Introduction

Rectangle Intersection Graphs
Unit Disk Graph
Discrete Piercing Set for Unit Disks

Approximation Algorithms for Geometric Intersection Graphs

Subhas C. Nandy
(nandysc@isical.ac.in)

Advanced Computing and Microelectronics Unit
Indian Statistical Institute
Kolkata 700108, India.
Outline

1. Introduction

2. Rectangle Intersection Graphs
   - Maximum Independent Set
   - Minimum Clique Cover

3. Unit Disk Graph

4. Discrete Piercing Set for Unit Disks
Introduction

Geometric Intersection Graph

Consider a set of objects distributed in $\mathbb{R}^d$.

The geometric intersection graph $G(V, E)$ with these set of objects is an undirected graph where

- Each node in $V$ corresponds to a distinct object in this set.
- Between a pair of nodes $v_i, v_j \in V$ there is an edge $e_{ij} \in E$ if the corresponding objects intersect.

Example: Interval graph, circular arc graph, rectangle intersection graph, unit disk graph etc.
Introduction

Usefulness:

Many practical problems are formulated as an optimization problem of some geometric intersection graph.

- In VLSI Physical Design rectangle intersection graphs play a major role.
- In wireless communication disk graphs play important role.
- In cartography, intersection graphs of rectangles/circles play important role.
Problems of Interest

**Characterization Problem**  Given a graph $G$, test whether it is the intersection of a set of objects of some desired shape.

**Solving some useful optimization problems**

We discuss on the existing results on the characterization problems of different intersection graphs, and available algorithms for solving the optimization problems.

More specifically, we concentrate on the *approximation algorithms* for some important optimization problems on *geometric intersection graphs*. 
Existing results on Characterization Problems

- Testing whether a given graph $G = (V, E)$ is an *Interval Graph* needs $O(|V| + |E|)$ time
  
  M. C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, 1979

- Testing whether a given graph $G$ is a *Rectangle Intersection Graph* is NP-complete.
  
  M. B. Cozzens, Higher and Multidimensional Analogues of Interval Graphs, 1981.

- Testing whether a given graph $G$ is a *Unit Disk Graph* is NP-complete.
  
Existing Algorithmic results on Interval Graphs and Circular Arc Graphs

Results:

- Both these graphs can be recognized in $O(n)$ time.
- Largest clique and Minimum clique cover can be computed in $O(n \log n)$ time.
- Maximum independent set can be computed in $O(n \log n)$ time.
- There exists an output sensitive algorithm (time complexity $O(n \log k)$) for the maximum independent set of interval graph.

1J. Snoeyink, *Maximum independent set of intervals by divide prune and conquer*, CCCG 2007
Definition

Any graph can be represented as the intersection graph of a set of rectangles in $d$ dimensional plane where $d \geq 1$. The *boxicity* of a graph $G$ is the minimum value of $d$.

**Boxicity-2** Graphs are called the **Rectangle Intersection Graphs**

**Assumption:** The rectangle representation of the graph is given.
Important Results on Boxcicity of a Graph

1. A graph has boxcicity one if and only if it is an interval graph.
2. Every outerplanar graph has boxcicity at most two.
3. Every planar graph has boxcicity at most three.
4. If a bipartite graph has boxcicity two, it can be represented as an intersection graph of axis-parallel line segments in the plane.
5. The upper bound of the boxcicity $d$ is $d \leq \tau + 1$ where $\tau$ is the tree-width $\tau$ of the graph.
Largest Clique of Rectangle Intersection Graph

Let $G = (V, E)$ be a rectangle intersection graph, and the corresponding rectangle representation is given.

**Algorithm:** (* Use Plane Sweep *)

- Process the horizontal boundaries of the rectangles in top-to-bottom order.
- An interval tree is used to maintain the sweep line status. Its each element represents a clique so far detected.
Largest Clique of Rectangle Intersection Graph

Let $G = (V, E)$ be a rectangle intersection graph, and the corresponding rectangle representation is given.

**Algorithm:** (* Use Plane Sweep *)

- Process the horizontal boundaries of the rectangles in top-to-bottom order.
- An interval tree is used to maintain the sweep line status. Its each element represents a clique so far detected.
- With each clique, the count of rectangle lying on it is stored.
When a top boundary of a rectangle is processed,

Two clique-intervals get split, and the count of all the cliques inside the range of the rectangle is increased by one.
When a top boundary of a rectangle is processed,

Two clique-intervals get split, and the count of all the cliques inside the range of the rectangle is increased by one.

