COMPUTING WELL-STRUCTURED SUBGRAPHS IN GEOMETRIC INTERSECTION GRAPHS

By

SATYABRATA JANA

A thesis submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

at

INDIAN STATISTICAL INSTITUTE

Supervisor

DR. SASANKA ROY

ADVANCED COMPUTING AND MICROELECTRONICS UNIT

INDIAN STATISTICAL INSTITUTE
203 B. T. ROAD, KOLKATA - 700108

July 2021

Dedication

I would like to dedicate this Thesis to my beloved PARENTS …

AUTHOR'S DECLARATION

I, Satyabrata Jana, hereby declare that this thesis titled, "Computing Well-Structured Subgraphs in Geometric Intersection Graphs" in accordance with the requirements of the Institute's Regulations and Code of Practice for "Doctoral" Degree Program and that it has not been submitted for any other academic award. I confirm that:

- The entire work of this "Thesis" was done, while in candidature for a "Doctoral" degree at this Institute,
- Where any part of this "Thesis" has previously been submitted for a degree or any other qualification at this University or any other institution, that has been properly stated.
- Where I have consulted the published work of others, that has been always clearly referred to the best of my knowledge,
- Where I have quoted from the work of others, the source is properly mentioned. With the exception of such quotations, this "Thesis" is entirely my own work,
- I have acknowledged all the main sources of help. I have also mentioned properly the work done in collaboration with, or with the assistance of others.

Satyabrata Jana

SATYABRATA JANA
KOLKATA
DATE: 31.07.2021

ACKNOWLEDGEMENT

First and foremost, I am extremely grateful to my thesis supervisor Dr. Sasanka Roy for his continuous guidance, support, and encouragement throughout my PhD journey. To work with him has been a real pleasure to me, with heaps of fun and excitement. Above all, he made me feel like a friend, which I appreciate from my heart.

It has been a pleasure to work with my co-authors Prof. Joseph S. B. Mitchell, Prof. Anil Maheswari, Dr. Mathew C. Francis, Dr. Sourav Chakraborty, Dr. Supantha Pandit, Dr. Sujoy Bhore, Dr. Saeed Mehrabi and Ms. Dalu Jacob. I would like to thank them for their efforts in our collaborative works.

I acknowledge each and every member of Advanced Computing and Microelectronics unit (ACMU), for a relaxed, family-like, and research-oriented environment.

I would like to thank my ISI hostel friends Sk Shahid Nadim, Anurag Sau, Tanujit Chakraborty, Indrajit Ghosh, Ashutosh Maurya, Narayan Rakshit, Dibyendu Sekhar Mandal, Tanmoy Das, Gopal Maity, Kaushik Kayal. The quality time I spent with them always relaxed me to overcome the stresses of research works. I am also thankful to my joyful Labmates like Ankush Acharyya, Subhash Bhagat, Avirup Das, Binayak Dutta, Sanjana Dey, Bibhuti Das, Rathindra Nath Dutta for their support.

I take this opportunity to thank my hometown friends Uttam Barik, Rahul Barik, Debendra Barik, Prasanta Bera, Manik Mahakur, Nanigopal Jana, Ratan Barik, Priyaranjan Bera for sharing enjoyable times that I normally used to spent during my stay at home.

I am also grateful to the Indian Statistical Institute, Kolkata, for giving me the opportunity to pursue the PhD, providing the fund and all necessary infrastructural resources to support my research works.

Finally, last but not the least, my deepest gratitude is to my Maa (Mother) Putul Jana and Baba (Father) Sribash Jana, for their love, care, blessings, and unconditional sacrifices for which I have no words to express. I am fortunate to have supportive and caring siblings, Kabita Jana and Debabrata Jana. I would like to express my gratitude to each and every member of our large family called “Jana Bhawan”, especially my uncles Nabadwip Jana, Achinta Jana, Subal Jana, Biren Jana, Dhiren Jana, Manik Jana, my cousins Amrita Jana and Atanu Jana for their care and unfailing emotional support.

Satyabrata Jana

Abstract

For a set of geometric objects, the associative geometric intersection graph is the graph with a vertex for each object and an edge between two vertices if and only if the corresponding objects intersect. Geometric intersection graphs are very important due to their theoretical properties and applicability. Based on the different geometric objects, several types of geometric intersection graphs are defined. Given a graph G, an induced (either vertex or edge) subgraph $H \subseteq G$ is said to be an well-structured subgraph if H satisfies certain properties among the vertices in H.

This thesis studies some well-structured subgraphs finding problems on various geometric intersection graphs. We mainly focus on computational aspects of the problems. In each problem, either we obtain polynomial-time exact algorithm or show NP-hardness. In some cases, we also extend our study to design efficient approximation algorithms and fixed-parameter tractable algorithms.

We study the construction of the planar Manhattan network (between every pair of nodes there is a minimum-length rectilinear path) of linear size for a given convex point set.

We consider the maximum bipartite subgraph problem, where given a set S of n geometric objects in the plane, we want to compute a maximum-size subset $S' \subseteq S$ such that the intersection graph of the objects in S' is bipartite.

We consider a variation of stabbing (hitting), dominating, and independent set problems on intersection graphs of bounded faces of a planar subdivision induced by a set of axis-parallel line segments in the plane.

We investigate the problem of finding a maximum cardinality uniquely restricted matching (having no other matching that matches the same set of vertices) in proper interval graphs and bipartite permutation graphs.

Finally, we consider the balanced connected subgraph problem on red-blue graphs (the color of each vertex is either red or blue). Here the goal is to find a maximum-sized induced connected subgraph that contains the same number of red and blue vertices.

Table of Contents

LIST OF FIGURES

List of Notations

$\mathbb{N}$	set of natural numbers.
$\mathbb{R}$	set of real numbers.
$\mathbb{R}^2$	the plane.
$[n]$	$\{1,2,\ldots,n\}$
$G(V,E)$	graph with vertex set V with edge set E
$V(G)$	vertex set of graph G
$E(G)$	edge set of graph G
G^c	complement of graph G
(u,v)	edge between the vertices u and v
$G[U]$	subgraph induced by U in G
$N(v)$	open neighborhood of a vertex v
$N[v]$	closed neighborhood of a vertex v
K_n	cliques of size n
$K_{m,n}$	complete bipartite graphs with partitions of size m and n
$x(p)$	x-coordinate of a point p
$y(p)$	y-coordinate of a point p
$x(L)$	x-coordinate of a vertical segment L
$y(L)$	y-coordinate of a vertical segment L
$\mathsf{proj}_L(p)$	orthogonal projection of a point p on the line containing the segment L
$\overline{pq}$	the line segment with end point p and q
$\|pq\|_1$	distance between the points p and q in L_1-norm
$\|pq\|_2$	distance between the points p and q in L_2-norm
$W_G(u,v)$	length of a shortest path between u and v in G
$\|X\|$	cardinality of set X
$\pi_G(x,y)$	set of all shortest L_1-paths between a pair of vertices x and y in the G
$\langle x,y\rangle$	an arbitrary shortest L_1-path between a pair of vertices x and y
$A \uplus B$	disjoint union of sets A and B
$A \triangle B$	symmetric difference between A and B, i.e., $(A\setminus B)\cup(B\setminus A)$

List of Acronyms

PTAS	Polynomial-time Approximation Scheme
FPT	Fixed-Parameter Tractable
OPT(Π)	size of an optimal solution of the problem Π
$\mathscr{OCP}(S)$	Ortho-Convex Polygon of a point set S
$\mathscr{H}(\mathscr{M})$	histogram partition of a polygon $\mathscr{M}$
MBS	Maximum Bipartite Subgraph
OCT	Odd Cycle Transversal
FVS	Feedback Vertex Set
MTFS	Maximum Triangle-free Subgraph
MIS	Maximum Independent Set
RP3SAT	Rectilinear Planar 3SAT
BCS	Balanced Connected Subgraph
BCS	Balanced Connected Path
EC3Set	Exact-Cover-by-3-Sets
STPG	Steiner Tree problem in planar graphs
RST	Rectilinear Steiner Tree
Ham-Path	Hamiltonian Path

CHAPTER 1

INTRODUCTION

Contents

In the real world scenario, there are many types of networks such as social networks, transportation networks, information networks which can be modeled as graphs. For example, think for an instance that consists of a set of people from a state, and from the instance, we want to select the maximum number of people who are pairwise unknown to each other. A natural model for this scenario is to construct a graph such that each person corresponds to one vertex, and we join an edge between two vertices whenever the corresponding pair of people know each other. Now our goal corresponds to finding a maximum independent set, i.e., a maximum-sized set of pairwise non-adjacent vertices in the resulting graph. Unfortunately, there is no polynomial-time algorithm that solves the problem for all instances. Like this, several problems in graph theory and combinatorial optimization involve determining if a given graph has a subgraph satisfying certain properties [W$^+$96]. Thus a typical graph theory problem consists of selecting the best among potential vertices or edges, subject to the constraints requiring that the corresponding vertices (or, edges) satisfy certain conditions. The objective of the problem is to minimize (or, maximize) the effective cost which is the number of the chosen vertices (or, edges). Examples include seeking vertex cover, matching, independent set, clique, feedback vertex set, odd cycle transversal, etc with optimum size. The optimal subgraph finding problems have a significant theoretical interest and arise in various practical applications of graph algorithms. In this thesis, we introduce a notion of *well-structured subgraph* and we are mainly concerned about finding an optimal size well-structured subgraph in a given graph. In most of cases, we try to solve a graph theoretic problem in the graphs that is restricted to a certain class. A graph class is a set of graphs that have a common property. For example, the class of bipartite graphs is the graphs that have no odd cycle. Below we define well-structured subgraphs and geometric intersection graphs, a class of graphs corresponding to a similar type of geometric objects.

1.1 Well-structured Subgraph

Let $G = (V, E)$ be a given graph. Now $H = (V', E')$ is called as subgraph of G if $V' \subseteq V$ and $E' \subseteq E$. A vertex induced subgraph of G is defined by $V' \subseteq V$ together with $E' = \{(u,v) : u, v \in V' \ \& \ (u,v) \in E\}$ and an edge induced subgraph of G is defined by $E' \subseteq E$ together with their incident vertices. Now we define well-structured subgraph. An induced (either vertex or edge) subgraph $H \subseteq G$ is said to be a well-structured subgraph if H satisfies certain property among the vertices in H. Following are examples of such subgraphs.

- Independent set (vertex induced): no pair of vertices are adjacent to each other
- Clique set (vertex induced): each pair of vertices are adjacent to each other
- Matching (edge induced): set of edges without common vertices
- spanning tree (edge induced): connectedness, acyclic

Note that, vertex cover, dominating set, feedback vertex set are examples of subgraphs that are not well-structured as there is no certain property that should be satisfied among the vertices in the subgraph. In particular, the definitions of these substructures also involve edges or vertices that are not in the subgraph H.

