Approximation Algorithms and PTAS of the Minimum Dominating Set and the 3-Path Vertex Cover Problem on Unit Disk Graphs

Haosheng Zhou, Jianhua Tu, Yuqing Kong

May 2020

Abstract

In this paper, we consider the minimum dominating set problem (MDS) and the minimum 3-path vertex cover problem (VCP3) on unit disk graphs (UDG). Based on shifting strategy, together with the application of data structures and geometric properties of unit disk graphs, we manage to propose PTAS and approximation algorithms for those two classical problems in graph theory.

Our contributions mainly lie in the usage of data structures for lowering time complexity and the generalization of shifting strategy for a wider class of problems with its natural extensions in designing PTAS.

Contents

Introduction

Unit Disk Graph (UDG)

In the whole paper, we restrict ourselves to the Euclidean plane $\mathbb{R}^2$. If a graph satisfies the property that for any two vertices $p, q \in \mathbb{R}^2$, there exists an edge between p and q if and only if $||pq|| \leq 1$ ($||pq||$ stands for the Euclidean distance between p and q), the undirected graph is called a unit disk graph. UDG, as a special case of geometric intersection graph, can be understood as the intersection graph of circles centered at all vertices with radius 1. It's obvious that by knowing the coordinates of all vertices, the distances between any pair of vertices are known, thus fixing the structure of a UDG generated by these vertices. V is always used as the vertex set of a UDG in the following context.

Why would we restrict ourselves to discussions on UDG? It has already been theoretically proved that PTAS for MDS and VCP3 do not exist on general graphs. That's the exact reason graphs with special but simple structures like UDG are becoming the focus of such research.

Minimum Dominating Set Problem (MDS)

If $\exists S \subset V$, such that $\forall v \in V - S, \exists s \in S$ such that there's an edge between v and s, then S is called a **dominating set** of a graph. Naturally, the minimum dominating set refers to the dominating set with the least number of vertices in the unweighted case. For the weighted version, an non-negative weight is assigned to each vertex as a weight function $w : V \to \mathbb{R}^+$ and "minimum" then refers to the sum of weights being minimal. In the following context, it's always assumed that UDG is weighted. This includes the unweighted version as a special case where $w \equiv 1$. It's a well-known fact that MDS on general graphs, alike maximum independent set problem, is NP-hard. Despite the toughness, MDS plays a crucial role in graph coloring problems and human network analysis etc., making it necessary to provide efficient approximation algorithms.

3-Path Vertex Cover Problem (VCP3) and Maximum Dissociation Set Problem (MDISS)

VCP3 is a direct generalization of the minimum vertex cover problem. If $\exists S \subset V$ such that for the edge set E and $\forall e \in E$, edge e always has one of its endpoints in S, then S is called a **vertex cover** of the graph. VCP3 considers all paths containing 3 different vertices in the graph in a more general sense. If $\exists S \subset V$ such that for any path $P \subset V$ in the graph containing 3 different vertices, $P \cap S \neq \emptyset$, then S is called a **3-path vertex cover**. Similar to the description in MDS, the minimum vertex cover and minimum 3-path vertex cover can be defined as the minimum with respect to sum of weights.

A standard technique in graph theory is to consider the **dual** of minimum vertex cover problem, turning it into the maximum independent set problem. If $\exists S \subset V$ such that any two vertices in S has no edge connecting them, then S is called an **independent set**. If S is the minimal vertex cover, then $V - S$ is the maximum independent set. The similar technique can be applied to VCP3, turning it into the dual problem called maximum dissociation set problem. The definition of a **dissociation set** S is that $S \subset V$ and $\Delta(G[S]) \leq 1$, i.e. the maximum degree in the induced subgraph of S cannot be larger than 1. It's not hard to find that VCP3 and MDISS are dual problems

of each other.

Here let us introduce another concept closely related to the minimum vertex cover and maximum independent set, called **minimal vertex cover** and **maximal independent set**. A minimal vertex cover S should first be a vertex cover, and satisfies the condition that there exists no vertex cover $S' \subset V$ such that $S' \subset S$, i.e. any subset of a minimal vertex cover can't be a vertex cover except itself. Similarly is maximal independent set defined, i.e. any set containing the maximal independent set except itself can't be an independent set. Intuitively, a minimal vertex cover should be expected to be close to a minimum vertex cover, but it's actually not the case! Simple counterexamples can be built in graph theory showing that a minimal vertex cover can be much greater than a minimum vertex set! As a result, it's especially important to distinguish the two word "minimal" and "minimum".

Unfortunately, it has been proved that both VCP3 and MDISS are NP-hard. However, despite the fact that VCP3 is a relaxation of the minimum vertex cover problem, the optimum of VCP3 enjoys less sum of weights. As a result, VCP3 enables us to lower the costs under the condition that the overall coverage maintains at a high level. VCP3 exhibits its practical value in road monitoring and the encryption of wireless sensor networks. To sum up, an effective polynomial-time approximation algorithm for VCP3 is necessary and valuable.

Prior Work

There exists a lot of fundamental works on the structure of UDG. It's known that judging whether a certain graph is a UDG is NP-hard, while the maximum independent set and maximum cut problem on UDG are also proved to be NP-hard [3, 7]. Although UDG is a special case in the family of geometric intersection graph, its structure still has its complexity in the field of algorithm and is worth looking into.

On the side of the algorithmic research conducted on UDG, a pile of rewarding research emerges in recent years. Leeuwen defines two concepts: "thickness" and "density" of UDG. From the perspective of "thickness", UDG is viewed under its restriction in a strip while for its "density", UDG is viewed after embedded into square grids [19]. These two concepts can work as prototypes of the general shifting strategy [8] on UDG we have made frequent use of in this paper. Da Fonseca et al. gives approximation algorithms for MDS on UDG, one with ratio $\frac{43}{9}$ and time complexity $O(n^2m)$[1], another with ratio $\frac{44}{9}$ and time complexity $O(n \log n)$ and the other with ratio $\frac{44}{9}$ and time complexity $O(n+m)$ [4]. These works are breakthroughs toward the least approximation ratio 5 at that time. Cabello et al. focus on the minimum spanning tree and minimum cut problem on UDG, asserting exact algorithms with time complexity $O(n \log n)$ and $O(n^2 \log^3 n)$ respectively [2]. Bonnet et al. look into the QPTAS and sub-exponential algorithms for the maximum clique problem on UDG, designing an algorithm with time complexity $2^{\tilde{O}(n^{\frac{3}{2}})}$ [1]. Marathe et al. start from the perspective of heuristic algorithms and study general optimizations problems on UDG to provide an approximation algorithm for the minimum vertex cover problem with ratio $\frac{3}{2}$ and for the minimum vertex coloring problem with ratio 3. The works are further generalized onto more general disk graphs [11]. Das et al. make use of data structures to put up an $O(n \log^3 n)$ algorithm with ratio 2.16 and an $O(n^2 \log^2 n)$ algorithm with ratio 2 for the maximum independent set problem on UDG [5]. Nieberg et al. provides an $O(n^{\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}})$ PTAS for MDS on UDG [12] while Wang et al. research the weighted case for MDS and give a PTAS for graphs with bounded

[1] In this paragraph, n denotes the number of vertices and m denotes the number of edges in the graph.

increments and certain degree restrictions, the time complexity is $O(n^{p\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}})$, with p to be the parameter for the bounded increment property [21].