When a bottom boundary is processed,

Report the largest clique contained in this rectangle.

At most two clique-intervals get merged with its neighboring interval.

The count of all the cliques inside the range of the rectangle is decreased by one.
**Result:** The number of maximal cliques in the graph can be $O(n^2)$ in the worst case.

**Result:** The largest clique can be computed in $O(n \log n)$ time.

**Application:** Given a set of points in 2D and a rectangle of specified size, report the position of the rectangle on the plane such that it can contain the maximum/minimum number of points.
Hardness results on the optimization problems for Rectangle Intersection Graph

**Hardness result:** Both minimum clique cover and maximum independent set problems for a rectangle intersection graph are NP-hard \(^2\).

**Approximation Result:** Both minimum clique cover\(^3\) and maximum independent set \(^4\) problems for the rectangle intersection graph of equal-width rectangles admit PTAS.


Maximum Independent Set of Rectangle Intersection Graph

**Objective:** Choose maximum number of mutually non-overlapping rectangles among a set of Rectangles $R$.

**Algorithm:**

- Draw a vertical line $\ell$ at the median of the $x$-coordinates of the vertical sides of the rectangles.
- Split the set $R$ into $R_1$, $R_2$ and $R_{12}$, where $R_{12} =$ set of rectangles stabbed by $\ell$; $R_1$ and $R_2$ are the set of rectangles to the left and right of $\ell$.
- Compute max-indep-set $I_{12}$ of $R_{12}$.
- Recursively compute max-indep-set $I_1$ and $I_2$.
- If $|I_1| + |I_2| < |I_{12}|$, report $I = I_{12}$ else $I = I_1 \cup I_2$. 
Maximum Independent Set of Rectangle Intersection Graph

**Complexity Analysis**

In a single level of the recursion hierarchy
Identifying $R_{12}$ and Computation of $I_{12}$ needs $O(n \log n)$ time.
Total size of $R_1$ and $R_2 = O(n)$ in the worst case.
Similarly, at each level of recursion, the total number of rectangles in all the partitions is $O(n)$.
Number of levels of recursion is $O(\log n)$.
Thus,
Overall running time is $O(n \log^2 n)$. 
Maximum Independent Set of Rectangle Intersection Graph

Analysis of approximation factor

Approximation factor $= \frac{|I^*|}{\log n}$.

**Proof:** For $n = 2$, result is obviously true.
Let it be true for $n \leq m$.
Let $I^*$, $I_1^*$, $I_2^*$, $I_{12}^*$ are max. independent set of $R$, $R_1$, $R_2$ and $R_{12}$.
and
$I_1$, $I_2$ and $I_{12}$ are computed independent set of $R_1$, $R_2$ and $R_{12}$. 
Maximum Independent Set of Rectangle Intersection Graph

Analysis of approximation factor

Approximation factor \( \frac{|I^*|}{\log n} \).

**Proof:** For \( n = 2 \), result is obviously true.
Let it be true for \( n \leq m \).
Let \( I^*, I_1^*, I_2^*, I_{12}^* \) are max. independent set of \( R, R_1, R_2 \) and \( R_{12} \) and
\( I_1, I_2 \) and \( I_{12} \) are computed independent set of \( R_1, R_2 \) and \( R_{12} \).

\[ I_{12} = I_{12}^* = I^* \cap R_{12}. \]

\[ |I_1| \geq \frac{|I_1^*|}{\log n/2} \geq \frac{|I^* \cap R_1|}{\log n-1}, \]

\[ |I_2| \geq \frac{|I^* \cap R_2|}{\log n-1} \]
Maximum Independent Set of Rectangle Intersection Graph

Analysis of approximation factor

Approximation factor \( = \frac{|I^*|}{\log n} \).