1.2 Geometric Intersection Graphs

An intersection graph is a graph such that each vertex corresponds to a set and there is an edge between two vertices if and only if their corresponding sets intersect. Any graph can be represented as an intersection graph over some sets. In this thesis, we consider geometric intersection graph families. It consists of intersection graphs for which the underlying collection of sets is restricted to geometric objects. Some of the important graph classes in this family that we consider in this thesis are interval graphs (intervals on the real line), circular-arc graphs (arcs on a circle), permutation graphs (line segments with endpoints lying on two parallel lines), unit-disk graphs (unit-disks in $\mathbb{R}^2$), unit-square graphs (unit-squares in $\mathbb{R}^2$), outer-string graphs (curves lying inside a disk, with one endpoint on the boundary of the disk), rectangle intersection graphs (rectangle in $\mathbb{R}^2$). In the past several decades, geometric intersection graph classes became very popular and they were extensively studied due to their interesting theoretical properties and applicability. There are many graph-theoretic problems that are NP-Hard for general graphs but polynomially solvable for a special class of geometric intersection graphs. For example, the clique decision problem is NP-Complete for general graphs [Kar72], however polynomially solvable for interval graphs [IA83], circular-arc graphs [IA83], permutation graphs [PW10], unit-disk graphs [CCJ91]. Our hope is to exploit the geometric properties of these restricted graph families to achieve theoretical results. This serves our two-fold interests behind this direction of research. It enables us to design algorithms and obtain theoretical results. This thesis is a study of the computational aspects of various well-structured subgraph finding problems with several new and other little-studied parameters mainly (we also study them for few other class of graphs as well) in different geometric intersection graph classes. Formally, the general framework of the problems we consider is as follows:

Well-structured Subgraph Finding Problems: Given a geometric intersection graph G , a problem Π, and a property π, find an optimum sized subgraph $H \subseteq G$ such that H is a feasible solution of Π in G and satisfy the property π among the vertices in H.

1.3 Scope of the Thesis

As mentioned earlier, we mainly study various well-structured subgraph finding problems. The problems that we consider in the thesis are motivated by facility location, art gallery problems, guarding regions, geometric spanners, graph matching, odd cycle transversal problems, etc. We analyze the complexity classes of the problems for different graph classes, mainly focus on several geometric intersection graph classes. We obtain polynomial-time exact algorithm for some problems. At the same time, we produce NP-Hardness results in some of the other problems. We also extend our study to design efficient approximation algorithms and fixed-parameter tractable algorithms. The main contributions of the thesis comprise the following problems.

Manhattan Network for Convex Point Sets

Let $G = (V, E)$ be a graph drawn in the plane such that every edge is horizontal or vertical. The weight of each edge $uv \in E$ is the length of the segment. Let $W_G(u,v)$ denote the length of the shortest path between a pair of vertices u and v in G. For a pair of points p and q, a Manhattan path between p,q is a rectilinear path whose length is equal to the distance between p and q in the L_1 metric. For a given point set, a graph G is said to be a Manhattan network for P if $P \subseteq V$ and between every pair of nodes in P there is a Manhattan path. In the Manhattan network problem, the objective is to construct a Manhattan network of small size for a set of n points. This problem was first considered by Gudmundsson et al. [GKKS07]. They give a construction of a Manhattan network of size $\Theta(n \log n)$ for general point sets in the plane. We study the construction of a Manhattan network for a given convex point set. More specifically, we consider the following problem.

- **Problem 1:** Given set S of n points, is given as input where the points are in convex position in $\mathbb{R}^2$, construct a planar Manhattan network for S using a linear (in terms of cardinality of S) number of Steiner points.

Considering an arbitrary point set P of size n as input, we can always construct a Manhattan network (say, G_∞) such that between any pair of points in P there are infinitely many Manhattan path between them in the network G_∞. Now we can claim that the Problem 1 is a well-structured subgraph finding problem due to the sense that we have to choose the edges in G_∞ such that the edge induced subgraph, say H in G_∞ satisfy the property: H contains a Manhattan path between each pair of points in P.

Maximum Bipartite Subgraphs

Odd Cycle Transversal (OCT) and Feedback Vertex Set (FVS) problems are two well-studied problems in graph theory. Given a graph $G = (V, E)$, the objective of the OCT problem is to find a minimum cardinality of $U \subseteq V$ such that $G[V \setminus U]$ is odd-cycle free, whereas the goal of FVS is $G[V \setminus U]$ is cycle free. We have considered the following problem which is closely related two these two above-mentioned problems, but in geometric settings.

- **Problem 2:** Given a set S of n geometric objects in the plane, we want to compute a maximum-size subset $S' \subseteq S$ such that the intersection graph of the objects in S' is bipartite, i.e., $G[S']$ is odd-cycle free.

By the definition of the problem, we have to find an optimum sized vertex induced subgraph, say $G[S']$ such that it satisfy bipartiteness property among S'. So, we can easily say that the Problem 2 is a well-structured subgraph finding problem. We have studied this problem on circular-arcs, unit squares, unit disks, and unit-height rectangles, and on some variants of unit-disks. We obtained several approximation algorithms along with some algorithmic results.

Covering and Packing of Rectilinear Subdivision

We study variations of some well-studied problems such as Set Cover, Independent Set, and Dominating Set problems in geometric settings. In the Set Cover problem, we are given a set of points and a set of geometric objects such that they together contain the set of points, and our aim is to find a minimum cardinality collection of objects that contains all of the input points. In the Independent Set problem, a set of objects is given as input and the goal is to find a maximum cardinality subset of objects that are pairwise independent, i.e., non-intersecting. In the Dominating Set problem, a set of objects is given as input and goal and choose a minimum cardinality subset of objects such that each of the remaining objects intersects with some of the chosen objects. We consider the following problems in the thesis.

- **Problem 3:** Given a set of m axis-parallel line segments that induce a planar subdivision $\mathscr{F}$, find a minimum cardinality set of points in the plane such that each face (closed) in F is intersected by one of the selected points (STABBING-SUBDIVISION), select a maximum cardinality subset $F' \subseteq F$ of faces (closed) such that any pair of faces in F' is non-intersecting (INDEPENDENT-SUBDIVISION), and lastly seek a minimum cardinality subset $F' \subseteq F$ of faces (closed) such that any face in $F \setminus F'$ has a non-empty intersection with a face in F' (DOMINATING-SUBDIVISION).

Earlier we mentioned that an independent set is a well-structured subgraph. So by the definition of the Problem 3, we can claim that INDEPENDENT-SUBDIVISION problem is a well-structured subgraph finding problem. Here we mainly show the NP-Hardnesses on the problems we consider.

Uniquely Restricted Matchings

Let $G = (V,E)$ be a graph. A set of edges $M \subseteq E(G)$ is said to be a *matching* if no two edges of M share a common vertex. A matching M in a graph G is said to be uniquely restricted if there is no other matching in G that matches the same set of vertices as M. We have studied the following problem in proper interval graphs and bipartite permutation graphs.

- **Problem 4:** Given a graph $G = (V,E)$, find a maximum cardinality matching $M \subseteq E$ such that M is uniquely restricted.

Now we can say that the Problem 4 is a well-structured subgraph finding problem due to the sense that in the problem we need to choose the edges in G such that the edge induced subgraph, say G' in G satisfy the property: there is no other matching in G' matches all the vertices in $V(G')$. We obtain linear-time algorithms for computing maximum cardinality uniquely restricted matchings in proper interval graphs and bipartite permutation graphs.

Balanced Connected Subgraphs

In the Graph Motif problem, given a graph $G = (V,E)$, a color function $col : V \to 2^{\mathscr{C}}$ on the vertices, and a multiset M of colors from $\mathscr{C}$, the objective is to find a subtree $T \subseteq G$ and a coloring that assigns a color from $col(v)$ to each vertex v in T, such that T carries exactly (also with respect to multiplicity) the colors in M. Note that if $\mathscr{C} = \{\text{red, blue}\}$, the function col gives one color of $\mathscr{C}$ to each vertex of G and the motif has same number of blues and reds then the solution of the Graph Motif problem gives a balanced connected subgraph (not necessarily a maximum balanced connected subgraph) of G. We have studied the following problem on red-blue graphs (each vertex is colored by either red or blue).

- **Problem 5:** Given a graph $G = (V,E)$, with each vertex in the set V having an assigned color, “red” or “blue” as input, our goal is to find a maximum-cardinality subset $V' \subseteq V$ of vertices that is color-balanced (having equal number of red and blue vertices), such that $G[V']$ is connected.

By the definition of the problem, we have to find a maximum sized vertex induced subgraph, say G'' such that it satisfy balancedness and connectedness property among the vertices in G'. So we can claim that the Problem 5 is a well-structured subgraph finding problem. We have considered this problem on bipartite graphs, chordal graphs, planar graphs, trees, split graphs, etc along with some well known geometric intersection graphs like interval, circular-arc, permutation, unit-disk, outer-string graphs, etc. We mainly present several hardness and algorithmic results.

1.4 Organization of the Thesis

Following is the organization of this thesis which consists of nine chapters. In Chapter 2, we describe basic notations, a few definitions, and concepts that have been used throughout the thesis. In this chapter, we define each of the graph classes formally. Chapter 3 presents the literature survey of the works relevant to this thesis. The next five chapters are devoted to five specific problems that we mentioned in the scope of the thesis. More specifically, in Chapter 4, we study on the construction of the planar Manhattan network of linear size for a given convex point set. Next in Chapter 5, we consider maximum bipartite subgraph problem, where given a set S of n geometric objects in the plane, we want to compute a maximum-size subset $S' \subseteq S$ such that the intersection graph of the objects in S' is bipartite [JMMR20]. Chapter 6 is devoted to a variation of stabbing (hitting), dominating, and independent set problem on intersection graphs of bounded faces of a planar subdivision induced by a set of axis-parallel line segments in the plane [JP19, JP20]. Chapter 7 is based on the problem of finding a maximum cardinality uniquely restricted matching in proper interval graphs and bipartite permutation graphs [FJJ18]. In Chapter 8, we consider balanced connected subgraph problem on red-blue graphs [BJPR19, BCJ$^+$21, BCJ$^+$19]. Finally, in Chapter 9, the thesis ends with mentioning some possible directions for further research.

CHAPTER 2

Preliminaries

Contents

We consider only finite graphs. Wherever it is not specified otherwise, "graph" shall mean an undirected graph. We use $G = (V,E)$ to denote a simple undirected graph G on the vertex set V with edge set E. We shall denote the vertex set and edge set of a graph G by $V(G)$ and $E(G)$, respectively. All graphs considered are simple, i.e., there are no loops or multiple edges. We make use of (u,v) and uv interchangeably to denoted an edge between the vertices u and v. For a subset of vertices $U \subseteq V$, we use $G[U]$ to denote the subgraph induced by U in a graph G. The open and closed neighborhood of a vertex v in a grah G are denoted by $N(v)$ and $N[v]$, respectively, where $N(v) = \{u : (u,v) \in E\}$ and $N[v] = N(u) \cup \{v\}$. Here we mention few definitions and concepts that have been used in this thesis.

2.1 Some Simple Graphs

Below we mention some simple undirected graph classes that have been considered throughout the thesis. For an introduction to graph theory one may refer to [W+96]. Also, a well-organized survey of more than 200 graph classes can be found in [BS+99].

Definition 2.1 (Geometric Graphs)**.** A geometric graph is a graph that is drawn in the plane so that the vertices are represented by points and the edges are represented by line segments connecting the corresponding points.

Definition 2.2 (Planar Graphs)**.** A planar graph is a graph that has a planar embedding that means it can be drawn in the plane so that there is no intersection on the edges except at their endpoints.

Definition 2.3 (Plane Graphs)**.** A plane graph is a planar graph drawn with its planar embedding.

Definition 2.4 (k-Plane Graphs)**.** A geometric graph $G = (V,E)$ is said to be a k-plane graph for some $k \in N$ if E can be partitioned into k disjoint subsets, $E = E_1 \uplus \cdots \uplus E_k$, such that $G_1 = (V,E_1),\ldots,G_k = (V,E_k)$ are all plane graphs, where $\uplus$ represents the disjoint union.

