Discrete Applied Mathematics 286 (2020) 91–99

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

Liar's dominating set problem on unit disk graphs ☆

Ramesh K. Jallu, Gautam K. Das *

Department of Mathematics, Indian Institute of Technology, Guwahati, India

ARTICLE INFO

Article history:
Received 30 June 2017
Received in revised form 30 October 2019
Accepted 17 January 2020
Available online 31 January 2020

Keywords:
Unit disk graph
Approximation algorithm
Dominating set
Liar's dominating set

ABSTRACT

In this paper, we consider Euclidean versions of the 2-tuple dominating set problem and the liar's dominating set problem. For a given set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n points in $\mathbb{R}^2$, the objective of the Euclidean 2-tuple dominating set problem is to find a minimum size set $D \subseteq \mathcal{P}$ such that $|N[p_i] \cap D| \geq 2$ for each $p_i \in \mathcal{P}$, where $N[p_i] = \{p_j \in \mathcal{P} \mid \delta(p_i, p_j) \leq 1\}$ and $\delta(p_i, p_j)$ is the Euclidean distance between p_i and p_j. The objective of the Euclidean liar's dominating set problem is to find a set $D(\subseteq \mathcal{P})$ of minimum size satisfying the following two conditions: (i) D is a 2-tuple dominating set of $\mathcal{P}$, and (ii) for every distinct pair of points p_i and p_j in $\mathcal{P}$, $|(N[p_i] \cup N[p_j]) \cap D| \geq 3$.

We first propose a simple $O(n \log n)$ time $\frac{63}{2}$-factor approximation algorithm for the Euclidean liar's dominating set problem. Next, we propose approximation algorithms to improve the approximation factor to $\frac{732}{\alpha}$ for $3 \leq \alpha \leq 183$, and $\frac{846}{\alpha}$ for $3 \leq \alpha \leq 282$. The running time of both the algorithms is $O(n^{\alpha+1}\Delta)$, where $\Delta = \max\{|N[p]| : p \in \mathcal{P}\}$. Finally, we propose a PTAS for the Euclidean 2-tuple dominating set problem.

1. Introduction

Let $G = (V, E)$ be a simple undirected graph. For a vertex $v \in V$, we define by $N_G(v) = \{u \in V \mid (v, u) \in E\}$ and $N_G[v] = N_G(v) \cup \{v\}$, the open and closed neighborhoods of v in G, respectively. *A subset D of V is a* **dominating set** *if for each vertex $v \in V$, $|N_G[v] \cap D| \geq 1$, that is, every vertex in V is either in D or adjacent to at least one vertex in D.* We say that a vertex v is dominated by u in G, if $(v, u) \in E$ and u is in a dominating set of G (here, we call v as a dominatee and u as a dominator). *A subset D of V is a k-***tuple dominating set** *if each vertex $v \in V$ is dominated by at least k vertices in D, i.e.,* $|N_G[v] \cap D| \geq k$ for each $v \in V$. The minimum cardinality of a k-tuple dominating set of a graph G is known as k-tuple domination number of G. *A subset D of V is a* **liar's dominating set** *if (i) for every $v \in V$, $|N_G[v] \cap D| \geq 2$, and (ii) for every distinct pair of vertices u and v,* $|(N_G[u] \cup N_G[v]) \cap D| \geq 3$. Liar's dominating set problem in an undirected graph $G = (V, E)$ asks to find a liar's dominating set of G with minimum size. The minimum cardinality of a liar's dominating set of a graph G is known as liar's domination number of G. Every 3-tuple dominating set is a liar's dominating set as it satisfies both the conditions, so the liar's domination number lies between 2-tuple and 3-tuple domination numbers.

The k-tuple dominating set problem is a well studied problem in the literature due to its primary application in wireless ad-hoc networks. A connected dominating set (CDS) is a dominating set whose induced subgraph is connected. A CDS is used as a virtual backbone in such networks as there is no centralized physical infrastructure such as routers. The nodes in a CDS act as routers. During the routing process the messages are exchanged via nodes in the CDS. The nodes in CDS are prone to failure due to several reasons, for example, accidental damage, energy depletion, etc. and hence to ensure the robustness of the network it is necessary to have certain degree of redundancy in the CDS.

☆ A preliminary version of this paper appeared in CALDAM, 2017.

* Corresponding author.
E-mail addresses: j.ramesh@iitg.ernet.in (R.K. Jallu), gkd@iitg.ernet.in (G.K. Das).

https://doi.org/10.1016/j.dam.2020.01.017
0166-218X/

The liar's dominating set problem is a variant of the dominating set problem. Dominating set and its variants have been extensively studied for last two decades due to its wide range of applications, e.g., networks, operations research, facility location, etc. [5]. Consider the following real-life scenario: we have an art gallery and the objective is to protect the paintings from an intruder such as a thief, or a saboteur. Assume that the gallery has multiple entrances and each entrance is a possible location for an intruder. A protection device such as a sensor, or a camera placed at an entrance can not only detect (and report precise location) the presence of the intruder entering through it, but also at all the entrances that are visible from it. Now, the goal is to place the protection devices as minimum as possible subject to the intrusion of an intruder at any entrance is detected and reported. We can model the gallery as a graph. A vertex in the graph represents an entrance and an edge corresponds to their respective entrances visibility one from the other. The goal can be achieved by finding a minimum dominating set (MDS) of the graph and placing the devices at all the vertices in the MDS. The protection devices are prone to failure. If any one of the devices placed is failed to detect the presence of the intruder, then we must place protection devices at all the vertices of a minimum 2-tuple dominating set of the graph. Hence, the set of protection devices in the security system is a single fault tolerant set. Due to transmission error it may so happen that all the protection devices detect the location correctly but some of these devices can lie at the time of reporting. Assume that at most one protection device in the closed neighborhood of the intruder can lie (misreport). That is, one of the protection devices can misreport any entrance in its closed neighborhood as the intruder location. In this scenario, in order to protect the paintings in the gallery we must place the protection devices at the vertices of a minimum liar's dominating set of the graph.