On the other hand, works on VCP3 and more general VCPk problems[2] on special graphs. Xiao et al. consider the parameterized VCP3 problem[3] and give a kernel of $5k$ vertices, together with an $O^*(1.7485^k)$ algorithm [22]. Kardos et al. look into MDISS on general graphs, providing an exact algorithm of time complexity $O^*(1.5171^n)$. In the meantime, they also a randomized polynomial-time approximation algorithm for VCP3 on general graphs with the expectation of the ratio as $\frac{23}{11}$ [9]. Tsur et al. consider the parametric version of this problem, decrease the time complexity to $O^*(1.713^k)$ [14]. Tu et al. conclude that the maximum number of MSS in a tree with n vertices is always $O(3^{\frac{n}{3}})$ [17] and also provide $O(n)$ approximation algorithms for VCP3 on cubic graphs with ratio 1.57 [16], also an $O(n(n+m))$ approximation algorithm for VCP3 in general graphs with ratio 2 [18]. For the PTAS of VCP3, Tu et al. give one that works on planar graphs with time complexity $O(27^{\lceil\frac{1}{\varepsilon}\rceil}\frac{1}{\varepsilon}n)$ [15]. Wang et al. provide an $O(n^{\frac{1}{\varepsilon^2}})$ PTAS for c-local problems on UDG with minimum degree 2 [20]. Liu et al. give an $O(n^{O(k^4(6k)^k\frac{1}{\varepsilon^2})})$ PTAS for connected kPVC problem on UDG [10].

Our Work

Our main work is first to make use of the divide-and-conquer method, giving an approximation algorithm giving minimal dominating sets of UDG. Geometric properties of UDG are the introduced to make slight modifications, providing us with an $O(n^{O(\frac{1}{\varepsilon})})$ PTAS for MDS on UDG.

Secondly, VCP3 problem, after being transformed into equivalent MDISS problem, fits in well with the shifting strategy. We first give an approximation algorithm with performance ratio 2 and time complexity $O(n^4\log^2 n)$ for the weighted case. Afterwards, strip splicing techniques are stated and thus providing us an easy access to the PTAS for VCP3 on UDG.

[2]VCPk is a further generalization of VCP3 that will not be discussed in this paper.
[3]The parameterized VCP3 problem asks if a graph has VCP3 of size at most k.

Data Structure and Shifting Strategy

Data Structure

Here we refer to [5] for the construction of data structures. For the completeness of this paper, lemmas are listed below while details are omitted.

Let P be a set of n vertices in $\mathbb{R}^2$ and R be a query range. An **emptiness inquiry** is defined as a judgement of whether $P \cap R = \emptyset$. The query returns any one of the vertices in $P \cap R$ if the intersection is not empty. Similarly, a **maximum query** is defined, with the only difference that now each vertex has a non-negative real number as its weight and the maximum query returns the vertex in $P \cap R$ with the largest weight. An **anti-disk query** is defined as the query among vertices that are outside the given disk, i.e. setting R as the exterior of the disk. Furthermore, an **n-anti-disk query** is defined by setting R as the union of the exterior of n given disks. Similar definitions hold for **in-disk query** and **n-in-disk query**.

It turns out that a maximum query can be reduced to an emptiness query in the following way:

Lemma 1 ([5]). *For set P with n weighted vertices as elements and a given query range R, if there exists a data structure to do emptiness query taking up space $S(n) = \Omega(n)$, with preprocessing time $B(n) = \Omega(n)$ and query time $Q(n)$, then there exists a data structure to do maximum query on the same R, taking up $O(S(n)\log n)$ space, with preprocessing time $O(B(n)\log n)$ and query time $O(Q(n)\log n)$.*

To construct a specific data structure for anti-disk emptiness query, we depend on the following lemma:

Lemma 2 ([5]). *For set P with n weighted vertices as elements, there exists a data structure to do anti-disk emptiness query, taking up space $O(n)$, with preprocessing time $O(n\log n)$ and query time $O(\log n)$.*

Lemma 3 ([5]). *For set P with n weighted vertices as elements, there exists a data structure to do in-disk emptiness query, taking up space $O(n)$, with preprocessing time $O(n\log n)$ and query time $O(\log n)$.*

Lemma 4 ([5]). *For set P with n weighted vertices as elements, there exists a data structure to do 2-anti-disk emptiness query, taking up space $O(n)$, with preprocessing time $O(n\log n)$ and query time $O(\log n)$.*

Although details are not provided here, the lemmas 2, 3, 4 above all make use of the Voronoi graph and the trapezoidal mapping in their constructions. The one in lemma 2 makes use of the Voronoi graph with order $n-1$, which is just the farthest-point Voronoi graph, while the one in lemma 3 makes use of the Voronoi graph with order 1 and the one in lemma 4 makes use of the Voronoi graph with order $n-2$. That explains why the time and space complexity of these data structures are the same.

Remark. *In the following context, we state lemmas and theorems only for anti-disk query, but actually these also hold for in-disk and 2-anti-disk query.*

Combining lemma 1 and 2 to get:

Lemma 5 ([5]). *For set P with n weighted vertices as elements, there exists a data structure to do anti-disk maximum query, taking up space $O(n\log n)$, with preprocessing time $O(n\log^2 n)$ and query time $O(\log^2 n)$.*

Under certain conditions, the query time in lemma5 can be further reduced with careful considerations:

Lemma 6 ([5]). *For set P with n weighted vertices as elements, with all weights as natural numbers taking values in $\{1, 2, ..., n\}$, there exists a data structure to do anti-disk maximum query, taking up space $O(n)$, with preprocessing time $O(n \log n)$ and query time $O((w_{max} - w_{ret}) \log n)$. Here w_{max} stands for the maximum weight in the data structure and w_{ret} stands for the returned weight value of the query. In particular, if no points are returned, set $w_{ret} = 0$.*

From Overmars and Leeuwen's work [13], the following lemma changes the static data structure into a semi-dynamic one.