Definition 2.5 (Cubic/3-Regular/Trivalent Graphs)**.** A cubic graph is a graph in which all the vertices have degree three.

Definition 2.6 (Subcubic Graphs)**.** A subcubic graph is a graph in which all the vertices have degrees at most three.

Definition 2.7 (Tolerance Graphs)**.** A graph is a tolerance graph if every vertex v of the graphs can be assigned a closed interval I_v on the real line and a tolerance t_v such that x and y are adjacent iff $|I_x \cap I_y| \geq \min\{t_x, t_y\}$.

Definition 2.8 (Clique Separable Graphs)**.** A graph is clique separable if every primitive (it has no clique cutset) induced subgraph of the graph is either complete k-partite or can be partitioned into a connected bipartite graph A and a clique B such that there is a join between A and B.

Definition 2.9 (i-Triangulated Graphs)**.** A graph is i-triangulated if every odd cycle of length at least 5 has at least two non-crossing chords.

Definition 2.10 (Split Graphs)**.** A split graph is a graph in which the vertices can be partitioned into a clique and an independent set.

Definition 2.11 (Threshold Graphs)**.** A graph is a threshold graph if it can be constructed from the empty graph by repeatedly adding either an isolated vertex or a dominating vertex. Equivalently, a graph is a threshold graph if there is a real number S (the threshold) and for every vertex v there is a real weight a_v such that: vw is an edge iff $a_v + a_w \geq S$.

Definition 2.12 (Block Graphs)**.** A graph is a block graph if every block (maximal 2-connected component) is a clique. Equivalently, block graphs are exactly the graphs for which, for every four vertices u, v, x, and y, the largest two of the three distances $d(u,v) + d(x,y), d(u,x) + d(v,y)$, and $d(u,y) + d(v,x)$ are always equal.

Definition 2.13 (k-Connected Graphs)**.** A k-connected graph is a graph such that it is not possible to make it disconnected by removing $k-1$ vertices.

Definition 2.14 (Cactus/Cacti Graphs)**.** A cactus is a connected graph in which any two simple cycles have at most one vertex in common.

Definition 2.15 (Star Graphs)**.** A star S_k is the complete bipartite graph $K_{1,k}$.

Definition 2.16 (Claw Graphs)**.** A claw is the star S_3.

Definition 2.17 (Perfect Graphs)**.** A perfect graph is a graph in which the chromatic number of every induced subgraph equals the size of the largest clique of that subgraph.

Definition 2.18 (Line Graphs)**.** The line graph of an undirected graph G is another graph $L(G)$ that represents the adjacencies between edges of G.

Definition 2.19 (Comparability Graphs)**.** A graph is a comparability graph if its edges can be oriented such that the orientation connects pairs of elements that are comparable to each other in a partial order.

Definition 2.20 (Co-comparability Graphs)**.** A graph is co-comparability if and only if the complement graph of the graph is comparability. Equivalently, A graph is a co-comparability if it is the intersection graph of curves from a line to a parallel line.

Definition 2.21 (Graphs of diameter k)**.** A graph has a diameter k if the distance between any pair of vertices is at most k. Here, the distance between a pair of vertices is defined as the number of edges in a shortest path between them.

2.2 Geometric Intersection Graphs

The class of geometric intersection graphs has been studied for many years for its theoretical aspects as well as for its applications. This class of graphs now appears in discrete and computational geometry. It has deep connections with complexity theory [SSŠ03, Sch09] and order dimension theory [Fel14, CHO$^+$14, CFHW18]. The geometric intersection graphs are basically intersection graphs of geometric objects. More precisely, given a collection O of geometric objects, the geometric intersection graph of O is the graph with a vertex for each object and between a pair of vertices, there is an edge between them if the corresponding objects in O intersect. So every geometric intersection graph has some underlying geometric objects. In this thesis, we focus on the following classes of geometric intersection graphs where all underlying geometric objects should lie in the plane.

Definition 2.22 (Interval Graphs)**.** A graph G is called an interval graph if and only if there exists a set I of intervals such that G is the Intersection Graph of I. The set of intervals is called the interval representation of G.

Definition 2.23 (Proper Interval Graphs)**.** Proper interval graphs are interval graphs that have an interval representation in which no interval properly contains any other interval.

Definition 2.24 (Circular-arc Graphs)**.** A graph G is called a circular-arc graph if and only if there exists a set A of arcs of a circle, such that G is the Intersection Graph of A. The set of arcs is called the circular-arc representation of G.

Definition 2.25 (Outerstring Graphs)**.** A graph G is called outerstring graph if and only if there exists a set C of curves lying inside a disk with one endpoint on the boundary, such that G is the Intersection Graph of C. The set of curves is called the outerstring representation of G.

Definition 2.26 (Permutation Graphs)**.** A graph G is called a permutation graph if and only if there exists a pair of parallel lines, a set L of line segments such that endpoints of each segment lie on the parallel lines, and G is the Intersection Graph of L. The set of segments along with the parallel lines is called the representation of G.

Definition 2.27 (Unit Disk Graphs)**.** A graph G is called a unit disk graph if and only if there exists a set D of unit disks such that G is the Intersection Graph of D. The set of intervals is called the unit disk representation of G.

Definition 2.28 (Unit Square Graphs)**.** A graph G is called a unit square graph if and only if there exists a set S of unit squares such that G is the Intersection Graph of S. The set of squares is called the unit square representation of G.

Definition 2.29 (Unit-height Rectangle Graphs)**.** A graph G is called a unit-height rectangle graph if and only if there exists a set R of unit-height rectangles such that G is the Intersection Graph of R. The set of rectangles is called the unit-height rectangles representation of G.

2.3 Complexity Theory

In complexity theory, a decision problem deals with "yes" or "no" question for an input. Below we mention some basic complexity classes, defined on decision problems, that will be used throughout the thesis.

Complexity Class P

P is a complexity class that represents the set of all decision problems that can be solved in polynomial time, i.e., in time $O(n^{O(1)})$. Thus, given an instance of the problem, the answer yes or no can be decided in polynomial time.

Complexity Class NP

NP (nondeterministic polynomial time) is a complexity class that represents the set of all decision problems having the property that their solutions can be verified in polynomial time.

Reducibility

For a pair of problems Π_1 and Π_2, we say problem the Π_1 is polynomial time reducible to the problem Π_2, if there exists a polynomial time computable function f from the instance set of $I(\Pi_1)$ of Π_1 to the instance set $I(\Pi_2)$ of π_2, such that $Y \in I(\Pi_1)$ is a yes-instance for Π_1 if and only if $f(Y) \in I(\Pi_2)$ is a yes-instance for Π_2.

Complexity Class NP-hard

NP-Hard is the complexity class that represents the set of all decision problems to which all problems in NP can be reduced to in polynomial time by a deterministic Turing machine.

Complexity Class NP-Complete

A problem Π that is said to be NP-Complete if and only if Π is in NP and every other problem in NP is polynomial time reducible to the problem Π.

For a detailed discussion on the topic of complexity theory and the theory of NP-completeness, one may refer to [GJ79b, AB09].

2.4 Asymptotic Analysis

The efficiency of an algorithm depends on the amount of time (time complexity), storage or memory (space complexity), and other resources required to implement the algorithm. To analyze the efficiency, we need to do an asymptotic analysis. For that, asymptotic notations can help us to measure efficiency. For a particular problem, an algorithm may not have the same performance for different types of inputs. It depends on the type of the input, size of the input, etc. Hence, asymptotic analysis is a study of change in performance for an algorithm with the change in the order of the input size. There are mainly five asymptotic notations.

- Big Oh notation (O): asymptotic upper bound
- Big Omega notation (Ω): asymptotic lower bound
- Theta notation (Θ): asymptotic tight bound
- Little Oh notation (o): asymptotic upper bound that cannot be tight
- Little Omega notation (ω): asymptotic lower bound that cannot be tight

Now we describe the notations. Below is the table that summarizes the key restrictions to make the definition of notations: Let $f(n)$ and $g(n)$ be a pair of functions from $\mathbb{N}$ to $\mathbb{R}\setminus\{0\}$. We say $f(n) \in \langle notation\rangle(g(n))$, if $\langle condition\rangle$ $c > 0$ where $c \in \mathbb{R}$, and there exists an integer constant $n_0 > 1$, such that $f(n)\langle relation\rangle$ $c \times g(n)$, for every integer $n \geq n_0$. For example, $f(n) \in O(g(n))$, if $\exists$ $c > 0$ where $c \in \mathbb{R}$, and there exists an integer constant $n_0 > 1$, such that $f(n) \leq c \times g(n)$, for every integer $n \geq n_0$. "Big-Theta" is combination of $O()$ and $\Omega()$. More specifically, $f(n) \in \Theta(g(n))$ if and only if $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

Family of notations		
notation	***condition***	***relation***
O	$\exists$	$\leq$
o	$\forall$	$<$
Ω	$\exists$	$\geq$
ω	$\forall$	$>$

2.5 Fixed Parameter Algorithm

A central notion in Parameterized Complexity is fixed-parameter tractability which is defined below. In the context of this thesis, we could obtain fixed-parameter tractable algorithms in some of the chapters to be discussed later.

Definition 2.30 (Parameterized Problem)**.** An instance of a parameterized problem is a tuple (X,k), where X is the given instance of the problem and k is called the parameter.

Definition 2.31 (Fixed-parameter Tractable)**.** A parameterized problem Π is called fixed-parameter tractable if there is an algorithm such that, given an instance $I = (X,k)$ of the problem, the algorithm correctly decides whether or not $(X,k) \in \Pi$ in time $O(f(k)poly(|X|))$, where $f(k)$ is an arbitrary (may be exponential) function on k that does not at all involve $|X|$, and $poly(|X|)$ is a polynomial function in $|X|$. The complexity class containing all fixed-parameter tractable problems is called FPT.

For a detailed discussion on the topic of parameterized complexity theory, one may refer to [CFK$^+$15].

2.6 Approximation Algorithm

In the thesis, we work on obtaining approximation algorithms for several problems on different geometric intersection graphs. Below we mention the various types of approximation schemes.

Definition 2.32 (Approximation Algorithms)**.** Let Π be a maximization (resp. minimization) problem. An algorithm A for the problem Π is called an t factor approximation algorithm for Π if and only if for any instance X of Π, A runs in time polynomial in $|X|$ and produces a feasible solution, such that $\mathsf{OPT}(X) \leq t \times A(X)$ (resp. $A(X) \leq t \times \mathsf{OPT}(X)$). Here $A(X)$ and $\mathsf{OPT}(X)$ are the values of the output of the algorithm and optimum solution of the problem Π for the given instance X, respectively.

Definition 2.33 (Polynomial-Time Approximation Scheme)**.** Let Π be a maximization (resp. minimization) problem. An algorithm A for the problem Π is a polynomial-time approximation scheme (PTAS) for Π if and only if for any instance X of Π and for any fixed $\epsilon > 0$, A runs in time polynomial in $|X|$ and produces a feasible solution, such that $A(X) \geq (1-\epsilon) \times \mathsf{OPT}(X)$ (resp. $A(X) \leq (1+\epsilon) \times \mathsf{OPT}(X)$). Here $A(X)$ and $\mathsf{OPT}(X)$ are the values of the output of the algorithm and optimum solution of the problem Π for the given instance X, respectively.

For a detailed discussion on the topic of approximation algorithms, one may refer to [Vaz01].