An algorithm for a minimization (resp. maximization) problem is said to be a *ρ-factor approximation algorithm* if for every instance of the problem the algorithm produces a feasible solution whose value is within a factor of ρ (resp. $\frac{1}{\rho}$) of the optimal solution value and runs in polynomial-time of the input size. Here, ρ is called as the approximation factor or the performance ratio of the algorithm.

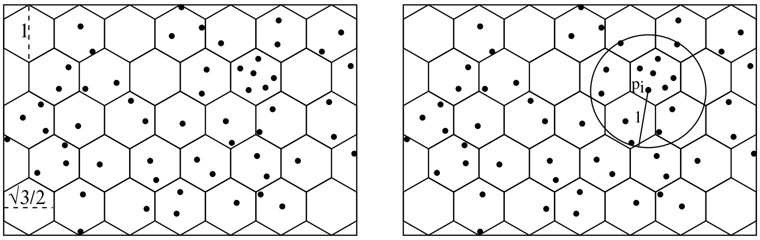
A *polynomial-time approximation scheme* (PTAS) for an optimization problem is a collection of algorithms $\{\mathcal{A}_\epsilon\}$ such that for a given $\epsilon > 0$, $\mathcal{A}_\epsilon$ is a $(1+\epsilon)$-factor approximation algorithm in case of minimization problem ($(1-\epsilon)$ in case of maximization). The running time of $\mathcal{A}_\epsilon$ is required to be polynomial in the size of the problem depending on ϵ.

2. Related work

The concept of k-tuple dominating set was first introduced by Haynes et al. [4]. Klasing and Laforest [8] studied the minimum k-tuple dominating set problem for general graphs and described a $(\ln|V| + 1)$-factor approximation algorithm and proved that the problem cannot be approximated within a ratio of $(1-\epsilon)\ln|V|$ for any $\epsilon > 0$, unless $NP \subseteq DTIME(|V|^{O(\log\log|V|)})$. The authors also showed that the problem can be approximated within a constant ratio if the degree of the graph is bounded by a constant. For p-claw free graphs (i.e., graphs which do not have bipartite graph $K_{1,p}$ as an induced subgraph), the authors proposed a $\frac{(p-1)}{2}(k-1+\frac{2}{\alpha})$-factor approximation algorithm. The problem is known to be NP-complete for split graphs and bipartite graphs due to Liao and Chang [9]. A subset D of V is said to be a k-tuple total dominating set, if every vertex in V is adjacent to at least k vertices of D. The objective of the k-tuple total dominating set problem is to find a minimum size k-tuple total dominating set of V. Pradhan [13] studied the k-tuple total dominating set problem and showed that the problem is NP-hard for split graphs and bipartite graphs. The k-tuple total dominating set problem cannot be approximated within $(1-\epsilon)\ln|V|$ for any $\epsilon > 0$, unless $NP \subseteq DTIME(|V|^{O(\log\log|V|)})$. The result holds even for bipartite and chordal graphs.

A **unit disk graph** (UDG) is an intersection graph of disks of equal radii in the plane. The minimum k-tuple dominating set problem is well studied in UDGs due to its applications in wireless networks. UDGs are 6-claw free and hence there is a 5-factor approximation algorithm for the minimum 2-tuple dominating set problem [8]. Yang et al. [17] considered the minimum connected k-tuple dominating set (i.e., a k-tuple dominating set whose induced subgraph is connected) problem and proposed a cluster-based algorithm with performance ratio $O(k^2)$ in UDGs. Shang et al. [15] presented an algorithm for the minimum k-tuple dominating set problem with performance ratio $6 + \ln\frac{5}{2}(k-1)$.

The liar's dominating set problem was first introduced by Slater in 2009 and proved that the problem is NP-hard for general graphs [16]. Later, Roden and Slater proved that the problem is NP-hard even for bipartite graphs [14]. Panda and Paul [10] proved that the problem is NP-hard for split graphs and chordal graphs. They also proposed a linear time algorithm for computing a minimum cardinality liar's dominating set in a tree. Bishnu et al. [2] proved that the liar's dominating set problem is W[2]-hard for general graphs. Alimadadi et al. [1] provided characterization of graphs and trees for which liar's domination number is $|V|$ and $|V|-1$, respectively. For connected graphs with girth (i.e., the length of a shortest cycle) at least five, they obtained an upper bound for the ratio between the liar's domination number and the 2-tuple domination number. Panda et al. [12] studied approximability of the problem and presented $3 + 2\ln(\Delta(G)+1)$-factor approximation algorithm, where $\Delta(G)$ is the degree of the graph. Panda and Paul [11] considered the problem for proper interval graphs and proposed a linear time algorithm for computing a minimum cardinality liar's dominating set. The problem is also studied for bounded degree graphs, and p-claw free graphs [12].



(a) (b)

Fig. 1. (a) Hexagonal partition of the rectangular region containing the points and, (b) A unit disk centered at a point p_i circumscribes the cell in which p_i lies.