Lemma 7 ([13]). *If a data structure storing n vertices solves a separable problem, taking up $S(n)$ space, with preprocessing time $B(n)$ and query time $Q(n)$, then there exists a semi-dynamic data structure solving the same problem that supports insertion of elements taking up space $O(S(n) + \log n)$, with insertion time $O\left(\frac{B(n) \log n}{n}\right)$ and query time $O(Q(n) \log n)$.*

Combining lemma 7 with lemma 5 and 6 to get:

Lemma 8 ([5]). *For set P with n weighted vertices as elements, there exists a semi-dynamic data structure for anti-disk maximum query that. supports insertions of elements, taking up space $O(n \log n)$, with insertion time $O(\log^3 n)$ and query time $O(\log^3 n)$.*

Lemma 9 ([5]). *For set P with n weighted vertices as elements, and the weights are all natural numbers taking values in $\{1, 2, ..., n\}$, there exists a semi-dynamic data structure for anti-disk maximum query that. supports insertions of elements, taking up space $O(n)$, with insertion time $O(\log^2 n)$ and query time $O((w_{max} - w_{ret}) \log^2 n)$.*

Shifting Strategy

The shifting strategy [8] is the most crucial technique we have used in this paper. This technique enables us to start from exact algorithms in small regions and end up with approximation algorithms in large regions. Leeuwen has provided a proof on this [19], but for the sake of the completeness of this paper, we give the proof here once more for a slightly modified version.

Firstly, let us clarify the concept of a **strip**. Given $r, h \in \mathbb{R}$, a strip starting from r with width h refers to the unbounded planar area $\mathbb{R} \times [r, r + h]$. As a result, if not clarified, a strip is by default a horizontal strip, and any square or rectangle region we mention in a later context has its edges parallel to the coordinate axes. The way shifting strategy is actually conducted is described in the proof of the theorem below.

Theorem 1 ([19]). *If there exists a strip with width $h \in \mathbb{N}$ and there exists an exact algorithm solving MDS/VCP3 within the strip with time complexity $T(m)$ for UDG with m vertices, then there exists an approximation algorithm solving MDS/VCP3 for UDG with n vertices of any locations. It has ratio $1+\frac{1}{h}$ and time complexity $O(T(n)+n \log n)$.*

Proof. Here we restrict ourselves to MDS. The case for VCP3 can be similarly proved.

There always exists a large enough rectangle such that all vertices of UDG are in the interior of this rectangle. By applying the shifting strategy, this rectangle will be partitioned into strips and exact algorithms are conducted within each strip, after which the strips are shifted in order to give a lower bound on the approximation ratio.

To be specific, for each $k \in \{1, 2, ..., h\}$, we choose the first strip to be of width k starting from the lower edge of this rectangle, leaving out a blank strip of width 1 on its upper side. Then the second strip with width h is fixed. After leaving out a blank strip of width 1 on the upper side of the second strip, choose the third strip with width h, etc.. By doing this, we get one partition of the rectangle called S_k. It's then obvious that we would have $S_1, ..., S_h$ as h different partitions depending on the width of the first strip. The name "shifting" comes directly from the structure that the change of the width of the first strip works as a shift for all other strips in a certain partition. We refer to the following Fig. 1 for further explanations.

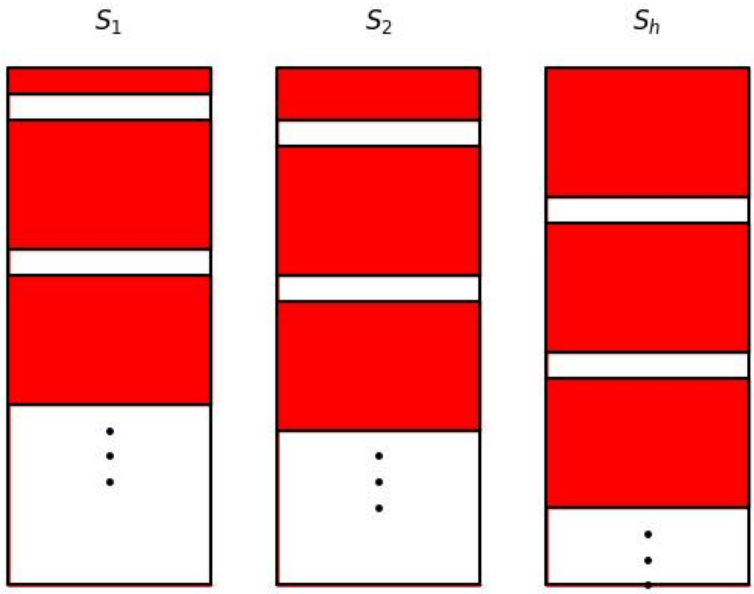

Figure 1: A plot for shifting strategy on UDG

The red strips are the ones within which exact algorithms run (for this plot h is taken as 5) and the blank strips are of width 1. For presentation purpose, the strips are drawn to be bounded, but actually they are horizontal unbounded strips that can extend to infinity by going left or right.
Here the only difference is the width of the first red strip, with partition S_i having the first red strip of width i. As can be seen from the plot, the overall structure is being shifted up for 1 unit as i changes.

It's natural that when the partition S_i is fixed, the exact algorithm A is applied on each single strip within S_i to get the MDS of each strip. The union of all strips' MDS vertices are then taken to get $A(S_i)$ as an approximation of the MDS on the whole UDG. Since we have h different partitions, this step gives us $R_1, ..., R_h$ as h different

approximations for the MDS of the whole UDG. It's natural that we then take the one with the minimum size R as the result of the whole approximation algorithm $S(A)$ that works on the whole UDG. By the process described:

$$|R| = \min\{|R_1|, ..., |R_h|\} \tag{1}$$

$$|R_i| = \sum_{J \in S_i} |OPT_J| \tag{2}$$

where the sum is conducted for all strips J within the partition S_i and OPT_J stands for the locally optimal MDS we get by running the exact algorithm A on strip J. Let OPT denote the actual MDS for the whole UDG and $OPT^{(i)}$ denote the set of vertices in OPT that can dominate some vertices in neighboring strips simultaneously under partition S_i.

By the geometric structure of UDG, since we have a width 1 gap between any two neighboring strips, $OPT^{(i)}$ $(i = 1, 2, ..., h)$ are mutually disjoint.

$$\sum_{J \in S_i} |OPT_J| \leq |OPT| + |OPT^{(i)}| \tag{3}$$

$$\sum_{i=1}^{h} (|OPT| + |OPT^{(i)}|) \leq (1+h)|OPT| \tag{4}$$

$$\tag{5}$$

It's then easy to see that:

$$\min_i \sum_{J \in S_i} |OPT_J| \leq \frac{1}{h} \sum_{i=1}^{h} \sum_{J \in S_i} |OPT_J| \leq (1 + \frac{1}{h})|OPT| \tag{6}$$

□

PTAS and Approximation Algorithms for MDS Problem

An Approximation Algorithm for MDS

We restrict ourselves to the **unweighted** case for the MDS problem. This approximation algorithm gives a minimal dominating set, but not necessarily minimum. As have been stated, the size of a dominating set can have very large difference from the true minimum dominating set, so this approximation algorithm does not have a bounded ratio. Nevertheless, it provides us with inspirations for the PTAS we are going to present in a later context.