CHAPTER 3

Review and Related Works

Contents

3.1 Construction of Geometric Spanner

A graph G is said to be a Manhattan network for a given point set P in the plane if $P \subseteq V$ and for each pair of points p and q in P, $W_G(p,q) = \|pq\|_1$, where $W_G(u,v)$ denotes the length of the shortest path between a pair of vertices u and v in G. In addition to P, graph G may also include a set T of Steiner points in its vertex set V.

The Minimum Manhattan Network (MMN) problem on P is to construct a Manhattan network for P of minimum possible length. By length we mean the sum of all edge lengths in G, i.e., $\sum\limits_{e=(u,v)\in E} \|uv\|_1$. Below in Figure 3.1, we show examples of a Manhattan network and a minimum Manhattan network, respectively, on a set P of points.

The MMN problem problem has a wide number of applications in city planning, network layouts, distributed algorithms [NS07], VLSI circuit design [GLN01], and

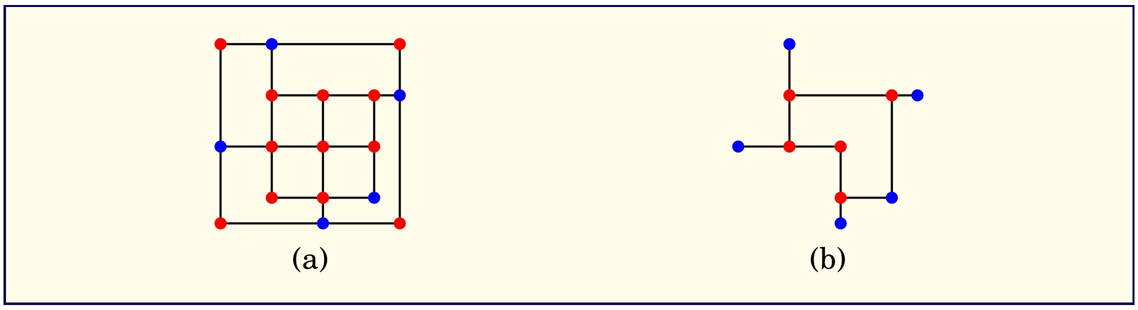


Figure 3.1: Blue circles represent the points in P, and red circles represent Steiner points. (a) A Manhattan network, and (b) a minimum Manhattan network.

computational biology [LAP03]. It was first introduced in 1999 by Gudmundsson et al. [GLN01]. Several approximation algorithms (with factors 4 [GKKS07], 2 [KIA02], and 1.5 [SU05]) with time complexity $O(n^3)$ have been proposed in the last few years, where n denotes the size of the given point set. Also, there are $O(n \log n)$ time approximation algorithms with factors 8 [GKKS07], 3 [BWWS06], and 2 [GSZ11]. Recently Chin et al. [CGS11] proved that the decision version of the MMN problem is strongly NP-Complete and Knauer et al. [KS11] presented a fixed-parameter algorithm running in $O^*(2^{14h})$ time (neglecting a factor that is polynomial in n) where the parameter h is the minimum number of axis-parallel straight-lines (either all horizontal or all vertical) that contain all the points in P.

In 2007, Gudmundsson et al. [GKKS07] considered a variant of the MMN problem where the goal is to minimize the number of vertices (Steiner points) and edges. In $O(n \log n)$ time, they constructed a Manhattan network with $O(n \log n)$ vertices and edges using a divide and conquer strategy. They also proved that there are point sets in $\mathbb{R}^2$ where every Manhattan network for these point sets will need $\Omega(n \log n)$ vertices and edges.

We mention two other problems related to Manhattan Network. One is the generalized minimum Manhattan network (GMMN) problem, and the other one is the rectilinear Steiner arborescence (RSA) problem. In the GMMN problem, given a set R of n pairs of terminals, which are points in $\mathbb{R}^2$, the target is to find a minimum-length rectilinear network similar to a Manhattan network with the sole difference that only the pairs in R are required to be connected by Manhattan paths. This problem is known to be NP-Hard. Also, there is an $O(\log n)$-factor approximation algorithm for the GMMN problem [DFK+18]. The RSA problem is defined as the special case of GMMN where the origin appears in every pair (hence, every terminal has to be connected by a Manhattan path to the origin) and all terminals lie in the first quadrant. This is also known to be NP-Hard [SS05]. It also admits a 2-factor approximation algorithm [RSHS92] and a PTAS (shown independently by Zachariasen [Zac00] and Lu and Ruan [LR00]).

A closely related problem is to construct a geometric spanner for a given point set. For a real number $t \geqslant 1$ and a point set S, a geometric graph $G = (S, E)$ is said to be a t-spanner of S if for any two points p and q in S, $W_G(p,q) \leqslant t|pq|$, where $|pq|$ is the Euclidean distance between two points p and q. The stretch factor of G is the smallest real number t such that G is a t-spanner of S. A number of algorithms have been proposed for constructing t-spanners for any given point set [NS07, AdBF+11]. Keil et al. [KG89] showed that the Delaunay triangulation of S is a 2.42-spanner of S. For convex point sets, Cui et al. [CKX11] proved that the Delaunay triangulation has a stretch factor of at most 2.33. For general point sets, Xia [Xia13] provided a 1.998-spanner, having a linear number of edges, applying Delaunay triangulation. Steiner points are also used for constructing spanners. For example, Arikati et al. [ACC+96] used Steiner points to answer exact shortest-path queries between any two vertices of a geometric graph. The authors also considered the problem of finding an obstacle-avoiding L_1-path between a pair of query points in the plane. They found a $(1+\epsilon)$-spanner with space complexity $O\left(n^2/\sqrt{r}\right)$, preprocessing time $O(n^2/\sqrt{r})$ and $O(\log n + \sqrt{r})$ query time, where ϵ is an arbitrarily small positive constant and r is an arbitrary integer, such that $1 < r < n$. Recently, Amani et al. [ABB+16] showed how to compute a plane 1.88-spanner in L_2-norm for convex point sets in $O(n)$ time without using Steiner points. For general point sets of size n, Gudmundsson et al. [GKKS07] constructed a $\sqrt{2} \approx 1.41$-spanner (which may not be planar) in the L_2-norm and it uses $O(n \log n)$ Steiner points. As a corollary of our construction in the thesis, for a convex point set, we obtain a planar $\sqrt{2} \approx 1.41$-spanner in the L_2-norm using $O(n)$ Steiner points (Chapter 4). Given a rectilinear polygon with n vertices, Schuierer [Sch96] presented a data structure that can report a shortest path (in the L_1-metric) for any pair of query points in that polygon in $O(\log n + k)$ time where k is the number of segments in the shortest path. De Berg [DB91] proved that given two arbitrary points inside a polygon, the L_1-distance between them can be reported in $O(\log n)$ time.

3.2 Maximum Bipartite Subgraph Problem

The Maximum Bipartite Subgraph (MBS) problem is defined as follows. Given a graph $G = (V, E)$, the goal is to compute a maximum-size subset $V' \subseteq V$ such that the induced subgraph $G[V']$ is bipartite. This MBS problem is NP-Complete for planar graphs with maximum degree four [CNR89]. For graphs with maximum degree three, Choi et al. [CNR89] showed that for a given constant k there is a vertex set of size k or less whose removal leaves an induced bipartite subgraph if and only if there is an edge set of size k or less whose removal leaves a bipartite spanning subgraph. As edge deletion graph bipartization problem is NP-Complete for cubic graphs [Yan78], the MBS problem is NP-Complete for cubic graphs. Moreover, the maximum edge deletion graph bipartization problem is solvable in $O(n^3)$ time for planar graphs [Had75, AI77] where n denotes the number of vertices of the input graph. Therefore, MBS is $O(n^3)$ time solvable for planar graphs with maximum degree three. For the vertex-weighted version of the MBS problem, Baiou et al. [BB16] showed that the MBS problem can be solved in $O(n^{3/2}\log n)$ time for planar graphs with maximum degree three. Cornaz et al. [CM07] studied a weighted variant of this problem: given a graph with non-negative weights on the edges, the goal is to find a maximum-weight bipartite subgraph. An edge subset $F \subseteq E$ is called independent if the subgraph induced by the edges in F (incident vertices) is bipartite; otherwise, it is called dependent. They showed that the minimum dependent set problem with non-negative weights can be solved in polynomial time.

The MBS problem is closely related to the Odd Cycle Transversal (OCT) problem. More explicitly, MBS and OCT are equivalent for the class of graphs on which OCT is polynomial-time solvable: an exact solution S for OCT gives $V(G)\setminus S$ as an exact solution for MBS within the same time bound. Given a graph $G=(V,E)$, the objective of the OCT problem is to compute a minimum-cardinality $U \subseteq V$ such that $G[V\setminus U]$ is bipartite. The OCT problem is known to be NP-Complete on planar graphs with degree at most 6 [CNR89]. For planar graphs with degree at most 3, OCT can be solvable in $O(n^3)$ time [CNR89] (even the weighted version of the problem). There are several results known concerning the parameterized complexity of OCT (i.e., given a graph G on n vertices and an integer k, is there a vertex set U in G of size at most k such that $G\setminus U$ is bipartite). Reed et al. [RSV04] first gave an algorithm with running time $O(4^k kmn)$. Lokshtanov et al. [LSS09] improved this running time to $O(3^k kmn)$. For planar graphs, Lokshtanov et al. [LSW12] provided an algorithm with running time $O(2^{O(k\log k)}n)$. Moreover, assuming the exponential time hypothesis [CEF12], i.e., 3SAT cannot be solved in subexponential time, the running time cannot be improved to $2^{O(k)}n^{O(1)}$.

Another problem that is related to MBS is the Feedback Vertex Set (FVS) problem. The objective of FVS is to compute a minimum-cardinality $U \subseteq V$ such that

$G[V \setminus U]$ is acyclic. In the last 30 years, various works on the complexity of feedback vertex set problems for different type of graphs have been published. Let us briefly give an overview of the complexity of FVS problem on related graph classes. This problem is known to be NP-Complete on bipartite graphs [Yan81] and planar graphs [GJ79b]. A minimum (weight) FVS can be computed by a polynomial time algorithm for bounded clique-width graphs ($O(n^2 2^{O(w \log w)})$) [BSTV13], chordal graphs [CF89, Spi03], interval graphs ($O(m+n)$) [LT97], permutation graphs ($O(mn)$) [Bra92, BK85, BK87, Lia94], co-comparability graphs ($O(n^2 m)$) [LC97], convex bipartite graphs ($O(n^2 m)$) [LC97], AT-free graphs ($O(n^8 m^2)$) [KMT08], Trapezoid Graph ($O(n^{2.68} + \gamma n)$) [HKM11], normal Helly circular-arc graph ($O(n+m+\gamma)$) [HNS16], circular-arc graphs [Spi03] where w is the clique width, γ depends on the number of maximal cliques, m and n are the number of edges and vertices of the corresponding graph. Also, there are exact algorithms that find a minimum FVS in a graph of n vertices in time $O(1.8899^n)$ [Raz06] and in time $O(1.7548^n)$ [FGPR08]. Concerning approximation algorithm, FVS approximating a minimum one within a constant factor can be efficiently found in undirected graphs. There are 2-factor approximation algorithm for general graph with running time $O(\min\{|E|\log|V|, |V|^2\})$ [BBF99] and $2-(2/(3\Delta-2))$-factor approximation algorithm for graphs with maximum degree Δ in $O(|V|^2)$ time [BBF95]. While considering parameterized (FPT) version of this problem: given a graph on n vertices and an integer k, the feedback vertex set problem asks for the deletion of at most k vertices to make the graph acyclic, several results have been obtained. Cao [Cao18] gave an $O(c^k n^2)$ (for some constant c) time algorithm to solve this. Chen at al. [CFL+08] provided an $O(5^k k n^2)$ time algorithm. To our knowledge, the running time of fastest FPT algorithm is $O(3.619^k n^{O(1)})$ [KP14].