2.1. Our contribution

In this paper, we consider the geometric version of the liar's dominating set problem and we call it as *Euclidean liar's dominating set problem*. In the Euclidean liar's dominating set (ELDS) problem we are given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$. For $p, q \in \mathcal{P}$, $\delta(p, q)$ denotes the Euclidean distance between p and q. For $p \in \mathcal{P}$, $N[p] = \{q \in \mathcal{P} \mid \delta(p, q) \leq 1\}$, in other words, $N[p]$ is the set of points that are within unit distance from p_i. Some times we say $N[p]$ as the closed neighborhood of p in the rest of the paper. We define $\Delta = \max\{|N[p]| : p \in \mathcal{P}\}$. *The objective of the ELDS problem is to find a minimum size subset D of $\mathcal{P}$ such that (i) for every point $p_i \in \mathcal{P}$ there exist at least two points in D which are of at most distance one unit from p_i, and (ii) for every distinct pair of points p_i and p_j in $\mathcal{P}$, $|(N[p_i] \cup N[p_j]) \cap D| \geq 3$, in other words, the number of points in D that are within unit distance with points in the closed neighborhood union of p_i and p_j is at least three*. subset D of $\mathcal{P}$ satisfying condition (i) is called as an Euclidean 2-tuple dominating set (or simply 2-tuple dominating set). Recently, Jallu and Das proved that the ELDS problem is NP-hard [7]. Unlike in general graphs, the ELDS problem admits constant factor approximation algorithms. Observe that the underlying graph of the point set $\mathcal{P}$ is a UDG, $G = (V, E)$, where $V = \mathcal{P}$ and there is an edge between two vertices if the Euclidean distance between the corresponding points is at most 1.

We present a simple $\frac{63}{2}$-factor approximation algorithm in $O(n \log n)$ time followed by a $\frac{732}{\alpha}$-factor approximation algorithm in $O(n^{\alpha+1}\Delta)$ time for $3 \leq \alpha \leq 183$ for the ELDS problem. We then extend the idea of $\frac{732}{\alpha}$-factor approximation algorithm to get a $\frac{846}{\alpha}$-factor approximation algorithm in $O(n^{\alpha+1}\Delta)$ time for $3 \leq \alpha \leq 282$. For the minimum 2-tuple dominating set problem, we propose a PTAS, which runs in $n^{O(k)}$ time for a given integer $k > 1$ and produces a solution whose size is at most $(1 + \frac{1}{k})^2|OPT|$, where OPT is an optimum solution.

2.2. Organization of the paper

The proposed $\frac{63}{2}$-factor approximation algorithm for the ELDS problem is discussed in Section 3. In Section 4, we propose a $\frac{732}{\alpha}$-factor approximation algorithm for $3 \leq \alpha \leq 183$. We further improve the approximation factor to $\frac{846}{\alpha}$ for $3 \leq \alpha \leq 282$. We design a PTAS for the 2-tuple dominating set problem in Section 5. Finally, we conclude the paper in Section 6.

3. $\frac{63}{2}$-factor approximation algorithm for the ELDS problem

Let $\mathcal{R}$ be an axis parallel rectangular region containing the point set $\mathcal{P}$. Since, for any liar's dominating set D, $|(N[p_i] \cup N[p_j]) \cap D| \geq 3$ for all $p_i, p_j \in \mathcal{P}$, we assume that $|\mathcal{P}| \geq 3$, and $|N[p_i]| \geq 2$ for every $p_i \in \mathcal{P}$. The objective of this section is to find a Euclidean liar's dominating set $D(\subseteq \mathcal{P})$ such that $|D| \leq \frac{63}{2}|OPT|$, where OPT is an optimum solution. We partition the region $\mathcal{R}$ into regular hexagonal cells with side length $\frac{1}{2}$ (see Fig. 1(a)). We have chosen side length as $\frac{1}{2}$ to make the maximum distance between any two points lying inside a cell is at most one.

For any two points p, q in $\mathcal{P}$, if $q \in N[p]$, then we say that q is a neighbor of p (some times we say that q is covered by a unit radius disk centered at p or, simply, q is covered by p) and vice versa. The following observations are true as the largest diagonal of each cell is of unit length.

Observation 3.1. *All the points inside a cell can be covered by a unit radius disk centered at any point in that cell (see Fig. 1(b)).*

Observation 3.2 ([3])**.** *A unit disk centered at a point in a cell cannot cover the points lying in more than 12 cells simultaneously.*

The pseudo code to find a liar's dominating set for the points in $\mathcal{P}$ is given in Algorithm 1.

Lemma 3.3. *The set D returned by Algorithm 1 is a liar's dominating set of $\mathcal{P}$.*

Algorithm 1 Liar's_Dominating_Set ($\mathcal{P}$)

Input: The set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n points.
Output: A liar's dominating set D of $\mathcal{P}$

```
1: Let 𝓡 be the minimum size axis parallel rectangle containing points in 𝓟
2: D = ∅
3: Partition the region 𝓡 into regular hexagonal cells of side length 1/2.
4: for (each non-empty cell H_i in the partition) do
5:     n_i ← the number of points in H_i
6:     if (n_i ≥ 3) then
7:         Choose any three arbitrary points p_i, p_j, and p_k from H_i.
8:         D = D ∪ {p_i, p_j, p_k}.
9:     else
10:        if (n_i = 2) then
11:            Let p_i and p_j be the points in H_i.
12:            D = D ∪ {p_i, p_j, p_k}, where p_k ∈ N[p_i] ∪ N[p_j] and is different from p_i and p_j.
13:        else(n_i = 1)
14:            Let p_i be the point in H_i.
15:            D = D ∪ {p_i, p_j, p_k}, where p_j and p_k are two neighbors of p_i, if exists, other wise, p_j is a neighbor of p_i and
    p_k is a neighbor of p_j(i ≠ k).
16:        end if
17:    end if
18: end for
19: Return D.
```