The main thought here is the two-layer shifting strategy, as it's named, applying shifting strategy mentioned in Theorem 1 both vertically and horizontally. By doing this, the problem is turned into finding an efficient exact algorithm in the $k \times k$ square area for MDS. To start the construction, we refer to the following lemmas concerning geometric properties of UDG.

Lemma 10 ([6]). *Assume p_a, p_b, p_c, p_d are four vertices within a strip of width 1 and $x(p_a) < x(p_b) < x(p_c) < x(p_d)$. Then if $\{p_a, p_b, p_c\}$ and $\{p_b, p_c, p_d\}$ are two sets such that any two vertices in the set are mutually independent, then $||p_a p_d|| > 1$.*

Lemma 11 ([6]). *Let n_k be the maximum number of mutually disjoint unit disks (their interiors have no intersection) that intersect with a given vertical line l whose centers are inside the strip with width $k >> 1$. Then $n_k \leq \frac{7}{3}k + 2$.*

It's easy to verify that the following lemma holds:

Lemma 12. *Let n'_k be the maximum number of mutually disjoint disks of radius $\frac{1}{2}$ that intersect with a given vertical line l whose centers are inside the strip with width $k >> 1$. Then $n'_k = O(k)$.*

Let's then consider a $k \times k$ square region S, with k to be a fixed constant that is large enough. Now a vertical line l_v pass through S to separate S as two parts with the same area. Similarly, a horizontal line l_h pass through S to separate S as two parts with same area. Denote $S(l_v, l_h)$ as the set of points whose distance to either l_v or l_h is not larger than $\frac{1}{2}$, denote $S(l_v)$ as the set of points whose distance to l_v is not larger than $\frac{1}{2}$, similarly define $S(l_h)$. $S - S(l_v, l_h)$ should be the union of four disjoint square region $S_1, S_2, ..., S_4$. Refer to Fig. 2 for further explanations. Set $V_i = V \cap S_i$ as the set of vertices in region S_i. By applying the divide-and-conquer method, the following approximation algorithm is given.

Note that firstly, the shortest possible distance between two vertices in different regions chosen from $S_1, S_2, ..., S_4$ can't be less than 1, so it's clear that by construction A_i must be the dominating set of V_i. Then, since $S(l_v), S(l_h)$ are both $1 \times k$ rectangles, Lemma 10 tells us how the "if" conditions in Alg. 1 come from. Last but not least, the size of the true MDS in the region $S(l_v, l_h)$ has at most a difference of $2n'_k$ from the size of R derived in Alg. 1. This is because if there are n'_k vertices $v_1, v_2, ..., v_{n'_k}$ as the output of Alg. 1 that are still not enough to dominate all vertices, assume there's a vertex v that has still not been dominated, then consider the disks centered at $v_1, v_2, ..., v_{n'_k}, v$ with radius $\frac{1}{2}$, they must be mutually disjoint but all has intersection with l_v, which is a contradiction to lemma 12!

So far we have proved that the approximation we get from Alg. 1 is a minimal dominating set. However, it's a pity that we can't get a bound on the performance ratio.

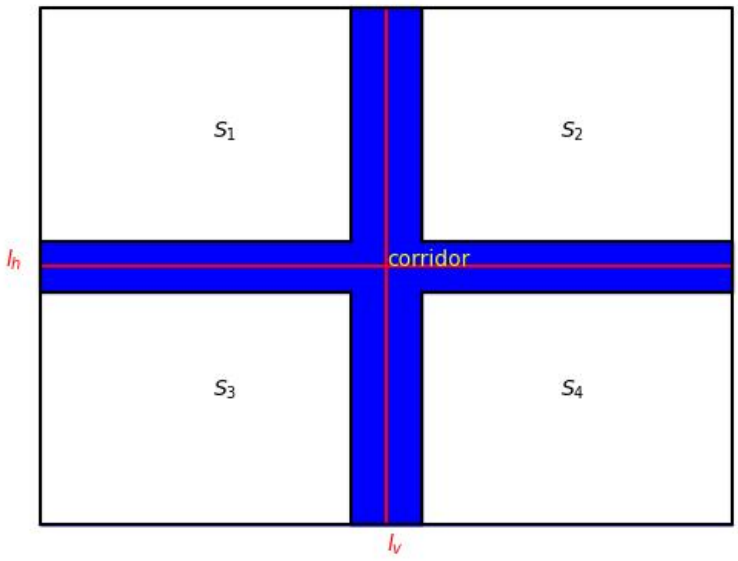

Figure 2: A plot for the construction with divide-and-conquer strategy

The two red lines separate the whole square into two parts with same area. The blue area is the corridor, and all vertices in the corridor form the set $S(l_v, l_h)$. Remaining are 4 smaller square regions, with any two smaller regions being independent of each other by construction. That enables us to proceed on smaller regions, turning large problems into smaller ones.

Theorem 2. *Given n vertices of an unweighted UDG, there exists an approximation algorithm Alg. 1 for MDS in time $O(n \log n)$.*

Proof. The calculation of time complexity requires us to denote $T(n, k)$ as the time taken for this algorithm to run in the case of a UDG with n vertices restricted in a $k \times k$ square region.

Note that the time-consuming step is actually the sorting step and it's assumed that k is a given constant. We get:

$$T(n, k) \leq 4T\left(n, \frac{k}{2}\right) + O(n \log n) \tag{7}$$

$$T(n, k) = O(k^2 n \log n) = O(n \log n) \tag{8}$$

□

Algorithm 1 Approximation Algorithm for MDS Problem

Input: The set P of m vertices $p_1, p_2, ..., p_m$ inside a $k \times k$ square region
Output: An approximation to the MDS on the restriction of UDG in this region as a subset of P

Apply the algorithm recursively to $V_1, V_2, ..., V_4$ to get their respective approximation of MDS denoted A_i
Denote all vertices dominated by A_i as D_i
Eliminate all dominated vertices to get $V' = V - \bigcup_{i=1}^{4} D_i$
$R = \emptyset$
Set $V_h = V' \cap S(l_h)$ and sort the vertices in V_h by their x-coordinates in ascending order
for $v \in V_h$ **do**
  if v is independent of the latest two points before it in R **then**
    $R = R \cup \{v\}$
  end if
end for
Set $V_v = V' \cap S(l_v)$ and sort the vertices in V_v by their y-coordinates in ascending order
for $v \in V_v$ **do**
  if v is independent of the latest two points before it in R **then**
    $R = R \cup \{v\}$
  end if
end for
return $\bigcup_{i=1}^{4} A_i \cup R$

A PTAS for MDS

The inspiration of the PTAS directly comes from Alg. 1. The reason of the failure to have control on the performance ratio in Alg. 1 is that we are applying the approximation algorithm recursively, so error accumulates. Instead, if we do exact searches in each small square region S_i to ensure that we are getting the exact MDS, we should be able to get a bound of the ratio. Luckily, this won't take exponential time since Lemma 12 is always on our side.