**Proof:** For \( n = 2 \), result is obviously true.
Let it be true for \( n \leq m \).
Let \( I^*, I_1^*, I_2^*, I_{12}^* \) are max. independent set of \( R, R_1, R_2 \) and \( R_{12} \).
and \( I_1, I_2 \) and \( I_{12} \) are computed independent set of \( R_1, R_2 \) and \( R_{12} \).

\[
I_{12} = I_{12}^* = I^* \cap R_{12}.
\]

| | \[ |I_1| \geq \frac{|I^*_1|}{\log n/2} \geq \frac{|I^* \cap R_1|}{\log n-1} \] | \[ |I_2| \geq \frac{|I^* \cap R_2|}{\log n-1} \] |
|---|---|---|

\[
|I| = \max(|I_{12}|, |I_1| + |I_2|) \geq \max(|I^* \cap R_{12}|, \frac{|I^* \cap R_1| + |I^* \cap R_2|}{\log n-1}) \geq \max(|I^* \cap R_{12}|, \frac{|I^*| - (|I^* \cap R_{12}|)}{\log n-1})
\]
Maximum Independent Set of Rectangle Intersection Graph

Analysis of approximation factor

Approximation factor $= \frac{|I^*|}{\log n}$.

Proof: For $n = 2$, result is obviously true.
Let it be true for $n \leq m$.
Let $I^*$, $I_1^*$, $I_2^*$, $I_{12}^*$ are max. independent set of $R$, $R_1$, $R_2$ and $R_{12}$.
and
$I_1$, $I_2$ and $I_{12}$ are computed independent set of $R_1$, $R_2$ and $R_{12}$.

\[ I_{12} = I_{12}^* = I^* \cap R_{12}. \]
\[ |I_1| \geq \frac{|I_1^*|}{\log n/2} \geq \frac{|I^* \cap R_1|}{\log n-1}, \]
\[ |I_2| \geq \frac{|I^* \cap R_2|}{\log n-1} \]
\[ |I| = \max(|I_{12}|, |I_1| + |I_2|) \geq \max(|I^* \cap R_{12}|, \frac{|I^* \cap R_1| + |I^* \cap R_2|}{\log n-1}) \]
\[ \geq \max(|I^* \cap R_{12}|, \frac{|I^*| - (|I^* \cap R_{12}|)}{\log n-1}) \]

If $|I^* \cap R_{12}| \geq \frac{|I^*|}{\log n}$ then
Theorem is proved.

Otherwise, surely
\[ \frac{|I^*| - (|I^* \cap R_{12}|)}{\log n-1} \geq \frac{|I^*|}{\log n}. \]
Maximum Independent Set of Rectangle Intersection Graph

**Special Case:** All rectangles are of same height $\delta$

2-factor Approximation Algorithm:
Maximum Independent Set of Rectangle Intersection Graph

**Special Case:** All rectangles are of same height $\delta$

2-factor Approximation Algorithm:

- Split the region into horizontal strips of width $\delta$ by drawing horizontal lines $\ell_1, \ell_2, \ldots, \ell_k$. 
Maximum Independent Set of Rectangle Intersection Graph

Special Case: All rectangles are of same height $\delta$

2-factor Approximation Algorithm:

- Split the region into horizontal strips of width $\delta$ by drawing horizontal lines $\ell_1, \ell_2, \ldots, \ell_k$.
- Each rectangle will be stabbed by exactly one horizontal line.
Maximum Independent Set of Rectangle Intersection Graph

**Special Case:** All rectangles are of same height $\delta$

2-factor Approximation Algorithm:

- Split the region into horizontal strips of width $\delta$ by drawing horizontal lines $\ell_1, \ell_2, \ldots, \ell_k$.
- Each rectangle will be stabbed by exactly one horizontal line.
- For each stabbing line $\ell_i$, optimally compute the maximum independent set of all the rectangles stabbed by $\ell_i$. 
**Analysis**

1. Compute $\text{IS}_{\text{odd}} = \text{union of the independent sets corresponding to the odd numbered lines.}$
2. Compute $\text{IS}_{\text{even}} = \text{union of the independent sets corresponding to the even numbered lines.}$
3. If $|\text{IS}_{\text{odd}}| < |\text{IS}_{\text{even}}|$ then $\text{IS} = \text{IS}_{\text{even}}$ else $\text{IS} = \text{IS}_{\text{odd}}$.

**Time complexity:** $O(n \log n)$

**Approximation Factor:** $\max(|\text{IS}_{\text{odd}}|, |\text{IS}_{\text{even}}|) \geq |I^*|/2$

**Application:** Map Labelling
Analysis

- Compute $IS_{odd} = \text{union of the independent sets corresponding to the odd numbered lines}$.  
- Compute $IS_{even} = \text{union of the independent sets corresponding to the even numbered lines}$. 