In the maximum k-colorable subgraph (MkCS) problem, we are given a graph $G = (V,E)$, and a positive integer k; the objective is to find a maximum sized-subset $U \subseteq V$ such that the vertices in U can be colored using k colors and no two adjacent vertices of U in G are assigned the same color. When $k = 1$, the MkCS problem is the same as the maximum independent set (MIS) problem. Given a graph G, the objective of MIS is to seek a maximum-cardinality subset of vertices such that no pair of them are adjacent. The MkCS problem with $k = 2$ can be viewed as the MBS problem. Lewis and Yannakakis [LY80] showed that for any k, MkCS problem is NP-Hard. For an arbitrary k, this problem remains NP-Hard for chordal graphs [YG87]. However, this problem is polynomial time solvable for chordal graphs for fixed k [YG87], interval graphs for any k [YG87], circular-arc graphs for $k = 2$ [Nar89], tolerance graphs for $k = 2$ [Nar89], clique-separable graphs for $k = 2$ [ABKK+10], and i-triangulated graphs for any k [ABKK+10]. Considering interval graphs, this problem is polynomial time solvable using the algorithm designed by Frank [Fra80]. It was shown that a simple greedy algorithm can solve M2CS (i.e., MBS) problem for interval graphs in $O(|V|+|E|)$ time [YG87]. Later Narasimhan [Nar89] designed a linear time algorithm for this problem assuming that the intervals are sorted with respect to their right endpoints. Lu and Tang [LT97] studied the MBS problem on weighted interval graphs where the goal is to find an induced bipartite subgraph with maximum weight. They obtain an $O(|V|+|E|)$-time algorithm for this problem.

3.3 Covering and Packing

The set cover problem [CLRS09] is a classical and fundamental question in computer science. This problem is usually formulated in terms of hypergraphs: the input of the problem is a hypergraph $\mathscr{H} = (X, \mathscr{F})$ where $\mathscr{F} \subseteq 2^X$ is a collection of subsets of X and our goal is to find a subset $\mathscr{F}' \subseteq \mathscr{F}$ of smallest cardinality that covers X. It is one of Karp's 21 NP-Complete problems shown to be NP-Complete in 1972 [Kar72]. This problem is even NP-Hard to approximate [CLRS09, Fei98].

The Set Cover, Independent Set, and Dominating Set problems are NP-Hard for simple geometric objects such as disks [FPT81], squares [FPT81], rectangles [FPT81], etc. We refer to page 5 for definitions of these problems in the geometric setting. There is a long line of research of these problems and its various variants and special cases [HM85a, MR10, MRR14, CH12, AW13, MP15, CE16, Pan17, vL09].

Korman et al. [KPR18] studied an interesting variation of the Set Cover problem, the Line-Segment Covering problem. In this problem, they cover all the cells of an arrangement formed by a set of line segments in the plane using a minimum number of line segments. Here covering a cell with a segment means the segment is a part of the boundary of the cell. They showed that the problem is NP-Hard, even when all segments are axis-aligned. In fact, they also proved that it is NP-Hard to cover all rectangular cells of the arrangement by a minimum number of axis-parallel line segments. In [GIK02], Gaur et al. studied the rectangle stabbing problem. Here given a set of rectangles, the objective is to stab all rectangles with a minimum number of axis-parallel lines. They provided a 2-factor approximation algorithm for this problem.

We mention another related problem called the art gallery problem. Here the goal is to place guards so that the guards together see the whole gallery. Many variants of this have been studied. Bose et al. [BCC$^+$12] studied guarding and coloring problems between lines. In [DS08], Dom et al. study the parameterized complexity of the rectangle stabbing problem and its variants. In d-dimensional rectangle stabbing problem, we are given a set of axis-parallel d-dimensional hyperrectangles, a set of axis-parallel $(d-1)$-dimensional hyperplanes and a positive integer k; the question is whether one can select at most k hyperplanes so that every hyperrectangle is intersected by at least one of them. The authors showed that the case $d \geq 3$ is $W[1]$-hard with respect to the parameter k. In [CRCS+94], Czyzowicz et al. condisered the guarding problem in rectangular art gallerys. They showed that if a rectangular art gallery is divided into n rectangular rooms, then $\lceil n/2 \rceil$ guards are always sufficient to protect all rooms in that rectangular art gallery. They also extended their result in non-rectangular galleries and 3-dimensional art galleries [CRCS+94].

Chen [Che01] studied the problem of finding a maximum independent set in a given map graph (on both vertex-weighted and unweighted graphs). A map graph is an intersection graph of faces in the plane, where two vertices share an edge of the graph if and only if their corresponding faces have at least one point in common. A map graph is a k-map graph if on its corresponding map has no more than k faces meet at a point. It is known that the Independent Set problem on planar graphs is NP-Hard. In [CGP98, CGP02], the authors showed that each planar graph is exactly a 3-map graph. Thus the unweighted version of the Independent Set problem restricted to map graphs is also NP-Hard. When the underlying maps are also given along with the map graphs, Chen [Che01] provided a PTAS for the unweighted Independent Set problem. Thorup [Tho98] gave a polynomial-time algorithm that constructs a map from a given map graph. However, this algorithm is very complicated and its running time is very high. In [Che01], Chen gave a PTAS for the k-map graph for some fixed integer k. The author also provided a PTAS for the Independent Set problem on the vertex weighted k-map graph for some fixed integer k. Further, Chen obtained two nontrivial polynomial-time approximation algorithms for the Independent Set problem on map graphs. Interestingly, in all of these cases, the algorithm did not construct a map for the given map graph. Recently, Fomin et al. [FLP+19] obtained parameterized subexponential time algorithms for various problems on map graphs.

Mustafa et al. [MR10] considered the problem of computing minimum geometric hitting sets in which, given a set of geometric objects and a set of points, the goal is to compute the smallest subset of points that hit all geometric objects. The problem is known to be strongly NP-Hard even for simple geometric objects like unit disks in the plane.

In [CGK+17], Claverol et al. considered the problem of stabbing line segments with rectilinear objects. In this problem, given a set S of n line segments in the plane, we say that a region $R \subseteq \mathbb{R}^2$ is a stabber for S if R contains exactly one endpoint of each segment of S . The authors of [CGK+17] provided optimal algorithms for reporting all combinatorially different stabbers for several shapes of stabbers. The problem of computing stabbing circles of a set S of n line segments in the plane has been studied very recently by Claverol et al. [CKP+18]. Edelsbrunner et al. [EMP+82] showed that the geometric problem of determining a line (called a stabbing line) which intersects each of n given line segments in the plane is solvable in $O(n \log n)$ time.

3.4 Uniquely Restricted Matching

Let $G = (V, E)$ be a graph. A set of edges $M \subseteq E(G)$ is said to be a *matching* if no two edges of M share a common vertex. The set of vertices in V that have an edge of M incident on them are called the *vertices matched by M*. A matching M is said to be *uniquely restricted* if there is no other matching that matches the same set of vertices as M. Golumbic at al. [GHL01] proved that the problem of finding a maximum cardinality uniquely restricted matching in an input graph is NP-Complete even for the special cases of split graphs and bipartite graphs. The authors also presented linear time algorithms for the problem on threshold graphs, proper interval graphs, cacti, and block graphs. In Chapter 7, we will show that the linear-time algorithm described for the problem for proper interval graphs in [GHL01] does not appear to work in all cases. This problem also has been studied in the approximation paradigm. Mishra [Mis11] showed that for bipartite graphs, the problem is hard to approximate within a factor of $O(n^{\frac{1}{3}-\epsilon})$, for any $\epsilon > 0$, unless NP = ZPP, and also that the problem is APX-Complete even for bipartite of degree at most 3. For 3-regular bipartite graphs, there was a 2-factor approximation algorithm for this problem [Mis11]. Later Baste et al. [BRS19] improved it and produced a 5/9-factor approximation algorithm for subcubic bipartite graphs. Penso at al. [PRdSS18] showed that the graphs having the property that every maximum matching is uniquely restricted can be recognized in polynomial time. Even if they showed that for a given graph, in polynomial time it is possible to check whether some maximum matching in the graph is uniquely restricted. Chaudhary and Panda [CP20] studied a variation of this problem called Min-Max-UR Matching problem. Here the goal is to find a maximal uniquely restricted matching of minimum cardinality in the given graph. The author showed that this problem is NP-Hard for chordal bipartite graphs. Further, they obtained an exact polynomial-time algorithm for bipartite permutation graphs and proper interval graphs. They also showed that this problem is APX-Complete for bounded degree graphs.

3.5 Balanced Subgraph Problem

Bichromatic input points, often referred to as "red-blue" input, has appeared extensively in numerous problems in computational geometry. Recently, Bandyapadhyay et al. [BBBN20] studied four fundamental graph-theoretic problems: Hamiltonian path, Traveling salesman, Minimum spanning tree, and Minimum perfect matching on geometric graphs induced by bichromatic (red and blue) points. Many of these problems are NP-Hard on Euclidean plane [Aro98, BvKL+09]. In [BBBN20], the authors showed almost all of these problems can be solved in linear time in two restricted settings such as colinear points and equidistant points on a circle. For a detailed survey on geometric problems with red-blue points see [KK03]. In [BMS14, DK01, DP02], colored points have been considered in the context of matching and partitioning problems. In [AAFM+15], Aichholzer et al. considered the balanced island problem and devised polynomial algorithms for points in the plane. On the combinatorial side, Balanchandran et al. [BMMP17] studied the problem of unbiased representatives in a set of bicolorings. Kaneko et al. [KKW17] considered the problem of balancing colored points on a line. Later on, Bereg et al. [BHK+15] studied balanced partitions of 3-colored geometric sets in the plane. In a biological population, vertex-colored graphs can be used to represents the connections and interactions between species where different species have different colors. In [FFHV11, IMTP18], the authors mentioned that the vertex colored graph problems have numerous applications in bioinformatics. Now, we define one problem, which is on vertex colored graph, called Graph Motif problem [LFS06, FFHV11, BS17].

In the Graph Motif problem, we are given a graph $G = (V,E)$, a coloring $col : V \to \mathscr{C}$ of the vertices in V where $\mathscr{C}$ is a set of colors, and a multiset M of colors of $\mathscr{C}$; the objective is to find a subset $V' \subseteq V$ such that the induced subgraph on V' is connected and $col(V') = M$, where $col(V')$ denotes the multiset of colors of vertices in V'. The basic motivation of the Graph Motif problem comes from the domain of biological network analysis [LFS06]. This problem has applications in social, technical networks [FFHV11, BvBF+11] and in the context of mass spectrometry [FFHV11, BRS09]. Fellows et al. [FFHV11] showed that the Graph Motif problem is NP-Complete for trees of maximum degree 3 where the given motif is a colorful set instead of a multiset (that is, no color occurs more than once). They also proved that the Graph Motif problem remains NP-Hard for bipartite graphs of maximum degree 4 and the motif contains only two colors. In this thesis, we will study balanced connected subgraph (BCS) problem. Here, a red-blue graph is given as input and our goal is to find a maximum-sized induced subgraph that is balanced (contains an equal number of red and blue vertices) as well as connected. We note that if $\mathscr{C} = \{\text{red, blue}\}$ and the motif has a same number of blues and reds, then the solution of the Graph Motif problem gives a balanced connected subgraph (not necessarily a maximum balanced connected subgraph). It is easy to observe that a solution to the Graph Motif problem (essentially) gives a solution to the BCS problem, with an impact of a polynomial factor in the running time. On the other hand, the NP-Hardness result for the BCS problem on a particular graph class implies the NP-Hardness result for the Graph Motif problem on the same class. So, the BCS problem is a special case of the Graph Motif problem.

