: Yang, B., Du, D.-Z., Wang, C.A. (eds.) COCOA 2008. LNCS, vol. 5165, pp. 278–285. Springer, Heidelberg (2008)