Time complexity: $O(n \log n)$ 

Approximation Factor: $\max(|IS_{odd}|, |IS_{even}|) \geq |I^*|/2$ 

Application: Map Labelling
Analysis

- Compute $IS_{odd} = \text{union of the independent sets corresponding to the odd numbered lines.}$
- Compute $IS_{even} = \text{union of the independent sets corresponding to the even numbered lines.}$
- If $|IS_{odd}| < |IS_{even}|$ then $IS = IS_{even}$ else $IS = IS_{odd}$. 
Analysis

- Compute $IS_{odd} = \text{union of the independent sets corresponding to the odd numbered lines.}$
- Compute $IS_{even} = \text{union of the independent sets corresponding to the even numbered lines.}$
- If $|IS_{odd}| < |IS_{even}|$ then $IS = IS_{even}$ else $IS = IS_{odd}$. 

Time complexity: $O(n \log n)$
Analysis

- Compute $IS_{odd} =$ union of the independent sets corresponding to the odd numbered lines.
- Compute $IS_{even} =$ union of the independent sets corresponding to the even numbered lines.
- If $|IS_{odd}| < |IS_{even}|$ then $IS = IS_{even}$ else $IS = IS_{odd}$.

**Time complexity:** $O(n \log n)$

**Approximation Factor:** $\max(|IS_{odd}|, |IS_{even}|) \geq |I^*|/2$

**Application:** Map Labelling
Polynomial Time Approximation Scheme (PTAS)

Objective:
Given an integer $k$, find an algorithm for finding an independent set whose size is at least $(1 + \frac{1}{k}) \times |I^*|$. 

Algorithm:

[Diagram showing vertical lines and rectangles intersecting at various points]
Objective:

Given an integer $k$, find an algorithm for finding an independent set whose size is at least $(1 + \frac{1}{k}) \times |I^*|.$

Algorithm:

- Partition the plane by horizontal lines at unit interval.
Polynomial Time Approximation Scheme (PTAS)

Objective:

Given an integer $k$, find an algorithm for finding an independent set whose size is at least $(1 + \frac{1}{k}) \times |I^*|$. 

Algorithm:

- Partition the plane by horizontal lines at unit interval.
- Let $R^k_i$ be the set of rectangles intersecting intersected by any one of the lines $\{\ell_i, \ell_{i+1}, \ldots, \ell_{i+k-1}\}$.
Objective:

Given an integer $k$, find an algorithm for finding an independent set whose size is at least $(1 + \frac{1}{k}) \times |I^*|$. 

Algorithm:

- Partition the plane by horizontal lines at unit interval.
- Let $R_i^k$ be the set of rectangles intersected by any one of the lines $\{\ell_i, \ell_{i+1}, \ldots, \ell_{i+k-1}\}$.
- Define $G_j = R_{j-1}^1 \cap \bigcap_{i \geq 0} R_{i(k+1)+j}^k$. 
**Objective:**

Given an integer $k$, find an algorithm for finding an independent set whose size is at least \((1 + \frac{1}{k}) \times |I^*|\).

**Algorithm:**

- Partition the plane by horizontal lines at unit interval.
- Let $R^k_i$ be the set of rectangles intersected by any one of the lines $\{\ell_i, \ell_{i+1}, \ldots, \ell_{i+k-1}\}$.
- Define $G_j = R^1_{j-1} \bigcap_{i \geq 0} R^k_{i(k+1)+j}$.
- Optimally find the largest set of disjoint rectangles among those in $G_j$. 
Objective:

Given an integer $k$, find an algorithm for finding an independent set whose size is at least $(1 + \frac{1}{k}) \times |I^*|$. 

Algorithm:

1. Partition the plane by horizontal lines at unit interval.
2. Let $R^k_i$ be the set of rectangles intersecting intersected by any one of the lines $\{l_i, l_{i+1}, \ldots, l_{i+k-1}\}$.
3. Define $G_j = R^1_{j-1} \cap \bigcap_{i \geq 0} R^k_{i(k+1)+j}$.
4. Optimally find the largest set of disjoint rectangles among those in $G_j$.
5. Finally, choose the largest among these $k$ independent sets.
Key Observation: No rectangle of $R_1^k$ intersects a rectangle in $R_{k}^{k+2}$. The line $\ell_{k+1}$ separates these subgroups.