Proof. Let H be a non-empty cell in the hexagonal partition of $\mathcal{R}$ and $p_i \in H \cap \mathcal{P}$. If $|H \cap \mathcal{P}| \geq 2$, then we choose at least two points from $H \cap \mathcal{P}$ as members of D in Algorithm 1 (see line numbers 8 and 12). Therefore, $|N[p_i] \cap D| \geq 2$ by Observation 3.1. Now, if $|H \cap \mathcal{P}| = 1$, then D contains p_i and one more point of $\mathcal{P}$, which is a neighbor of p_i (see line number 15 in Algorithm 1). So, in this case also $|N[p_i] \cap D| \geq 2$. Therefore, $|N[p_i] \cap D| \geq 2$ for any $p_i \in \mathcal{P}$.

Now, we prove that every pair of points p_i and $p_j (p_i \neq p_j)$ in $\mathcal{P}$ have at least three points in D in their closed neighborhood union.

Case 1. p_i and p_j belong to a same cell, say H_i.

(a) H_i contains more than two points of $\mathcal{P}$: D contains three points from H_i (see line number 8 in Algorithm 1). The points p_i and p_j may or may not be part of these three points. In either case $|(N[p_i] \cup N[p_j]) \cap D| \geq 3$ holds by Observation 3.1.

(b) H_i contains only two points of $\mathcal{P}$: in this case both p_i and p_j must be in D and one of its neighbors also must be in D (see line number 12 in Algorithm 1). Hence, $|(N[p_i] \cup N[p_j]) \cap D| \geq 3$.

Case 2. p_i and p_j belong to two distinct cells, say H_i and H_j, respectively.

(a) $|(H_i \cup H_j) \cap \mathcal{P}| \geq 3$, i.e., either $|H_i \cap \mathcal{P}| \geq 2$ and $|H_j \cap \mathcal{P}| \geq 1$, or $|H_i \cap \mathcal{P}| \geq 1$ and $|H_j \cap \mathcal{P}| \geq 2$: D contains at least 3 points from $(H_i \cup H_j) \cap \mathcal{P}$ (see line numbers 8, 12, and 15 in Algorithm 1). Therefore, $|(N[p_i] \cup N[p_j]) \cap D| \geq 3$ (by Observation 3.1).

(b) $|(H_i \cup H_j) \cap \mathcal{P}| = 2$, i.e., $|H_i \cap \mathcal{P}| = 1$ and $|H_j \cap \mathcal{P}| = 1$: in this case, the algorithm chooses p_i and p_j as members of D along with at least one neighbor of p_i (resp. p_j) (see line number 15 in Algorithm 1). Therefore, in this case also $|(N[p_i] \cup N[p_j]) \cap D| \geq 3$.

Thus the lemma follows. □

Theorem 3.4. *The approximation factor and the running time of the proposed algorithm (Algorithm 1) are $\frac{63}{2}$ and $O(n \log n)$, respectively.*

Proof. Let OPT be an optimum solution for the point set $\mathcal{P}$. Consider a non-empty cell H_i in the hexagonal partition. A point in H_i can be dominated by a point in H_i itself or/and also by a point in at most 18 surrounding cells (see Fig. 2), and these 18 cells are the cells that are at most unit distance away from H_i. Let p_i be a point in H_i. Observe that the number of points in OPT dominating (covering) the point p_i is at least 2 (due to first condition of LDS). Hence, the points in OPT dominating the point p_i, say p_j and p_k, should be from H_i or/and its 18 surrounding cells (shown as solid cells in Fig. 2).

The point p_j (resp. p_k) can cover points in at most 12 cells including H_i (by Observation 3.2). Hence, the points covered by p_j (resp. p_k) must lie within the unit radius disk centered at p_j (resp. p_k). However, they may cover some common

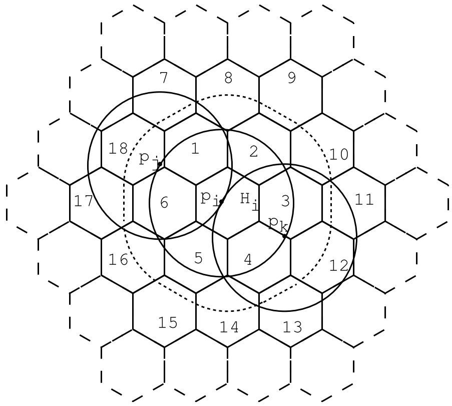
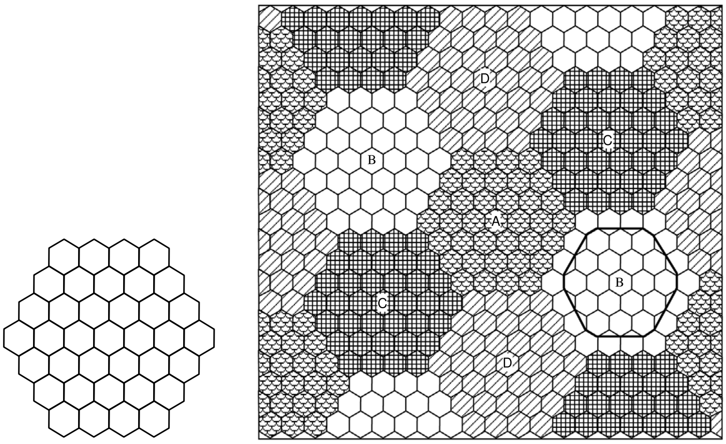


Fig. 2. Cells within unit distance (solid cells) from H_i.