The BCS problem is closely related to the Maximum Node Weight Connected Subgraph (*MNWCS*) problem [Joh85, EK14]. In the *MNWCS* problem, we are given a connected graph $G(V,E)$, with an integer weight associated with each vertex (node) in V, and an integer bound B; the objective is to decide whether there exists a subset $V' \subseteq V$ such that the subgraph induced by V' is connected and the total weight of the vertices in V' is at least B. In the *MNWCS* problem, if the weight of each vertex is either $+1$ (red) or -1 (blue), and if we ask for a largest connected subgraph whose total weight is exactly zero, then it is equivalent to the BCS problem. The *MNWCS* problem along with its variations have numerous practical applications in various fields (see [EK14] and the references therein). We believe some of these applications also serve well to motivate the BCS problem.

Kobayashi et al. [KKM$^+$19] provided an exact exponential-time algorithm of BCS problem for general graphs (in $2^{n/2}n^{O(1)}$ time). The authors also considered a weighted version of BCS (called as WBCS) problem and showed weakly NP-Hardnesses on star graphs and strongly NP-Hardness on split graphs and properly colored bipartite graphs. Darties et al. [DGKP19] proved the NP-Completeness of the decision variant of BCS problem in bounded-diameter and bounded- degree graphs: bipartite graphs of diameter four, graphs of diameter three and bipartite cubic graphs. Recently, Bhore et al. [BHK$^+$20] considered a new version of independent set and dominating set problem on vertex colored interval graphs, called f-Balanced Independent Set (f-BIS) and f-Balanced Dominating Set (f-BDS). Given a vertex-colored graph with k colors, a subset of vertices is said to be f-balanced if it contains f vertices from each color class. In the f-BIS and f-BDS problems, the goal is to find an independent set and a dominating set, respectively, that is f-balanced. They showed NP-Completeness for both the problems on proper interval graphs.

CHAPTER 4

MANHATTAN NETWORK FOR CONVEX POINT SETS

Contents

Related Publication:

1. Satyabrata Jana, Anil Maheshwari, and Sasanka Roy. "Linear Size Planar Manhattan Network for Convex Point Sets." In *Computational Geometry: Theory and Applications*, 2021.

4.1 Introduction

In computational geometry, constructing a minimum-length Manhattan network is a well-studied topic [GLN01]. Let $G = (V, E)$ be an edge-weighted geometric graph such that every edge is horizontal or vertical. The weight of an edge in E is the L_1-distance between its endpoints. For a pair of vertices u and v in G, we use $W_G(u,v)$ to denote the length of a shortest path between u and v in G.

Definition 4.1 (Manhattan network)**.** For a given point set P in the plane, a graph G is said to be a Manhattan network of P if $P \subseteq V$ and for every $p,q \in P$, $W_G(p,q) = \|pq\|_1$, where $\|pq\|_1$ is the distance between p and q in L_1-norm.

In other words, G is said to be a Manhattan network of P if $P \subseteq V$ and for each pair of points in P there exists a Manhattan path (or, shortest L_1-path), that is, a path of axis-parallel line segments whose total length equals the pair's L_1-distance. The graph G may also include a set T of Steiner points in its vertex set V.

A set of points is said to be a convex point set if all of the points are vertices of their convex hull. A Manhattan network is said to be planar if there exists a planar embedding (embedded in the plane without any edge crossings). It is said to be plane if itself is a planar embedding. Gudmundsson et al. [GKKS07] showed that there exists a convex point set for which a plane Manhattan network requires $\Omega(n^2)$ vertices and edges. We describe their construction: Let P be the set of $4(n-1)$ points in the plane defined as follows (see Figure 4.1(a)):

$$P = \bigcup_{i=1}^{n-1} \{(i,0),(i,n),(0,i),(n,i)\}$$

If G is a plane Manhattan network for P, then there must be a shortest L_1-path between every pair of points $(i,0),(i,n)$ and $(0,i),(n,i)$, where i is an integer satisfying $1 \leqslant i \leqslant (n-1)$. These paths need to be orthogonal straight-line segments because in the first case, the x-coordinates are the same, and in the second case, the y-coordinates are the same. This would force us to add Steiner points at all the $\Theta(n^2)$ intersection points. For an illustration, see Figure 4.1(b).

A natural question that arises is what if we want the network to be planar (and not necessarily plane)? For the above example, we can construct a planar Manhattan network $G = (V = P \cup T, E)$ of $O(n)$ size. Figure 4.1(c) depicts a planar Manhattan network for the above example that uses four Steiner points $q_{00} = (0,0)$, $q_{0n} = (0,n)$, $q_{n0} = (n,0)$, $q_{nn} = (n,n)$. Its planarity can be verified in Figure 4.1(d) where a corresponding planar embedding is given. The correctness of the construction, that is G is a Manhattan network, can be trivially seen by looking at Figure 4.1(d). For example, for the pair $((0,n-2),(n,2))$ there is a Manhattan path, in G, consisting of the edges $((0,n-2),(n,n-2))$ and $((n,n-2),(n,2))$.

In this work, we focus on the following problem:

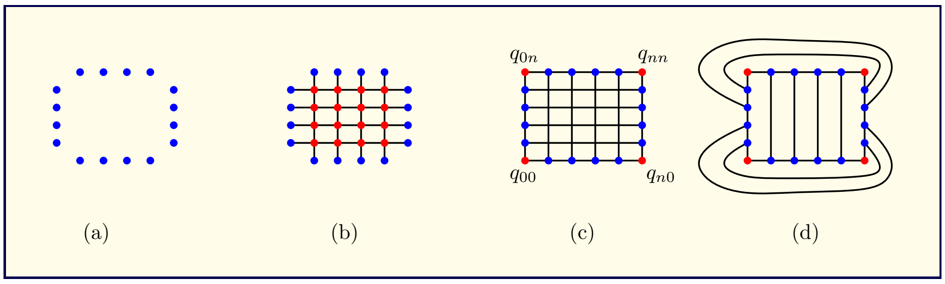


Figure 4.1: Blue circles represent the points in P, and red circles represent Steiner points. (a) The point set P, (b) construction of a plane Manhattan network for the point set P, (c) a planar Manhattan network G^* of P, and (d) a planar embedding of G^* where the curved edges have weight/length n.

A set S of n points, is given as input where the points are in convex position in $\mathbb{R}^2$,

➩ construct a planar Manhattan network for S using a linear (in terms of cardinality of S) number of Steiner points.

Showing planarity of the network is important as the all-pairs shortest-path computation is much faster on planar graphs than on general graphs [Fre91]. Also, the running time to compute shortest paths does not depend on a planar embedding; the planarity condition is sufficient.

4.1.1 Our contributions

In linear time, we construct a planar Manhattan network G for a convex point set S of size n, assuming the points of S are given in sorted order along their convex hull. The network G uses $O(n)$ Steiner points as vertices. Our network is also a $\sqrt{2}$- spanner in L_2-norm. We also show that the construction in Gudmundsson et al. [GKKS07] needs $\Omega(n \log n)$ points even for a convex point set. Also, their network might not be planar even for a convex point set.

4.2 Manhattan Network for General Point Sets

In this section, we compare our algorithm with the best-known Manhattan network algorithm for general point sets by Gudmundsson et al. [GKKS07]. We consider the case that the input is a convex point set and we observe that the algorithm of Gudmundsson et al. needs asymptotically more Steiner points in the worst case and in contrast to our network, the network costructed by Gudmundsson et al. might not be planar.

For general point sets, Gudmundsson et al. [GKKS07] proved the following theorem.

Theorem 4.1. *[GKKS07] Let P be a set of n points. A Manhattan Network for P consisting of $\Theta(n \log n)$ vertices and edges can be computed in $O(n \log n)$ time.*

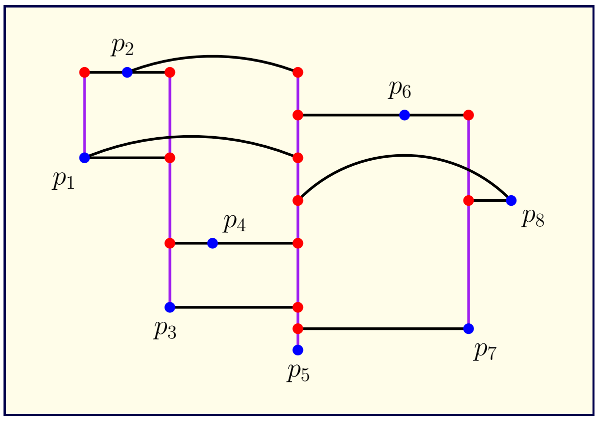


Figure 4.2: Construction of the Manhattan network for S. Points in S are in blue color and Steiner points are in red color.

Their construction is as follows: Assuming that no two points in P are on the same vertical line, sort the points in P according to their x-coordinate. Let m be the median x-coordinate in P. Then draw a vertical line L_m through $(m,0)$. For each point p of P, take an orthogonal projection on the line L_m. Add Steiner points at each projection and join p with its corresponding projection point. Then recursively do the same, on the points that have smaller x-coordinate than p and on the points that have greater x-coordinate than p. Add a Steiner point at each projection. Figure 4.2 illustrates the algorithm of Gudmundsson et al. [GKKS07].

Now, we make two observations on the construction of Gudmundsson et al. [GKKS07]. First, there are convex point sets where this construction needs $\Omega(n \log n)$ Steiner points. For instance, this is the case for any convex point set of size n where no two points have the same x or y- coordinate. According to the construction of Gudmundsson et al. [GKKS07], the vertical line L_m through $(m,0)$ would contain the projection of all points p of P. This process would run recursively on two subsets S_l and S_r of points. Here $S_l = \{q \in S | x(q) < x(m)\}$ and $S_r = \{q \in S | x(q) > s(m)\}$. Thus most of the points are projected onto $O(\log n)$ different vertical lines and the result holds. Secondly, there are convex point sets for which this construction is not planar. Figure 4.3 depicts such an instance of 16 points in convex position. The Manhattan network, as computed by the algorithm of Gudmundsson et al. [GKKS07], has a minor homeomorphic to $K_{3,3}$. Hence, it is not planar. We summarize our observations of Gudmundsson et al.'s algorithm in the lemma below.