We use the dynamic programming based routine\(^5\) for computing the largest independent set of rectangles intersected by $k$ consecutive horizontal lines.

Analysis

**Time Complexity:** \( O(n \log n + n^{2k-1}) \).

**Approximation Ratio:**
Analysis

Time Complexity: $O(n \log n + n^{2k-1})$.

Approximation Ratio:

- A group is formed by deleting all the rectangles that intersect every $(k+1)$-th line.
Analysis

**Time Complexity:** $O(n \log n + n^{2k-1})$.

**Approximation Ratio:**

- A group is formed by deleting all the rectangles that intersect every $(k + 1)$-th line.
- The union of all rectangles in $R \setminus G_j$ is intersected by at most $\left\lceil \frac{m}{k+1} \right\rceil$ lines.
Analysis

**Time Complexity:** $O(n \log n + n^{2k-1})$.

**Approximation Ratio:**

- A group is formed by deleting all the rectangles that intersect every $(k + 1)$-th line.
- The union of all rectangles in $R \setminus G_j$ is intersected by at most $\left\lceil \frac{m}{k+1} \right\rceil$ lines.
- We compute maximum independent set for each $G_j$ and choose the largest one.
Analysis

**Time Complexity:** $O(n \log n + n^{2k-1})$.

**Approximation Ratio:**

- A group is formed by deleting all the rectangles that intersect every $(k + 1)$-th. line.
- The union of all rectangles in $R \setminus G_j$ is intersected by at most $\lceil \frac{m}{k+1} \rceil$ lines.
- We compute maximum independent set for each $G_j$ and choose the largest one.
- By pegion hole principle, it can miss at most $\frac{OPT}{k+1}$ rectangles.
Analysis

**Time Complexity:** $O(n \log n + n^{2k-1})$.

**Approximation Ratio:**

- A group is formed by deleting all the rectangles that intersect every $(k + 1)$-th line.
- The union of all rectangles in $R \setminus G_j$ is intersected by at most $\left\lceil \frac{m}{k+1} \right\rceil$ lines.
- We compute maximum independent set for each $G_j$ and choose the largest one.
- By pigeon hole principle, it can miss at most $\frac{OPT}{k+1}$ rectangles.
- Thus we have an $(1 + \frac{1}{k})$ factor approximation.
Objective:
Choose minimum number of points on the plane such that each rectangle contains at least one of those points

These points will be referred as piercing points

Application:
Given a set of points in 2D and a rectangle of specified size, report the minimum number of rectangles required to cover all the points.
A 2-approximation algorithm

We consider only the special case where rectangles are of same height, say $\delta$.

**Algorithm:**
A 2-approximation algorithm

We consider only the special case where rectangles are of same height, say $\delta$.

**Algorithm:**

- Split the region into horizontal strips of width $\delta$ by drawing horizontal lines $l_1, l_2, \ldots, l_k$. 
A 2-approximation algorithm

We consider only the special case where rectangles are of same height, say $\delta$.

**Algorithm:**

- Split the region into horizontal strips of width $\delta$ by drawing horizontal lines $\ell_1, \ell_2, \ldots, \ell_k$.
- Each rectangle will be stabbed by exactly one horizontal line.
A 2-approximation algorithm

We consider only the special case where rectangles are of same height, say $\delta$.

**Algorithm:**

- Split the region into horizontal strips of width $\delta$ by drawing horizontal lines $\ell_1, \ell_2, \ldots, \ell_k$.
- Each rectangle will be stabbed by exactly one horizontal line.
- For each stabbing line $\ell_i$, optimally compute the minimum clique cover of all the rectangles $L_i$ stabbed by $\ell_i$. 
A 2-approximation algorithm (contd.)

Theorem

The clique cover with all these piercing points is of size $2 \times |OPT|$, where $|OPT|$ is the size of the optimal clique cover.

Proof:
A 2-approximation algorithm (contd.)

Theorem

*The clique cover with all these piercing points is of size $2 \times |OPT|$, where $|OPT|$ is the size of the optimal clique cover.*

Proof:

- Let $L_{odd} = \{\ell_1, \ell_3, \ldots\}$ (* odd numbered stabbing lines *)
- $OPT_{odd} = \text{Piercing points chosen for the rectangles stabbed by the lines in } L_{odd}$. 
A 2-approximation algorithm (contd.)