(a) (b)

Fig. 3. (a) A single 37-hexagon, and (b) a four coloring scheme of the hexagonal grid.

cells. The points p_k and p_j cover more cells when they have less number of common cells covered. This scenario happens when p_j and p_k are the end points of a diameter of the unit radius disk centered at p_i. In this case they can have at most three common cells, and cover points in 21 distinct cells. Our algorithm chooses at most 3 points for each cell. Hence, the approximation factor is $\frac{21\times3}{2} = \frac{63}{2}$.

For a point $p_i \in \mathcal{P}$, computing cell number of p_i can be done in constant time as we know the coordinate position of each p_i in $\mathcal{P}$. We can store the non-empty grid cells in a data structure such as balanced binary search tree. For a point $p_i \in \mathcal{P}$ we compute cell number of it and check for the presence of the cell in the data structure. If the cell is not present, we insert the cell in the data structure. Hence, we have in hand that how many and which points of $\mathcal{P}$ are in a cell. After processing the points we just need go through the data structure to report the required points. For one point we spend $O(\log n)$ time, hence for n points it takes $O(n \log n)$ time. Thus the running time for Algorithm 1 follows. □

4. Improving the approximation factor

A 37-hexagon is a combination of 37 adjacent regular hexagonal cells such that one cell is surrounded by 36 other cells (see Fig. 3(a)). Let us consider a partition of the rectangular region $\mathcal{R}$ into 37-hexagons such that no point of $\mathcal{P}$ lies on the boundary of any 37-hexagon in the partition, and a 4 coloring scheme of it (see Fig. 3(b)). Consider a 37-hexagon colored with A. Its adjacent six 37-hexagons are colored B, C, and D (in Fig. 3(b) different patterns have been shown), such that opposite pair of 37-hexagons receives the same color.

Lemma 4.1. *In the 4-coloring scheme, the minimum distance between any two same colored 37-hexagons is greater than or equal to 5.*

Proof. Let H' and H'' be two same colored 37-hexagons in the partition, and a 37-hexagon H lies between H' and H''. Draw a maximum radius circle that can fit entirely in H. This circle must touch the common boundary between (i) H and H', and (ii) H and H''. Draw a line segment between two points where the circle touches the boundaries. Observe that the segment is the diameter of the circle and its length is 5. Thus, the lemma follows. □

Consider a 37-hexagon H, and H can be viewed as a 19-hexagon, say H', surrounded by 18 cells. Let us consider the convex hull overlay, say CH, of the set of corners of H' (shown as loop in Fig. 3(b)). Observe that the maximum distance between any two points in the convex hull overlay CH is at most $\frac{5\sqrt{3}}{2}(> 4)$. Let $S = CH \cap \mathcal{P}$, $S' = \{p \in \mathcal{P} \mid \delta(p, q) \leq 1 \textit{ for } q \in S\}$, and $S'' = \{p \in \mathcal{P} \mid \delta(p, q) \leq 1 \textit{ for } q \in S'\}$. We first propose an approximation algorithm to find a liar's dominating set $S_H \subseteq S''$ for S'. Next, we use the above approximation result to design an approximation algorithm to find out a liar's dominating set for $\mathcal{P}$. Let $OPT_{S'_H}$ denote an optimal liar's dominating set of S'.

Lemma 4.2. $|OPT_{S'_H}| \leq 183$.

Proof. The points in $S' \setminus S$ lie in at most 24 cells around H by definition of S' (i.e., one layer around H). Hence, the points in S' can span over 61 cells. If we choose at most three points for each cell, we get a liar's dominating set. Thus, the cardinality of $OPT_{S'_H}$ cannot be more than 183. □

Lemma 4.3. *If H_1 and H_2 are two same colored 37-hexagons, then $OPT_{S'_{H_1}}$ and $OPT_{S'_{H_2}}$ are independent, i.e., $OPT_{S'_{H_1}} \cap OPT_{S'_{H_2}} = \emptyset$.*

Proof. The proof follows from Lemma 4.1. □

The detailed pseudo code to find a liar's dominating set for the points lying in a given 37-hexagon H is given in Algorithm 2. For a given α ($3 \leq \alpha \leq 183$), we choose all possible $t = 1, 2, \ldots, \alpha - 1$ combinations of points in S'' to find a 2-tuple dominating set of S'. For each combination, we check the combination of points is a 2-tuple dominating set or not (line number 5). While considering the subsets, if there exists a subset S_H that is a 2-tuple dominating set, then for every distinct pair of points p_i and p_j in S', we check whether $|(N[p_i] \cup N[p_j]) \cap S_H| \geq 3$ or not (line number 7). A subset S_H satisfying the above condition is reported and is a liar's dominating set for S'. In the algorithm Boolean variable *flag* is used to ensure that the set returned by the algorithm is a feasible solution.