To be specific, when we are dealing with the domination of the vertices in the region $S(l_v, l_h)$, we search through all of its subsets with size no larger than $2n'_k$. After determining such a subset, we derive the MDS of the S_i respectively. The matter of change here is just the order of operations. By first determining the MDS of all vertices in the "corridor" and then considering the remaining vertices inside square region S_i, we can ensure that we are deriving the exact MDS. Refer to Alg. 2 for detailed steps.

We immediately conclude that:

Theorem 3. *Given n vertices of an unweighted UDG, there exists an approximation algorithm Alg. 2 for MDS. This gives a PTAS of MDS with performance ratio $1 + \varepsilon$ in time $O\left(n^{O(\frac{1}{\varepsilon})}\right)$.*

Proof. Apply the double shifting strategy with strip width $k \in \mathbb{N}$ as described above to partition the region into $k \times k$ square regions. By Thm. 1, such double shifting strategy gives a ratio $\left(1 + \frac{1}{k}\right)^2$.

Algorithm 2 Exact MDS in Square Region

Input: The set P of m vertices $p_1, p_2, ..., p_m$ inside a $k \times k$ square region
Output: An exact MDS as a subset of P

1: **for** all $A \subset S(l_v, l_h)$ and $|A| \leq 2n'_k$ **do**
2: D denotes all vertices dominated by A
3: $V' = V - D$
4: $V'_i = V' \cap V$ $(i = 1, 2, 3, 4)$
5: Apply the algorithm recursively for each V_i to get their MDSs respectively, denoted D_i
6: Count the number of points in $\bigcup_{i=1}^{4} D_i \cup D$
7: **end for**
8: **return** The solution with the minimum size

Use the same notations as in Thm. 2 to compute time complexity. By Lemma 12:

$$T(n,k) \leq 4T\left(n, \frac{k}{2}\right) n^{2n'_k} \tag{9}$$

$$T(n.k) = n^{O(k)} \tag{10}$$

Setting $1 + \varepsilon = \left(1 + \frac{1}{k}\right)^2$ gives the conclusion.

□

Approximation Algorithms and PTAS for 3PVC

Approximation Algorithm for VCP3

By applying shifting strategy, the problem is transformed to the design of an exact algorithm for VCP3 within a strip. Note that by duality, VCP3 is equivalent to MDISS, with the transformation between costs time $O(n)$. In the following context, we focus on the exact algorithm for MDISS on UDG within a strip.

In the following paragraphs, an approximation algorithm for VCP3 with ratio 2 is provided. Note that although this algorithm costs more time than those Tu et al. provide for general graphs, shifting strategy also holds for MDISS, meaning that by adopting our strategy, an approximation algorithm for MDISS with ratio $1 - \frac{1}{h}$ can be derived simultaneously with the same time complexity. Our strategy is to use strips H with width 1, let P denote the vertex set of the UDG with n vertices and p_i for the i-th vertex in P. The UDG of our consideration now is weighted.

We refer to the following lemmas:

Lemma 13 ([11]). *Define the left neighborhood of a vertex p as $\{q \in P| \ ||pq|| \leq 1, x(q) \leq x(p)\}$. Then within H, a strip with width 1, for any $p \in P$, there are at most 3 mutually independent vertices in p's left neighborhood.*

By the definition of a dissociation set, each vertex in a dissociation set appears in one of two forms: (i): it's independent of any other vertices in the dissociation set (ii): there exists a unique vertex in the dissociation set such that there's and edge between these two vertices. This tells us that for each vertex in a dissociation set, it either appears independently or appears as a pair with another vertex. As a result, two vertices are viewed as a vertex pair and adding vertices as pairs into the dissociation set is considered. Here $EMPTY$ is introduced as the empty vertex, independent of any other vertex to simplify our notations. If a vertex pair includes $EMPTY$, it means that there's actually only one vertex in this pair.

Let $Pair_i$ denote the vertex pair that includes p_i and has p_i as the rightest vertex. Let $Q = \{p_1, ..., p_m\}$ be the set of all vertices within the strip with $x(p_1) \leq x(p_2) \leq ... \leq x(p_m)$ to hold. Given $1 \leq j \leq i \leq m$ such that $||p_i p_j|| > 1$, define the set $S_{i,j} \subset Q$ to be the MDISS with $Pair_i$ being the rightest vertex pair and $Pair_j$ being the second rightest vertex pair. For boundary conditions, it's natural that $S_{i,0} = \{p_i\}$. Set $W_{i,j}$ as the sum of the weights of all vertices in $S_{i,j}$. Then we can update all values of $S_{i,j}$ by dynamic programming. Refer to Alg. 3 for details.

Here let us make some explanations for the variables we use in Alg. 3. D is a 2-dimensional array whose elements are the data structures we mention in Lemma 5 and 6. Each data structure stores tuples like $(j, p_j, b, p_b, W[i, j])$, whose meaning is that assume the MSS has the last vertex pair $Pair_i = \{p_i, p_a\}$, then when the second last vertex pair $Pair_j$ containing p_j takes the value $\{p_b, p_j\}$, the separation set has the largest sum of weights $W[i, j]$. Note that maximal inquiries are always conducted with respect to $W[i, j]$.

The array S is used to take record of the indices of the former vertex of a certain vertex in the MSS for backtracing purpose. For a MSS ending with $Pair_i$ and having $Pair_j$ as the second last vertex pair, $S[i][j][0]$ stands for the vertex index of the vertex other than p_i in the vertex pair $Pair_i$, $S[i][j][1]$ is always set as j, $S[i][j][2]$ stands for the vertex index of the vertex other than p_j in the vertex pair $Pair_j$ and $S[i][j][3]$ stands for the vertex index of the vertex on the right side in the former vertex pair of $Pair_j$.

Algorithm 3 Exact MDISS inside a strip with width 1

Input: The set P of m vertices $p_1, p_2, ..., p_m$ ordered by their x-coordinates in ascending order inside a strip with width 1