**Theorem**

The clique cover with all these piercing points is of size $2 \times |OPT|$, where $|OPT|$ is the size of the optimal clique cover.

**Proof:**

- Let $L_{odd} = \{\ell_1, \ell_3, \ldots\}$ (* odd numbered stabbing lines *)
  
  $OPT_{odd} = $ Piercing points chosen for the rectangles stabbed by the lines in $L_{odd}$.

- The piercing points chosen for the rectangles stabbed by $\ell_i \in L_{odd}$ do not lie in any of the rectangles stabbed by the lines $L_{odd} \setminus L_i$. 
A 2-approximation algorithm (contd.)

**Theorem**

The clique cover with all these piercing points is of size $2 \times |OPT|$, where $|OPT|$ is the size of the optimal clique cover.

**Proof:**

- Let $L_{odd} = \{\ell_1, \ell_3, \ldots\}$ (* odd numbered stabbing lines *)
  
  $OPT_{odd} = $ Piercing points chosen for the rectangles stabbed by the lines in $L_{odd}$.

- The piercing points chosen for the rectangles stabbed by $\ell_i \in L_{odd}$ do not lie in any of the rectangles stabbed by the lines $L_{odd} \setminus L_i$.

- Thus, $OPT_{odd}$ can be computed in polynomial time.
Theorem

The clique cover with all these piercing points is of size $2 \times |OPT|$, where $|OPT|$ is the size of the optimal clique cover.

Proof:

- Let $L_{odd} = \{\ell_1, \ell_3, \ldots\}$ (* odd numbered stabbing lines *)
  
  $OPT_{odd} = $ piercing points chosen for the rectangles stabbed by the lines in $L_{odd}$.

- The piercing points chosen for the rectangles stabbed by $\ell_i \in L_{odd}$ do not lie in any of the rectangles stabbed by the lines $L_{odd} \setminus L_i$.

- Thus, $OPT_{odd}$ can be computed in polynomial time.

- $|OPT_{odd}| \leq |OPT|$, and similarly $|OPT_{even}| \leq |OPT|$.

Hence, $|OPT_{odd}| + |OPT_{even}| \leq 2 \times |OPT|$. 
PTAS for Minimum Clique Cover

A PTAS is available for the minimum clique cover problem \(^6\).

**Result**

For a set of \(n\) equal width rectangles and a constant \(\epsilon\), a \((1 + \epsilon)\)-factor approximation of the minimum clique cover can be obtained in \(O(n^{1/\epsilon^2})\) time.

---

Unit Disk Graph

**Definition**

A graph that can be represented as the intersection graph of a set of circles of same radius is called the *unit disk graph*.
Cliques in Unit Disk Graph

Let $v_i, v_j, v_k \in V$ form a clique in the unit disk graph $G = (V, E)$ and $C_i, C_j, C_k$ are the corresponding circles. Now, $C_i, C_j, C_k$ may or may not have a common region.

In the former case it is called a geometric clique, and in the latter case it is called a graphical clique.

![Geometric Clique](image1)

![Graphical Clique](image2)
Largest Clique in Unit Disk Graph

Let $A$ and $B$ be a pair of points such that $\text{dist}(A, B) \leq 1$.

$R_{AB} \Rightarrow$ Intersection of two closed disks of radius $\text{dist}(A, B)$, and centered at $A$ and $B$ respectively.

$H_{AB} \Rightarrow R_{AB} \cap V$

**Observations:**

- If $A$ and $B$ are the maximally distant points in a set $V'$, then $V' \subseteq H_{AB}$.
- If $C$ is the vertex set of a maximum sized clique in $G$ then $C \subseteq H_{AB}$ for some $A, B \in V$ with $\text{dist}(A, B) \leq 1$.

Thus, we inspect all possible pair of points $(A, B)$ with $\text{dist}(A, B) \leq 1$ to identify a pair $(a, b)$ such that $H_{ab}$ contains the maximum clique.
Largest Clique in Unit Disk Graph

Partition $R_{AB}$ into $R_{AB}^0$ and $R_{AB}^1$ by a straight line joining $A$ and $B$.

Define $H_{AB}^0 = R_{AB}^0 \cap V$, and $H_{AB}^1 = R_{AB}^1 \cap V$.

**Observation:** If two points $\alpha, \beta \in H_{AB}^0$ (resp. $H_{AB}^1$), then $dist(X, Y) \leq 1$. Thus, $(\alpha, \beta)$ is an edge in $G$.