Lemma 4.4. *For a given 37-hexagon H, Algorithm 2 produces a solution S_H for the set S' from S'' with size is at most $\frac{183}{\alpha}|OPT_H|$, where $3 \leq \alpha \leq 183$ and OPT_H is an optimum solution for the points lying in H.*

Proof. If the algorithm cannot produce a solution of size $\alpha - 1$ for given α ($3 \leq \alpha \leq 183$), then $|OPT_H| \geq \alpha$. Observe that, our algorithm may produce a solution S_H whose size is 183, in the worst. Hence $\frac{|S_H|}{|OPT_H|} \leq \frac{183}{\alpha}$. □

Lemma 4.5. *Algorithm 2 runs in $O(n^{\alpha+1}\Delta)$ time, where $\Delta = \max\{|N[p]| : p \in \mathcal{P}\}$ and $3 \leq \alpha \leq 183$.*

Proof. Algorithm 2 chooses all possible $\alpha - 1$ combinations for a given α. If any of these combinations satisfies LDS conditions, Algorithm 2 reports the combination of points. If it is not possible to find a solution of size $\alpha - 1$, the algorithm chooses at most three points for each non-empty cell (like in Algorithm 1). Steps 4–19 in Algorithm 2 can be done in $O(n^{t-1})(O(\Delta)+O(n^2\Delta)) = O(n^{t+1}\Delta)$. Hence, steps 3–20 take $\sum_{t=1}^{\alpha-1} O(n^{t+1})\Delta$ time. Steps 1 and 22 can be done in $O(n \log n)$ time. Therefore the total running time of Algorithm 2 is $O(n^{\alpha+1}\Delta)$. Thus, the running time result follows. □

We consider each 37-hexagon and compute a feasible solution (using Algorithm 2) for the points lying in it. Two same colored 37-hexagons can be solved independently as the minimum distance between them is greater than 4. Let S_j be the union of solutions generated by the algorithm the 37-hexagons colored j, for $j \in \{A, B, C, D\}$. The set $\mathcal{S} = \bigcup_{j \in \{A,B,C,D\}} S_j$ is reported.

Theorem 4.6. *The set $\mathcal{S}$ is a liar's dominating set of $\mathcal{P}$*

Proof. In Algorithm 2, for a 37-hexagon H, we find a liar's dominating set S_H for S' from S'' (see the definition of S' and S'' defined previously in this section). Now, $S = \bigcup_{H \in \{\text{all 37-hexagons in } \mathcal{R}\}} S_H$. Thus, the theorem follows. □

Theorem 4.7. *The 4-coloring scheme gives a $\frac{732}{\alpha}$-factor approximation algorithm for the ELDS problem and the algorithm runs in $O(n^{\alpha+1}\Delta)$ time, where $3 \leq \alpha \leq 183$.*

Proof. By Lemma 4.1, any two same colored 37-hexagons are more than five units apart. Therefore, we can solve them independently. Let OPT_j be the union of solutions in optimal solution for the 37-hexagons colored j, for $j \in \{A, B, C, D\}$. Also, let OPT be an optimum solution for the point set $\mathcal{P}$, hence, $|OPT_j| \leq |OPT|$ for each $j \in \{A, B, C, D\}$. Observe that,

Algorithm 2 Liar's_Dominating_Set_Septa-hexagon (H, α)

Input: Point set $\mathcal{P}$, a 37-hexagon H and an integer $3 \le \alpha \le 183$.
Output: A liar's dominating set of size $\le \alpha - 1$ for the set S' from S'' (if exists), otherwise a set $S_H (\subseteq S'')$ of size at most 183.

1: Obtain sets S, S', and S''.
2: $flag = 0$.
3: **for** $(t = 1, 2, \ldots, \alpha - 1)$ **do**
4:   **for** (each combination S_H of t points in S'') **do**
5:     **if** $(|S_H \cap N[p_i]| \ge 2 \ \forall p_i \in S')$ **then**
6:       **for** (every distinct pair of points p_i and p_j in S') **do**
7:         **if** $(|(N[p_i] \cup N[p_j]) \cap S_H| \ge 3)$ **then**
8:           $flag = 1$.
9:         **else**
10:           $flag = 0$.
11:           **break**/*break the for loop in line number 6 */
12:         **end if**
13:       **end for**
14:     **end if**
15:     **if** $(flag = 1)$ **then**
16:       Return S_H.
17:       **break**/*break the for loop in line number 3 */
18:     **end if**
19:   **end for**
20: **end for**
21: $S_H \leftarrow \emptyset$
22: **if** $(flag = 0)$ **then**
23:   Consider any three points from each non-empty cell corresponding to S' and add it to S_H (if a non-empty cell contains less than three points we choose the remaining points from its neighbors as in Algorithm 1).
24: **end if**
25: Return S_H.

OPT_j is the optimum for color class j, where we dominate the sets S' with respect to the group of 37-hexagons of color j using the points from S'' and not just S'. The 4-coloring scheme reports the set $\mathcal{S}$, which is the union of the solutions for all 37-hexagons. Therefore, $|\mathcal{S}| \le \frac{183}{\alpha}(|OPT_A| + |OPT_B| + |OPT_C| + |OPT_D|)$. Implies, $|\mathcal{S}| \le \frac{732}{\alpha}|OPT|$ as $|OPT_i| \le |OPT|$ for each $i \in \{A, B, C, D\}$.

The running time result follows from Lemma 4.5 and the fact that a point in $\mathcal{P}$ participates a finite number of times in the algorithm. □

4.1. Further improvement of the approximation factor

The best approximation factor that can be achieved using the algorithm proposed in Section 4 is 4 for $\alpha = 183$. We can extend the idea used for 37-hexagonal partition further to get approximation factor 3. A 66-hexagon is a combination of 66 cells arranged in six rows and each row contains 11 cells (see Fig. 4(a)). We partition the rectangular region $\mathcal{R}$ containing the points in $\mathcal{P}$ into 66-hexagons such that no point of $\mathcal{P}$ lies on the boundary and consider a 3-coloring scheme of it (see Fig. 4(b)).