Output: An exact MDISS as a subset of P

$EMPTY$ stands for the empty vertex with weight 0, independent of any other vertex
$D = array[1...m][1...m]$ as an array of data structures stated in Lemma 5 for the weighted case and Lemma 6 for the unweighted case. Data structures are to do 2-anti-disk maximum query
$S = array[1...m][0...m][0...3]$ an array of integers
$W = array[1...m][0...m]$ an array of weights
$max = (1, 0)$
for $i = 1, 2, ..., m$ **do**
  $S[i, 0, 0] = S[i, 0, 1] = S[i, 0, 2] = S[i, 0, 3] = 0$
  $W[i, 0] = w(p_i)$
  if $W[i, 0] > W[max]$ **then**
    $max = (i, 0)$
  end if
end for
for $i = 2, 3, ..., m$ **do**
  $T = \emptyset$
  for $j = 1, 2, ..., i - 1$ **do**
    if $||p_i p_j|| > 1$ **then**
      A_i is the disk centered at p_i with radius 1 and A_j is the disk centered at p_j with radius 1
      $submax = (0, 0, 0, 0), result = 0$
      for $p_a \in A_i \cup \{EMPTY\}$ and $j < a < i$ and $||p_a p_j|| > 1$ **do**
        for $p_b \in A_j \cup \{EMPTY\}$ and $b < j$ and $||p_a p_b|| > 1$ and $||p_a p_i|| > 1$ **do**
          B is set as the complement of the union of the disks of radius 1 centered at p_a and p_i
          $k =$2-anti-disk-max-query$(D[j, b], B)$
          if $result < W[j, k] + w(p_a) + w(p_i)$ **then**
            $result = W[j, k] + w(p_a) + w(p_i), submax = (a, j, b, k)$
          end if
        end for
      end for
      for $temp = 0, ..., 3$ **do**
        $S[i][j][temp] = submax[temp]$
      end for
      $W[i, j] = W[j, k] + w(p_{submax[0]}) + w(p_i)$
      $T = T \cup \{(j, p_j, b, p_b, W[i, j])\}$
      if $W[i, j] > W[max]$ **then**
        $max = (i, j)$
      end if
    end if
  end for
  $D[i][a] = preprocess(T)$
end for
$(i, j) = max, MDISS = \emptyset$
while $j \neq 0$ **do**
  $MDISS = MDISS \cup \{p_i, p_{S[i][j][0]}\}$
  $i = j$
  $j = S[i][j][3]$
end while
return $MDISS - \{EMPTY\}$

Variable max means that the current MSS has p_{max} as the ending vertices, $submax$ records the (a, j, b, k) that maximizes $result$, and $result$ stands for the sum of weights of the MSS ending with p_k concatenating $Pair_i, Pair_j$ on its right side.

The outer loops of Alg. 3 aims to update subarray $S[i][j]$ while the inner loop we look through any vertex p_a relevant with p_i and any vertex p_b relevant with p_j and pick out the best p_a, p_b. Note that we have to make sure that any vertex in $Pair_i$ is independent of any vertex in $Pair_j$.

After completing the consideration of all possible outcomes of the MSS that ends with p_i, we preprocess the data structure $D[i][a]$ so that it remains invariant until the end of the algorithm. Starting from row 40 in Alg. 3, we backtrace all vertices in the MSS and eliminate the empty vertex to get the exact optimal result.

Theorem 4. *Given a strip with width 1, and m vertices of UDG, Alg. 3 correctly computes the exact MDISS with time $O(m^4 \log^2 m)$ and space $O(m^3 \log m)$. In particular, if we are dealing with the unweighted case, the time is $O(m^4 \log m)$ and the space is $O(m^3)$.*

Proof. Note that we ensure that vertices from $Pair_i$ and $Pair_j$ are independent and we assume that the vertex pair on the left hand side of $Pair_i$ is $Pair_k$, so vertices in $Pair_i, Pair_j, Pair_k$ are mutually independent. By the lemmas, the algorithm gives the correct result.

Note that by Lemma 8, we do $O(m^4)$ times of inquiries, each with time $O(\log^2 m)$, and we keep $O(m^2)$ data structures, each with space $O(m \log m)$. According to Lemma 9, when it's the unweighted case, we can reduce both time and space complexity by a factor of $\log n$. □

Combine Thm. 4 with Thm. 1, we get the following theorem.

Theorem 5. *Given a set of n weighted vertices of a UDG, there exists an approximation algorithm for VCP3 with ratio 2 and with time $O(n^4 \log^2 n)$, space $O(n^3 \log n)$. For the unweighted case, the algorithm works with time $O(n^4 \log n)$, space $O(n^3)$.*

Strip Splicing Technique

In this section, we introduce how to build a trade-off between the time complexity and the performance ratio of the approximation for VCP3. The main thought is to create wider strip by splicing smaller strips. As a result, we would expect a wider range of geometric properties of UDG to work on wider strips.

We would refer to the following lemma:

Lemma 14 ([5])**.** *p_a, p_b, p_c are three vertices of the UDG inside the strip with width $\frac{\sqrt{3}}{2}$ and $x(p_a) < x(p_b) < x(p_c)$. If $||p_a p_b|| > 1, ||p_b p_c|| > 1$, then $||p_a p_c|| > 1$.*

It can be seen that Lemma 10 and Lemma 14 are both lemmas that state the "independency" of vertices in the restriction of UDG to a certain strip. We claim that it is this kind of independency that ensures the efficiency and good performance of the algorithms. Naturally, we would like to generalize this kind of independency to broader contexts by asking a question that in a strip with what width can we ensure that if $p_a, ..., p_e$ are five vertices sorted

by their x-coordinates in a strip in ascending order, with $p_a, ..., p_d$ and $p_b, ..., p_e$ being two sets such that any two vertices selected from the set are mutually independent, then p_a, p_e are independent.

The following lemma answers this question.

Lemma 15. *If $p_a, ..., p_e$ are five vertices such that $x(p_a) < ... < x(p_e)$ in a strip with width $\sqrt{3}$, now that $p_a, ..., p_d$ are mutually independent and $p_b, ..., p_e$ are mutually independent, then p_a, p_e are independent.*

Proof. Partition the strip H with width $\sqrt{3}$ into two strips each with width $\frac{\sqrt{3}}{2}$, the upper one is denoted as H_1 and the lower one as H_2. They are separated by a line l.

Without loss of generality, assume $p_a \in H_1$ and we discuss the locations of the other four vertices.

(i): If $p_e \in H_1$

If at least one of p_b, p_c, p_d are in H_1, by Lemma 14, proved.

If $p_b, p_c, p_d \in H_2$, then

$$x(p_d) - x(p_b) \geq 1 \tag{11}$$

$$||p_a p_e|| \geq x(p_e) - x(p_a) \tag{12}$$

$$x(p_e) - x(p_a) \geq x(p_d) - x(p_b) \geq 1 \tag{13}$$

(ii): If $p_e \in H_2$

If $p_b, p_c, p_d \in H_2$, draw the symmetric point of p_e with respect to l, denoted p'_e. It's obvious that

$$||p_b p'_e|| \geq ||p_b p_e|| \tag{14}$$

$$||p_c p'_e|| \geq ||p_c p_e|| \tag{15}$$

$$||p_d p'_e|| \geq ||p_d p_e|| \tag{16}$$

So p_b, p_c, p_d, p'_e are mutually independent, $p'_e \in H_1$. It's proved that p_a, p'_e are independent. Since $||p_a p_e|| \geq ||p_a p'_e||$, proved.

If at least one of p_b, p_c, p_d is in H_1, if there are at least two of these points in H_1, we have

$$||p_a p_e|| \geq x(p_e) - x(p_a) \tag{17}$$

$$x(p_e) - x(p_a) \geq x(p_d) - x(p_a) \geq 1 \tag{18}$$

proved.