Lemma 4.1. *There exist convex point sets for which the algorithm of Gudmundsson et al. [GKKS07] produces a non-planar Manhattan network. Also, for every n, there is a convex point set of n points where this construction needs $\Omega(n \log n)$ Steiner points.*

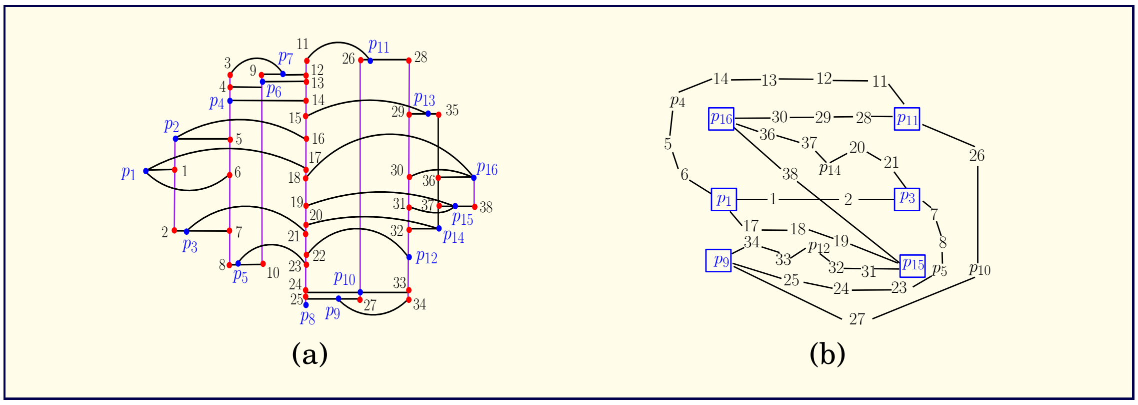


Figure 4.3: (a) The Manhattan network G_A of a convex point set $A = \{p_1, \ldots, p_{16}\}$ (blue color) as produced by the algorithm of Gudmundsson et al. [GKKS07]. Points colored in red are Steiner points. (b) A subgraph of G_A that is homeomorphic to $K_{3,3}$ (vertices of $K_{3,3}$ are depicted as boxes/rectangles).

Thus, our algorithm that we present in Section 4.3 is an improvement for convex point sets as it uses only $O(n)$ Steiner points and is always planar.

4.3 Planar Manhattan Network **for Convex Point Sets**

In this section, we construct a linear-size planar Manhattan network G for a convex point set S. The network G uses $O(n)$ Steiner points and can be constructed in linear time. We organize this section as follows. In Section 4.3.1, we construct a histogram partition $\mathscr{H}(\mathscr{OCP}(S))$ of an ortho-convex polygon $\mathscr{OCP}(S)$ of the convex point set S. In Section 4.3.2, we construct our desired graph $G = (V, E)$ where $S \subseteq V$. We also show that the graph G is of linear size, and it can be computed in linear time. Section 4.3.3 is devoted to prove that the graph G is a Manhattan network for S. In Section 4.3.4, we show that G is planar. Our main idea is as follows: Draw a straight-line Manhattan network for S where crossings occur only between vertical and horizontal edges. To get a planar embedding, we then redraw all, say, vertical edges by non-intersecting curves within the exterior face around the convex hull. Hence, only horizontal edges remain in the convex hull's interior and, thus, all crossings are eliminated.

A polygonal chain, with n vertices in the plane, is defined as an ordered set of vertices $(v_1,\ldots,v_n)$, such that any two consecutive vertices v_i, v_{i+1} are connected by the line segment $\overline{v_i v_{i+1}}$, for $1 \leqslant i < n$. It is said to be closed when it divides the plane into at least two disjoint regions with $v_1 = v_n$. A simple polygon is a bounded region that is enclosed by a closed polygonal chain in $\mathbb{R}^2$. In our definition, we consider that the region of a polygon includes both its boundary and interior. A line segment is orthogonal if it is parallel either to the x-axis or to the y-axis.

Definition 4.2. (Orthogonal polygon) A polygon is said to be orthogonal if all of its sides are orthogonal.

Definition 4.3. (Ortho-convex polygon)[DR90] An orthogonal polygon $\mathscr{P}$ is said to be ortho-convex if every horizontal or vertical line segment connecting a pair of points in $\mathscr{P}$ lies entirely in $\mathscr{P}$.

Definition 4.4. (Shortest L_1-path) A path between two points p and q is said to be a shortest L_1-path between them if the path consists of orthogonal line segments with total length $\|pq\|_1$.

Lemma 4.2. *[CNV08] For all point pairs in an ortho-convex polygon $\mathscr{P}$, there exist a shortest L_1-path between them in $\mathscr{P}$.*

4.3.1 Ortho-convex polygon $\mathscr{OCP}(S)$ and histogram partition $\mathscr{H}(\mathscr{OCP}(S))$

Let $S = \{p_1,\ldots,p_n\}$ be a convex point set of size n in $\mathbb{R}^2$. For any point $p \in S$, let $x(p)$ and $y(p)$ be its x- and y-coordinate[1], respectively. We assume that the points in S are ordered with respect to a counter-clockwise orientation along their convex hull. Without loss of generality, let this ordering be $p_1,\ldots,p_n$ and also assume that p_1 is the top-most point in S, i.e., the point having the largest y-coordinate in S (for multiple points having the largest y-coordinate, we take the one that has the smallest x-coordinate). We denote the top-most point of S as t. Analogously, let ℓ, b, and r denote the left-most (for multiple points, we take the one that has the smallest y-coordinate), the bottom-most (for multiple points, we take the one that has the smallest x-coordinate) and the right-most point of S (for multiple points,

[1]we assume the coordinate axes are horizontal and vertical in the plane where the x-axis is oriented to the right and the y-axis is oriented to the bottom.

we take the one that has the smallest y-coordinate), respectively. So $t = p_1$. The overall idea to construct an ortho-convex polygon $\mathscr{OCP}(S)$ for S is as follows: we connect the points in order such that between any pair of consecutive points there is at most one Steiner point such that the set of all Steiner points is convex. Now we describe the construction in detail. We will consider the point set for the case that $x(p_1) \leqslant x(b)$. For the case of $x(p_1) \geqslant x(b)$, both the construction and the proof are symmetric (by vertical reflection with respect to the x-axis). A polygonal chain is said to be xy-monotone if any orthogonal line segment intersects the chain in a connected set. Now we will construct an ortho-convex polygon $\mathscr{OCP}(S)$, where the points in S lie on the boundary of $\mathscr{OCP}(S)$. The ortho-convex polygon $\mathscr{OCP}(S)$ consists of four xy-monotone chains. Let us denote these chains as $C_{t\ell}$, $C_{\ell b}$,C_{br} and C_{rt}. The chain $C_{t\ell}$ is composed of the points $p_1(=t), p_2, \ldots, \ell$. Analogously, $C_{\ell b}$, C_{br}, and C_{rt} are defined. While constructing the chain $C_{t\ell}$, we do the following: For any pair of consecutive points p_i, p_{i+1}, if $x(p_{i+1}) < x(p_i)$ and $y(p_{i+1}) < y(p_i)$ then we draw two line segments $\overline{p_i\ s_{i,i+1}}, \overline{s_{i,i+1}\ p_{i+1}}$, where $s_{i,i+1} = (x(p_{i+1}), y(p_i))$, else we draw the line segment $\overline{p_i\ p_{i+1}}$. In Algorithm 1, we describe the construction of $C_{t\ell}$. The construction for all the other monotone chains follow the same set of rules. We use $s_{i,i+1}$ to denote the Steiner point (if it exists [2]) that is in between p_i and p_{i+1} with respect to a counter-clockwise orientation in the boundary of the polygon $\mathscr{OCP}(S)$. See Figure 4.4 for an illustration.

Algorithm 1 Construction of the chain $C_{t\ell}$

Input: A set of m points $p_1(=t), p_2, \ldots, p_m(=\ell)$ such that $x(p_{j+1}) \leqslant x(p_j), y(p_{j+1}) \leqslant y(p_j)$ for $1 \leqslant j < m$

Output: The chain $C_{t\ell}$

```
1: for $j = 1$ to $(m-1)$ do
2:     if $x(p_j) = x(p_{j+1})$ or $y(p_j) = y(p_{j+1})$ then
3:         Join the points $p_j$ and $p_{j+1}$ with the line segment $\overline{p_j\ p_{j+1}}$
4:     else
5:         Create a Steiner point $s_{j,j+1} = (x(p_{j+1}), y(p_j))$
6:         Join the points $p_j$ and $p_{j+1}$ with the line segments $\overline{p_j\ s_{j,j+1}}$ and $\overline{s_{j,j+1}\ p_{j+1}}$
7:     end if
8: end for
```

In Figure 4.5, we illustrate an example of a convex point set S of size 15 and an ortho-convex polygon $\mathscr{OCP}(S)$.

Definition 4.5. (Histogram) A histogram H is an orthogonal polygon consisting of a boundary edge e, called its base, such that for any point $p \in H$, there exists a point

[2]it does not exist if p_i and p_{i+1} have a same coordinate, i.e., $x(p_i) = x(p_{i+1})$ or $y(p_i) = y(p_{i+1})$.

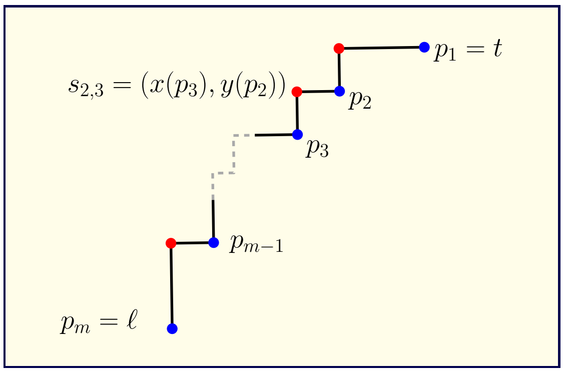


Figure 4.4: Construction of the chain $C_{t\ell}$ from a given convex point set (blue color).

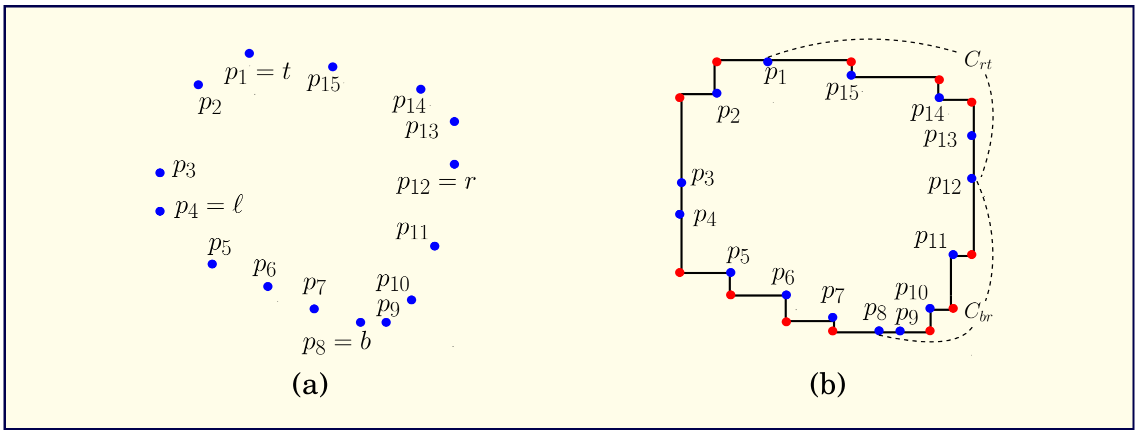


Figure 4.5: (a) Example of a set S of 12 points in convex position, (b) $\mathscr{OCP}(S)$ of S.

$q \in e$ such that the line segment $\overline{pq}$ is orthogonal and lies completely in H.

If the base is horizontal (respectively, vertical) we say that H is a *horizontal* (respectively, *vertical*) histogram. If its interior is above the base, it is called an *upper* histogram. Similarly, we can define the *lower*, *left*, and *right* histograms. Figure 4.6 depicts some examples of histograms with their bases. In this work, we have considered histograms both in general as well as in the degenerate case. Figure 4.6(b) illustrates an example of a degenerate histogram. Note that a histogram can have multiple bases. There is a base edge containing another base edge of the same histogram in a degenerate histogram.