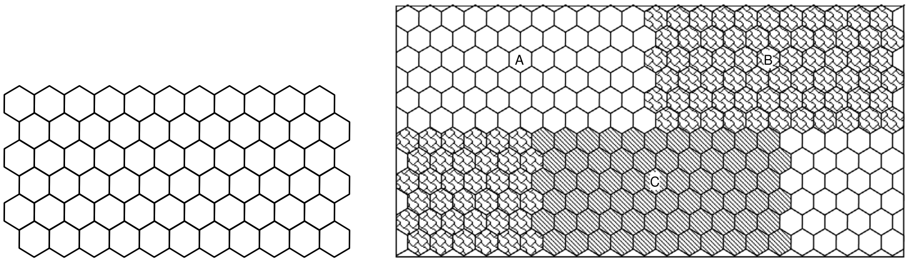
By using the technique in Section 4, we can get $\frac{282}{\alpha}$-factor approximation algorithm for the points lying in a 66-hexagon, where $3 \le \alpha \le 282$, and we have the following theorem.

Theorem 4.8. *The 3-coloring scheme gives a $\frac{846}{\alpha}$-factor approximation algorithm for the ELDS problem in the plane and the algorithm runs in $O(n^{\alpha+1}\Delta)$ time, where $3 \le \alpha \le 282$.*

5. PTAS for the 2-tuple dominating set problem

In this section, we present a $(1 + \frac{1}{k})^2$-factor approximation algorithm (i.e., a PTAS) for the 2-tuple dominating set problem in $n^{O(k)}$ time for a given integer $k > 1$. To the best of our knowledge there is no PTAS available in the literature. The proposed PTAS is based on the shifting strategy proposed by Hochbaum and Maass [6].

We apply the shifting strategy in two levels. The first level contains $k + 1$ iterations. In the ith iteration ($0 \le i \le k$), we partition the region $\mathcal{R}$, containing the given point set $\mathcal{P}$, into vertical strips of width k as follows: the first strip is of width i and the remaining strips are of width k. Note that the width of the last strip may not be k. Let the strips be



(a) (b)

Fig. 4. (a) A single 66-hexagon, and (b) a three coloring scheme of the hexagonal grid.

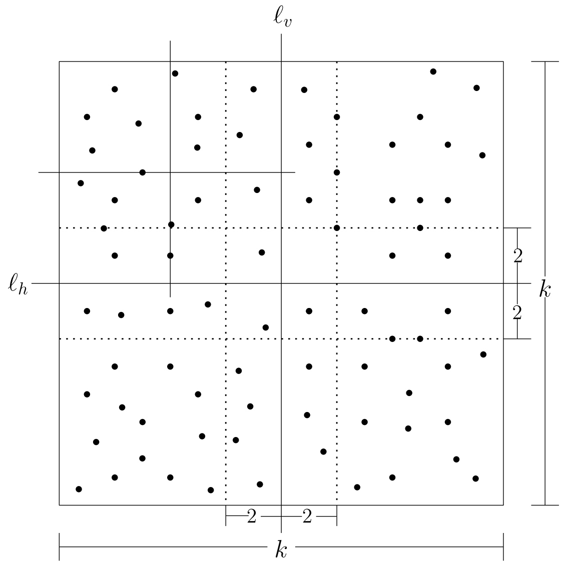


Fig. 5. Demonstration of PTAS: partitioning a square of size $k \times k$.

$H_1, H_2, \ldots, H_\ell$, in order, from left to right. Let p be a point in either H_{i-1} or H_{i+1} for $2 \leq i \leq \ell - 1$. If the removal of the points in H_i leaves the point p isolated, i.e., $|N[p]| = 1$ after removing the points in H_i, then the point p is considered to be part of H_i. Let $P_1, P_2, \ldots, P_\ell$ be the subset of points in $H_1, H_2, \ldots, H_\ell$, respectively.

The basic idea in the shifting strategy is as follows: consider any iteration, say i. Suppose we have an approximation algorithm $\mathcal{A}$ to solve a strip of width k. Apply algorithm $\mathcal{A}$ to each strip in the partition of the current iteration i. By considering the union of all the solutions for each strip, we obtain a feasible solution for the original problem. Repeat the same strategy for every iteration, and report the minimum size solution, say SOL, among all the iterations.

Lemma 5.1 (*The Shifting Lemma [6]*). *$|SOL| \leq \alpha_{\mathcal{A}}(1 + \frac{1}{k})|OPT|$, where k is the shifting parameter, $|OPT|$ is the size of the optimal solution, and $\alpha_{\mathcal{A}}$ is the performance ratio of the algorithm $\mathcal{A}$.*

The second level is as follows: in an iteration of the first level, we consider the strips H_i with $P_i \neq \emptyset$ and apply shifting strategy same as in the first level by considering horizontal partition of each vertical strip H_i. We solve each square of size $k \times k$ optimally (see Section 5.1) and consider the union of the solutions in all the strips to get a solution of that iteration.

5.1. Computing an optimal solution for the points lying in a square of size $k \times k$

Let Q be a subset of points in $\mathcal{P}$ lying in a square χ of size $k \times k$. We find a minimum 2-tuple dominating set of Q using divide and conquer technique. Partition the square χ into four sub-squares each of size $\frac{k}{2} \times \frac{k}{2}$ using the horizontal and vertical lines L_h and L_v (see Fig. 5).

Let $S \subseteq Q$ be the set of points whose distance from L_h and L_v is at most two. The number of points in S that are part of an optimal solution is at most $O(k)$ and which is independent of number of points in the square. This observation can be justified as follows: consider a strip of width 2 and length k with L_h as its bottom (resp. top) boundary. Partition the strip into cells of size $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$. The total number of cells in the partition is $4k$. Note that a point in a cell has distance at most 1 to any other point in that cell. Therefore, there can be at most two points from each non-empty cell in any optimum solution and hence, there can be at most $16k$ points in any optimum solution from the bottom and top strips where L_h is the common boundary. A similar argument works in case of L_v too.