Otherwise, if exactly one of p_b, p_c, p_d is in H_1, then the other two of p_b, p_c, p_d are in H_2. By Lemma 14, $||p_a p_e|| \geq x(p_e) - x(p_a) \geq x(p_e) - x(p_b) \geq 1$, proved.

□

Such lemma is a specific example of the strip splicing technique, but it's enough to express the mian thought we use. Let's define the **k-independence property** of a strip as the property that for any $k+1$ vertices in this strip sorted by their x-coordinates in ascending order, the mutual independence of k consecutive vertices always imply the

independence of the first and last vertex. As a matter of fact, strip with width $\sqrt{3}$ has 4-independence property as we have just proved.

The general conclusions for k-independence is as follows.

Theorem 6. *For $k+1$ vertices in a strip with width h, if k-independence is to hold, it suffices to set $h = \max\left\{\lfloor\frac{k}{3}\rfloor, \frac{\sqrt{3}}{2}\lfloor\frac{k}{2}\rfloor\right\}$.*

Proof. The proof is a simple application of the pigeonhole principle.

If we put $k+1$ vertices into a strip splicing with h strips with width 1, there exists one strip with at least $\lceil\frac{k+1}{h}\rceil$ vertices. By Lemma 10, we only need to have $\lceil\frac{k+1}{h}\rceil \geq 4$, thus $h \geq \lfloor\frac{k}{3}\rfloor$.

Similarly, by using Lemma 14, and view the strip as the splicing of n strips with width $\frac{\sqrt{3}}{2}$, we only need to have $\lceil\frac{k+1}{n}\rceil \geq 3$, so $n \geq \lfloor\frac{k}{2}\rfloor, h \geq \frac{\sqrt{3}}{2}n$.

□

The advantage of applying such technique is that it optimizes the performance ratio at the cost of increasing time and space complexity, a trade-off.

Note that in shifting strategy, i.e. Thm. 1, we require the width h to be an integer. This gives rise to the phenomenon that in practice, we won't take the width of the strip to be $\sqrt{3}$ although the 4-independence property holds. Instead, we would floor the width to get 1. However, the approximation algorithm constructed by taking the strip width as 1 has the same performance ratio but lower time and space complexity compared to that constructed with strip width $\sqrt{3}$. We call such phenomenon **domination**, i.e. the performance by using strip with width 1 dominates that using strip with width $\sqrt{3}$.

By viewing strip with width 1 and $\frac{\sqrt{3}}{2}$ as "basis", we want to figure out when domination exists by comparing $\lfloor\frac{k}{3}\rfloor$ and $\lfloor\frac{\sqrt{3}}{2}\lfloor\frac{k}{2}\rfloor\rfloor$. The latter expression has one more floor operator since it's not necessarily an integer.

Lemma 16. *For $k \in \mathbb{N}, k \geq 10, \lfloor\frac{k}{3}\rfloor < \lfloor\frac{\sqrt{3}}{2}\lfloor\frac{k}{2}\rfloor\rfloor$.*

Proof. The proof is based on a simple discussion on the remainder of k divided by 6.

(i): If $k \equiv 0\ (mod\ 6)$, $k = 6p(p \geq 2, p \in \mathbb{N})$, then $\lfloor\frac{k}{3}\rfloor = 2p$ with $\lfloor\frac{\sqrt{3}}{2}\lfloor\frac{k}{2}\rfloor\rfloor = \lfloor\frac{3\sqrt{3}}{2}p\rfloor \geq \lfloor\frac{5}{2}p\rfloor$. When p is even, proved. When p is odd, $\lfloor\frac{5}{2}p\rfloor = \frac{5p-1}{2}$, proved.

Similarly, for $k \equiv 1, 2\ (mod\ 6)$ the lemma holds.

(ii): If $k \equiv 3\ (mod\ 6)$, $k = 6p+3(p \geq 2, p \in \mathbb{N})$, then $\lfloor\frac{k}{3}\rfloor = 2p+1$ with $\lfloor\frac{\sqrt{3}}{2}\lfloor\frac{k}{2}\rfloor\rfloor = \lfloor\frac{3\sqrt{3}}{2}p + \frac{\sqrt{3}}{2}\rfloor = \lfloor\frac{5}{2}p + \frac{\sqrt{3}}{2} + (\frac{3\sqrt{3}-5}{2})p\rfloor$. It's easy to know that $\frac{\sqrt{3}}{2} + (\frac{3\sqrt{3}-5}{2})p > 1$, so $\lfloor\frac{\sqrt{3}}{2}\lfloor\frac{k}{2}\rfloor\rfloor \geq \lfloor\frac{5}{2}p\rfloor + 1$. When p is even, proved. When p is odd, $\lfloor\frac{5}{2}p\rfloor = \frac{5p-1}{2}$, proved.

Similarly, for $k \equiv 4, 5\ (mod\ 6)$ the lemma holds.

□

By the simple calculations in Lemma 16 above, we conclude that when $k \geq 10$, no domination exists. When $k < 10$, however, Lemma 16 still holds only for $k = 8$. So we conclude that when $k = 8$ or $k \geq 10, k \in \mathbb{N}$, no domination exists, otherwise we always prefer using strip with width 1.

PTAS for VCP3

For strip H with k-independence property, we transform VCP3 into MDISS problem. When we are using dynamic programming to get the exact MDISS within strips, we need to fix $k-1$ parameters to ensure that the algorithm gives correct result. Since we are considering vertex pairs in such process, each additional parameter increases the time complexity by a multiple of $O(m^2)$ (m is the number of vertices in the strip). Similarly, each additional parameter increases the space complexity by a multiple of $O(m)$.

Now we get two families of approximation algorithms for VCP3 based on the splicing of strips with width $\frac{\sqrt{3}}{2}$ and 1, the proofs of two theorems below are trivial.

Theorem 7. *For n weighted vertices of a UDG, take width $h = \lfloor \frac{k}{3} \rfloor$ as a splicing of strips with width 1, for which k-independence holds. There exists an approximation algorithm for VCP3 with performance ratio $1 + \frac{1}{h}$, time complexity $O(n^{2k-2} \log^2 n)$ and space complexity $O(n^k \log n)$. In particular, for the unweighted case, the algorithm has time complexity $O(n^{2k-2} \log n)$ and space complexity $O(n^k)$.*

Theorem 8. *For n weighted vertices of a UDG, for $k = 8$ or $k \geq 10, k \in \mathbb{N}$, take width $h = \lfloor \frac{\sqrt{3}}{2} \lfloor \frac{k}{2} \rfloor \rfloor$ as a splicing of strips with width $\frac{\sqrt{3}}{2}$, for which k-independence holds. There exists an approximation algorithm for VCP3 with performance ratio $1 + \frac{1}{h}$, time complexity $O(n^{2k-2} \log^3 n)$ and space complexity $O(n^{k-1} \log n)$. In particular, for the unweighted case, the algorithm has time complexity $O(n^{2k-2} \log^2 n)$ and space complexity $O(n^{k-1})$.*

The families of algorithms in Thm. 7 and Thm. 8 exhibits a trade-off between the accuracy and time complexity, which forms the PTAS of the VCP3 when the parameter k changes.