**Result:** The subgraph of $G$ induced by the points in $H_{AB}$ is the complement of a bipartite graph.

Computing maximum clique in $G$ is equivalent to computing max. indep. set in $\overline{G}$.

**Result:** The maximum independent set of a bipartite graph with $n$ vertices can be found in $O(n^{2.5})$ time.

**Overall Time Complexity:** $O(n^{4.5})$. 
Largest Geometric Clique in Unit Disk Graph

The largest geometric clique in an unit disk graph represents an area on the plane where maximum number of unit disks overlap.

**Applications:** Given a set of points and a circle, position the circle on the plane to cover maximum number of points.

**Observations:**

- Each clique region is adjacent to the point of intersection of a pair of circles.
- The number of intersection points is $O(n^2)$ in the worst case.
- A trivial $O(n^3)$ time algorithm is to consider each intersection point, and inspect how many points are inside that circle.
- The best known algorithm runs in $O(n^2)$ time\(^7\).

---

\(^7\)S. Cabello, J. Miguel Diáez-Báñez, S. Langerman, C. Seara and I. Ventura, *Facility location problems in the plane based on reverse nearest neighbor queries*, EJOR-2009
Minimum Clique Cover in Unit Disk Graph

**The Problem:** Given an unit disk graph, choose the minimum number of cliques to cover all the nodes of the graph.

**Geometric version:** Given a set of circles, choose minimum number of points on the floor such that each circle contains at least one of the chosen points.

**Computational hardness:** NP-complete \(^8\),

---

A 3-Approximation Algorithm

Observation: If the centers of the unit disks lie in a strip of width $\sqrt{3}$, then the corresponding unit disk graph is a co-comparability graph$^9$.

Algorithm:
- Split the entire plane is divided into strips of width $\sqrt{3}$.
- Compute the minimum clique cover of the unit disk graph with disks in each strip. This can be done in polynomial time.
- Output all the cliques obtained in all the strips.

Analysis of the approximation factor: Since an unit disk can span in at most 3 strips, the approximation factor of the algorithm is 3.

Maximum Independent Set in Unit Disk Graph

**The Problem:** Given a set $S$ of unit disks, find a subset $S^*$ of $S$ maximum cardinality such that the members in $S^*$ are mutually non-overlapping.

**Computational hardness:** NP-complete $^{10}$,

**A trivial 3-approximation algorithm:**

- Choose the left-most disk $C_0$.
- Choose $C_0$ in the maximum independent set. This prohibits us not to choose at most 3 non-intersecting disks, since the number of mutually non-intersecting disks intersecting $C_0$ is at most 3.
- Remove $C_0$ and all the disks overlapping on it from $S$
- Repeat the same steps if the set $S$ is not empty.

---

A 2-approximation algorithm for the maximum independent set for unit disk graph

- Divide the plane into strips of width 2. Then an unit disk can span on at most 2 strips.
- We optimally compute the maximum independent set of unit disks inside a strip as follows:

- Divide the strip into squares of side 2 by drawing vertical lines.
- Consider a pair of vertical lines $\ell_1$ and $\ell_2$, and a pair of disks intersecting to $\ell_1$ and $\ell_2$. 
A 2-approximation algorithm for the maximum independent set for unit disk graph

- For each strip optimally compute the maximum independent set using the following algorithm.
- Take the union of the max-indep-sets of either the odd strips or the even strips (which one is larger).
A 2-approximation algorithm for the maximum independent set for unit disk graph

Max-Indep-Set in a strip:
- Construct a graph $G_i = (V_i, E_i)$ with the set of circles $C_i$ in a strip $i$ as follows:

  $V_i$: \( \{c \in C_i\} \cup \{(c_\alpha, c_\beta) | c_\alpha \text{ and } c_\beta \text{ are intersected by a vertical line and they are mutually non-intersecting}\} \)

  Node weight - 1 or 2 depending on its number of circles.

  $E_i$: \( \{(v_\alpha, v_\beta) | v_\alpha, v_\beta \in V_i, v_\alpha \text{ and } v_\beta \text{ correspond to two different vertical line, and the circles of node } v_\alpha \text{ does not intersect with the circles of node } v_\beta\} \).
A 2-approximation algorithm (contd.)

**Result**

The maximum independent set of the circles in strip $i$ corresponds to the maximum weight path in $G_i$.