We consider all possible combinations of points from S of size at most $O(k)$ and for every combination we do the following: consider the combination as part of solution and apply recursive procedure to solve each of the sub-problem independently. We consider a best possible solution once the process ends.

Lemma 5.2. *Given a set Q of m points in a square χ of size $k \times k$, a minimum size 2-tuple dominating set of Q can be obtained in $m^{O(k)}$ time.*

Proof. Let OPT_χ be an optimal solution for the points lying in χ. Note that our algorithm checks all combinations of points of size $|OPT_\chi|$. Thus, the combination of points in OPT_χ must appear at some stage. If $T(m, k)$ is the time complexity for finding a minimum 2-tuple dominating set for the points in χ, then $T(m, k) = 4 \times T(m, \frac{k}{2}) \times m^{O(k)} = m^{O(k)}$. □

Theorem 5.3. *Given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$ and integer $k > 1$, the proposed 2-level shifting strategy in Section 5 produces a 2-tuple dominating set for $\mathcal{P}$ with size at most $(1 + \frac{1}{\alpha})^2|OPT|$ in time $n^{O(k)}$, where OPT is an optimal solution.*

Proof. In an iteration of the first level, a solution to each non-empty strip H_i is obtained by applying the shifting strategy horizontally. In this second level, each non-empty strip H_i is partitioned into strips of size $k \times k$ (perhaps, except the first and the last strips). An optimal solution for a strip of size at most $k \times k$ can easily be obtained (refer Section 5.1). By considering union of the solutions for each strip we can obtain a solution for H_i. By Lemma 5.1, the best solution among all the iterations performed on H_i has cardinality at most $(1 + \frac{1}{k})$ times the cardinality of the optimal solution for H_i. So, we have an approximation algorithm for H_i with performance ratio $1 + \frac{1}{k}$. As we are using nested shifting strategy, the theorem follows by substituting $\alpha_{\mathcal{A}}$ by $1 + \frac{1}{k}$ in Lemma 5.1. The running time follows from Lemma 5.2. □

6. Conclusion

In this paper, we first proposed a simple $O(n \log n)$ time $\frac{63}{2}$-factor approximation algorithm for the Euclidean liar's dominating set problem. Next, we proposed two approximation algorithms to improve the approximation factor to $\frac{732}{\alpha}$ for $3 \leq \alpha \leq 183$, and $\frac{846}{\alpha}$ for $3 \leq \alpha \leq 282$. The running time of both the algorithms is $O(n^{\alpha+1}\Delta)$. Finally, we proposed a PTAS for the minimum 2-tuple dominating set problem using nested shifting strategy.

References

[1] A. Alimadadi, M. Chellali, D.A. Mojdeh, Liar's dominating sets in graphs, Discrete Appl. Math. (2016).
[2] A. Bishnu, A. Ghosh, S. Paul, Linear kernels for k-tuple and liar's domination in bounded genus graphs, 2013, arXiv preprint arXiv:1309.5461.
[3] M. De, G.K. Das, P. Carmi, S.C. Nandy, Approximation algorithms for a variant of discrete piercing set problem for unit disks, Int. J. Comput. Geom. Appl. 23 (06) (2013) 461–477.
[4] T.W. Haynes, S. Hedetniemi, P. Slater, Domination in Graphs: Advanced Topics, Marcel Dekker, 1997.
[5] T.W. Haynes, S. Hedetniemi, P. Slater, Fundamentals of Domination in Graphs, CRC Press, 1998.
[6] D.S. Hochbaum, W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI, J. ACM 32 (1) (1985) 130–136.
[7] R.K. Jallu, S.K. Jena, G.K. Das, Liar's dominating set in unit disk graphs, in: International Computing and Combinatorics Conference, Springer, 2018, pp. 516–528.
[8] R. Klasing, C. Laforest, Hardness results and approximation algorithms of k-tuple domination in graphs, Inform. Process. Lett. 89 (2) (2004) 75–83.
[9] C.-S. Liao, G.J. Chang, k-Tuple domination in graphs, Inform. Process. Lett. 87 (1) (2003) 45–50.
[10] B. Panda, S. Paul, Liar's domination in graphs: Complexity and algorithm, Discrete Appl. Math. 161 (7) (2013) 1085–1092.
[11] B. Panda, S. Paul, A linear time algorithm for liar's domination problem in proper interval graphs, Inform. Process. Lett. 113 (19) (2013) 815–822.
[12] B.S. Panda, S. Paul, D. Pradhan, Hardness results, approximation and exact algorithms for liar's domination problem in graphs, Theoret. Comput. Sci. 573 (2015) 26–42.
[13] D. Pradhan, Algorithmic aspects of k-tuple total domination in graphs, Inform. Process. Lett. 112 (21) (2012) 816–822.
[14] M.L. Roden, P.J. Slater, Liar's domination in graphs, Discrete Math. 309 (19) (2009) 5884–5890.
[15] W. Shang, P. Wan, F. Yao, X. Hu, Algorithms for minimum m-connected k-tuple dominating set problem, Theoret. Comput. Sci. 381 (1–3) (2007) 241–247.
[16] P.J. Slater, Liar's domination, Networks 54 (2) (2009) 70–74.
[17] S. Yang, F. Dai, M. Cardei, J. Wu, On multiple point coverage in wireless sensor networks, in: Proceedings of the 2nd International Conference on Mobile Adhoc and Sensor Systems, MASS, IEEE, 2005.