Theorem 9. *For n weighted vertices of a UDG, take width $h = \lfloor \frac{k}{3} \rfloor$ as a splicing of strips with width 1, for which k-independence holds. There exists a PTAS for VCP3 with performance ratio $1 + \varepsilon$ and time complexity $O(n^{O(\frac{1}{\varepsilon})})$.*

Proof. If we use strip with width 1:

Set $\varepsilon = \frac{1}{h} = \frac{1}{\lfloor \frac{k}{3} \rfloor}$, simple calculations show that the time complexity has upper bound $O(n^{\frac{6}{\varepsilon}+4} \log^2 n)$=$O(n^{O(\frac{1}{\varepsilon})})$, when it comes to the unweighted case, the time complexity reduces by a factor of $\log n$, so the conclusion still holds.

If we use strip with width $\frac{\sqrt{3}}{2}$:

Set $\varepsilon = \frac{1}{h} = \frac{1}{\lfloor \frac{\sqrt{3}}{2} \lfloor \frac{k}{2} \rfloor \rfloor}$, simple calculations show that the time complexity has upper bound

$O(n^{2+\frac{8}{\sqrt{3}}+\frac{8}{\sqrt{3}}\frac{1}{\varepsilon}} \log^3 n)$=$O(n^{O(\frac{1}{\varepsilon})})$.

So these two schemes only differ in constants, but both provide PTAS for VCP3 in time $O(n^{O(\frac{1}{\varepsilon})})$.

□

Bibliography

[1] É. Bonnet, P. Giannopoulos, E. J. Kim, P. Rzażewski, and F. Sikora. Qptas and subexponential algorithm for maximum clique on disk graphs. *arXiv preprint arXiv:1712.05010*, 2017.

[2] S. Cabello and L. Milinković. Two optimization problems for unit disks. *Computational Geometry*, 70:1–12, 2018.

[3] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990.

[4] G. D. da Fonseca, C. M. de Figueiredo, V. G. P. de Sá, and R. C. Machado. Efficient sub-5 approximations for minimum dominating sets in unit disk graphs. *Theoretical Computer Science*, 540:70–81, 2014.

[5] G. K. Das, G. D. da Fonseca, and R. K. Jallu. Efficient independent set approximation in unit disk graphs. *Discrete Applied Mathematics*, 280:63–70, 2020.

[6] G. K. Das, M. De, S. Kolay, S. C. Nandy, and S. Sur-Kolay. Approximation algorithms for maximum independent set of a unit disk graph. *Information Processing Letters*, 115(3):439–446, 2015.

[7] J. Díaz and M. Kamiński. Max-cut and max-bisection are np-hard on unit disk graphs. *Theoretical computer science*, 377(1-3):271–276, 2007.

[8] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

[9] F. Kardoš, J. Katrenič, and I. Schiermeyer. On computing the minimum 3-path vertex cover and dissociation number of graphs. *Theoretical Computer Science*, 412(50):7009–7017, 2011.

[10] X. Liu, H. Lu, W. Wang, and W. Wu. Ptas for the minimum k-path connected vertex cover problem in unit disk graphs. *Journal of Global Optimization*, 56(2):449–458, 2013.

[11] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995.

[12] T. Nieberg and J. Hurink. A ptas for the minimum dominating set problem in unit disk graphs. In *International Workshop on Approximation and Online Algorithms*, pages 296–306. Springer, 2005.

[13] M. H. Overmars and J. van Leeuwen. Worst-case optimal insertion and deletion methods for decomposable searching problems. *Information Processing Letters*, 12(4):168–173, 1981.

[14] D. Tsur. Parameterized algorithm for 3-path vertex cover. *Theoretical Computer Science*, 783:1–8, 2019.

[15] J. Tu and Y. Shi. An efficient polynomial time approximation scheme for the vertex cover p 3 problem on planar graphs. *Discussiones Mathematicae: Graph Theory*, 39(1), 2019.

[16] J. Tu and F. Yang. The vertex cover p3 problem in cubic graphs. *Information Processing Letters*, 113(13):481–485, 2013.

[17] J. Tu, Z. Zhang, and Y. Shi. The maximum number of maximum dissociation sets in trees. *Journal of Graph Theory*, 96(4):472–489, 2021.

[18] J. Tu and W. Zhou. A factor 2 approximation algorithm for the vertex cover p3 problem. *Information Processing Letters*, 111(14):683–686, 2011.

[19] E. J. van Leeuwen. Approximation algorithms for unit disk graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 351–361. Springer, 2005.

[20] L. Wang, X. Zhang, Z. Zhang, and H. Broersma. A ptas for the minimum weight connected vertex cover p3 problem on unit disk graphs. *Theoretical computer science*, 571:58–66, 2015.

[21] Z. Wang, W. Wang, J.-M. Kim, B. Thuraisingham, and W. Wu. Ptas for the minimum weighted dominating set in growth bounded graphs. *Journal of Global Optimization*, 54(3):641–648, 2012.

[22] M. Xiao and S. Kou. Kernelization and parameterized algorithms for 3-path vertex cover. In *International Conference on Theory and Applications of Models of Computation*, pages 654–668. Springer, 2017.

Acknowledgements

I would like to first extend my sincere gratitude for professor Jianhua Tu, my advisor for this paper. Professor Tu selected an interesting and important, but not too difficult topic for me, and spent a lot of time guiding me through this research experience. Throughout the communication with professor Tu, I continuously gained new ideas and kept raising questions on the existent theorems and lemmas. It was really a rewarding experience for me to take this very first try into the research fields.

Secondly, I would like to thank the authors of the papers I refer to. Their theoretical breakthroughs and subtle thoughts inspired me a lot. These thoughts are not always difficult to figure out, but appears as a natural extension of the concepts or theorems we have already had in hands. Because of their great work, I am also able to make my very little contribution to the community.

I would also like to thank my friends and family. They encouraged me when I met with difficulties. I very much appreciate their emotional support.

The formal thesis was submitted in May 2020, but the English version translation and small revisions are actually done in August 2022, after I have already graduated from New York University getting my Master's degree in Mathematics and will start to do my PhD studies in Statistics and Applied Probability at UCSB.

Four years' bachelor's life in Peking University was the most precious experience I have ever had in my life. As far as I'm concerned, although PKU still has a long way to go in order to be competitive internationally in the academic field, I did feel the sense of freedom in my four years. That's not only the freedom to select courses in various fields, but more importantly the freedom to speak and act as I hope as long as nobody is hurt, the freedom to find motivations and lifelong interests and the freedom to speak aloud for the common people and to fight for them a better life. I would always remember the day and nights I spent wandering through the Weiming Lake and the Boya Tower, I am really looking forward to returning to a better PKU in the near future!

Written on August 3, 2022, in New York