**Time Complexity:** $O(n^4)$

**Reason:**
- Time complexity of the maximum weight path in a directed acyclic graph - $O(m)$, $m =$ the number of edges in the graph.
- Time complexity $O(n^4) =$ worst case number of edges in $G_i$. 
Discrete Piercing Set for Unit Disks

**Problem**

We are given a set of points $P$, where each point corresponds to an unit disk centered at that point. The objective is to choose minimum number of points in $P$ such that each disk contains at least one chosen point.
A constant factor algorithm

- Partition the plane into a grid whose each cell is of size
  \( \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}} \)
A constant factor algorithm

- Partition the plane into a grid whose each cell is of size $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$
- Since the maximum distance between any two points in a grid cell is less than or equal to 1, we can pierce all the disks centered at points of $P$ in a particular cell by choosing any one member $p \in P$ lying in that cell.
A constant factor algorithm

- Partition the plane into a grid whose each cell is of size \( \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}} \).
- Since the maximum distance between any two points in a grid cell is less than or equal to 1, we can pierce all the disks centered at points of \( P \) in a particular cell by choosing any one member \( p \in P \) lying in that cell.
- It may cover point(s) in the other cell(s).

But, we show that a disk centered at a point \( p \in P \) inside a grid cell may cover (some or all) points in at most 14 other grid cells.
Algorithm

- Consider a $5 \times 5$ grid structure.
- The length of its each side of a single cell is $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$.
- The cells are numbered as $1, 2, \ldots, 25$. 
Algorithm

- Consider a $5 \times 5$ grid structure.
- The length of its each side of a single cell is $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$.
- The cells are numbered as $1, 2, \ldots, 25$.

The cell 13 is split into four parts, namely A, B, C and D.

Observation - 1
- A disk of radius 1 centered at any point in sub-cell $A$ may cover (some or all) points in only 15 cells, numbered 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18 and 19.
- The same fact can be observed for the sub-cells B, C and D.
More Detailed Observation

Theorem

A single disk centered at a point inside a cell can not cover points in more than 14 cells simultaneously.
More Detailed Observation

Theorem

A single disk centered at a point inside a cell can not cover points in more than 14 cells simultaneously.

- a single disk centered at a point \( p \in A \) can not cover points in cell number 4 and 16 simultaneously.

**Reason:** Let \( u \) and \( v \) be the bottom-left and top-right corners of the cells 4 and 16 respectively.

Thus, \( \text{dist}(u, v) = 2 \),

- Let \( p \) be a point properly inside cell A. Therefore, \( \text{dist}(u, p) + \text{dist}(p, v) > 2 \).
More Detailed Observation

Similarly, it can be shown that

- a point $p \in B$ can cover (some or all) points in cells numbered 2, 3, 4, 7, 8, 9, 10, 12, 13, 14, 15, 17, 18, 19 and 20, but it can not cover a point in cell 2 and a point in cell 20 simultaneously.

- a point $p \in C$ can cover (some or all) points in cells numbered 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 22, 23 and 24, but it can not cover a point in cell 6 and a point in cell 24 simultaneously.

- a point $p \in D$ can cover (some or all) points in cells numbered 7, 8, 9, 10, 12, 13, 14, 15, 17, 18, 19, 20, 22, 23 and 24 but it can not cover a point in cell 10 and a point in cell 22 simultaneously.
Algorithm

Select one point in each non-empty cell for the piercing.

Approximation Factor - 14

Time Complexity - $O(n \log n)$

- We shall not construct the grid explicitly.
- We maintain an height balanced binary tree $T$ for storing the non-empty grid cells.
- Each element of $T$ is a tuple $(\alpha, \beta)$ indicating the indices of a non-empty cell.
- For each point $p_i = (x_i, y_i) \in P$, we compute the indices of the grid cell $(\alpha, \beta)$; Check whether $(\alpha, \beta) \in T$; If not present, then insert $(\alpha, \beta)$ along with $p_i$.
- Finally, we visit $T$, and draw a disk with center at the point $p$ attached to each cell in $T$. 

Conclusion

- Some recent results on the *approximation algorithms* for different optimization problems on rectangle intersection graph and unit disk graphs are studied.
- The applications of those results in different practical problems are also highlighted.
- Several other application specific problems are always evolving, and the research in this direction is becoming important every day.