Definition 4.6. (Histogram Partition) A histogram partition of an orthogonal polygon is a partition of the interior of that polygon into histograms.

We can always construct a histogram partition of any given orthogonal polygon [Sch96]. Now we construct a histogram partition $\mathscr{H}(\mathscr{OCP}(S))$ of $\mathscr{OCP}(S)$.

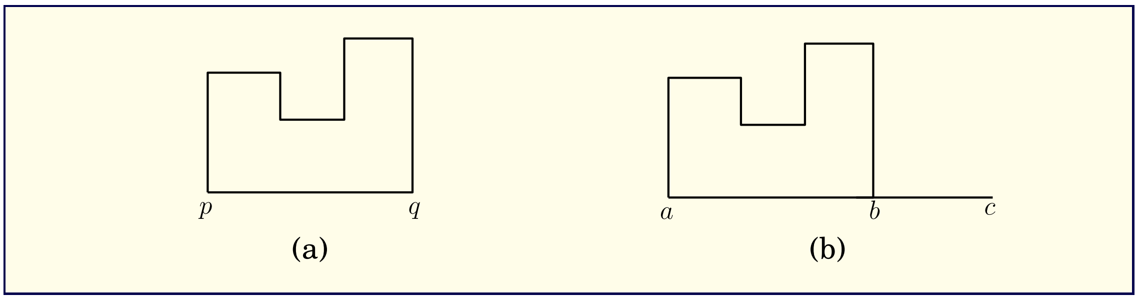


Figure 4.6: Examples of histograms. (a) general case: base $\overline{pq}$, (b) degenerate case: $\overline{ab}$ and $\overline{ac}$ are bases.

Let $L = \overline{pq}$ be a vertical line segment such that both the points p and q are on the boundary of $\mathscr{OCP}(S)$. We use H^r_L to denote a right-vertical histogram with base $L = \overline{pq}$ where this unique histogram is obtained by cutting the orthogonal polygon $\mathscr{OCP}(S)$ by L into two halves and taking the part of the right half that does not lie below $y_{\min}(L)$ or above $y_{\max}(L)$ as H^r_L. Formally we can write $H^r_L = \mathscr{OCP}(S) \cap L_r$, where $L_r = \{(x, y) : x \geqslant x(L), y_{\min}(L) \leqslant y \leqslant y_{\max}(L)\}$. Similarly, we use H^ℓ_L to denote a left-vertical histogram with the base L. For an horizontal line segment $L' = \overline{p'q'}$, in same way, we define $H^u_{L'}$ and $H^b_{L'}$ as upper-horizontal and lower-horizontal histograms, respectively, with base L'. Let $\mathsf{proj}_L(x)$ be the orthogonal projection of a point x on the line containing an horizontal or vertical segment L. For a set A of orthogonal line segments and a point set S, we say A *can see* S if for every $x \in S$ there is at least one line segment $L \in A$ such that $\mathsf{proj}_L(x) \in L$. For a vertical (resp. horizontal) line segment L, we define $x(L)$ (resp. $y(L)$) to be the x-coordinate (resp. y-coordinate) of L.

We obtain a histogram partition $\mathscr{H}(\mathscr{OCP}(S))$ of $\mathscr{OCP}(S)$ by iteratively drawing vertical and horizontal lines as follows (see Figure 4.7):

Step 1: Let q_1 be the intersection point of the boundary of $\mathscr{OCP}(S)$ with the vertical line containing p_1. Note that, as we consider the point set for the case that $x(t) \leqslant x(b)$, we have $q_1 \in C_{\ell b}$. First, we draw a vertical line segment $L_1 = \overline{p_1q_1}$. We define two sets $S(H^\ell_{L_1})$ and $S(H^r_{L_1})$ such that $S(H^\ell_{L_1}) = \{q \in S : y(t) \geqslant y(q) \geqslant y(q_1)$ and $x(q) \leqslant x(q_1)\}$, $S(H^r_{L_1}) = \{q \in S : y(t) \geqslant y(q) \geqslant y(q_1)$ and $x(q) \geqslant x(q_1)\}$. In this step, we construct two vertical histograms $H^\ell_{L_1}$ and $H^r_{L_1}$. If $S(H^\ell_{L_1}) \cup S(H^r_{L_1}) = S$, i.e., L_1 can see S we stop, else we proceed to Step 2.

Step 2: Let q_2 ($\notin C_{\ell b}$) be the intersection point of the boundary of $\mathscr{OCP}(S)$ with the horizontal line containing q_1. Then we draw a horizontal line segment $L_2 = \overline{q_1q_2}$. Here we define the set $S(H^b_{L_2}) = \{z \in S : x(q_1) \leqslant x(z) \leqslant x(q_2)$ and $y(z) \leqslant y(q_2)\}$. In this step, we construct the lower histogram $H^b_{L_2}$ with base

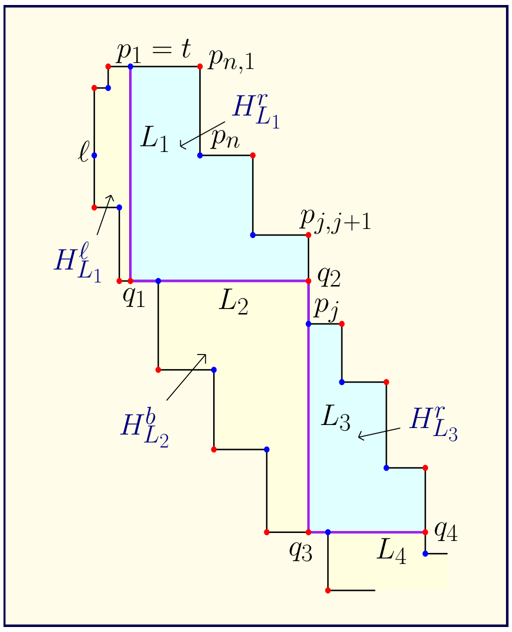


Figure 4.7: $\mathscr{H}(\mathscr{OCP}(S))$ of a convex point set S.

L_2. Note that boundary of $H^b_{L_2}$ (degenerate) is defined by the points q_1, q_2, p_j and q_3. If $S(H^\ell_{L_1}) \cup S(H^r_{L_1}) \cup S(H^b_{L_2}) = S$, i.e., $\{L_1, L_2\}$ can see S we stop, else we proceed to the next step.

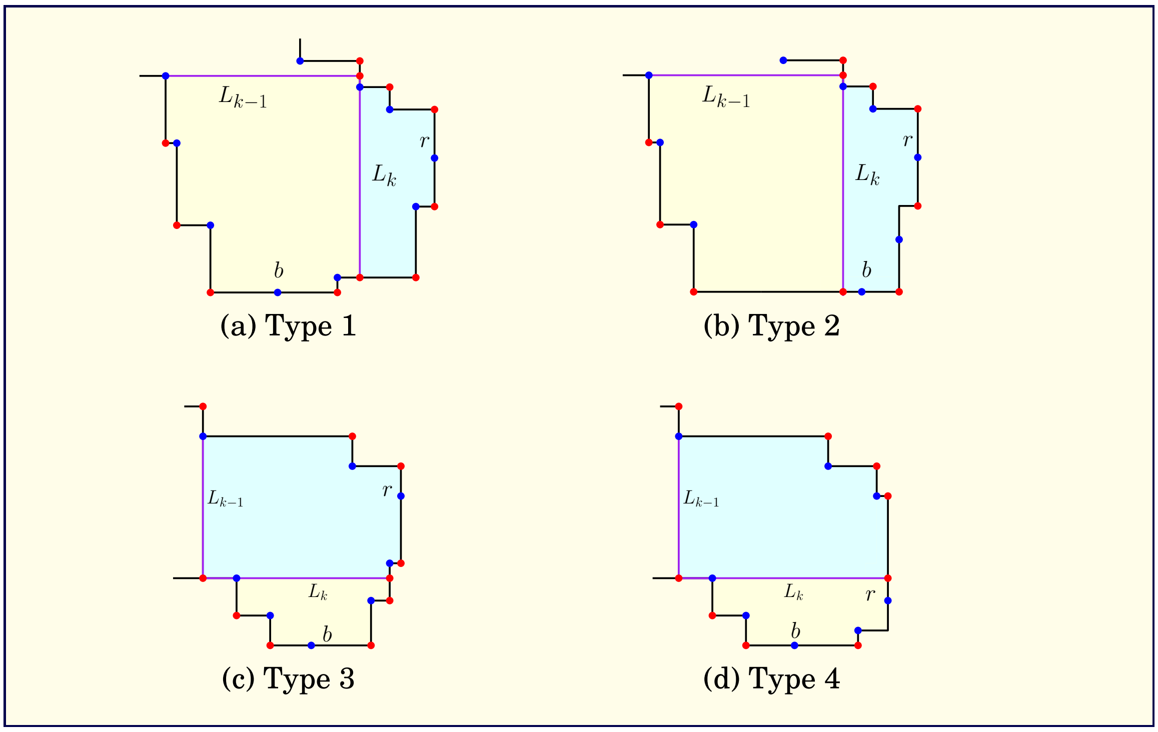
Step 3: Let q_3 ($\notin C_{rt}$) be the intersection point of the boundary of $\mathscr{OCP}(S)$ with the vertical line containing q_2. Then we draw a vertical line segment $L_3 = \overline{q_2q_3}$. Here we define the set $S(H^r_{L_3}) = \{w \in S : y(q_2) \geqslant y(w) \geqslant y(q_3) \text{ and } x(q) \geqslant x(q_3)\}$. In this step, we construct the right histogram $H^r_{L_3}$ with base L_3. Here boundary of $H^r_{L_3}$ (degenerate) is defined by the points q_2, p_j, q_3 and q_4. If $S(H^\ell_{L_1}) \cup S(H^r_{L_1}) \cup S(H^b_{L_2}) \cup S(H^r_{L_3}) = S$, i.e., $\{L_1, L_2, L_3\}$ can see S we stop, else we proceed in a similar manner.

Notice that, in each step of our construction, we draw an axis-parallel line segment through some points from S inside the polygon. Also after each step, the set of still-invisible points in S gets smaller and forms an ortho-convex polygon that we partition recursively into histograms. This ensures that our process terminates. We assume that it terminates after k steps, and we obtain a set $\mathscr{L}$ of k orthogonal line segments $\{L_1, \ldots, L_k\}$ for some $k \in \mathbb{N}$ such that $\{L_1, \ldots, L_k\}$ can see S. In this process, we add k Steiner points $\{q_i : 1 \leqslant i \leqslant k\}$. Each q_i belongs to the boundary of $\mathscr{OCP}(S)$.

The process terminates in one of the following four configurations. These four configurations depend on whether L_k is vertical or horizontal and whether b and r are contained in the same histogram with base L_k or lie in different ones (see Figure 4.8).

- **Type-1** L_k is vertical and $\text{proj}_{L_{k-1}}(b) \in L_{k-1}$, i.e., L_{k-1} sees b.
- **Type-2** L_k is vertical and $\text{proj}_{L_{k-1}}(b) \notin L_{k-1}$.
- **Type-3** L_k is horizontal and $\text{proj}_{L_{k-1}}(r) \in L_{k-1}$, i.e., L_{k-1} sees r.
- **Type-4** L_k is horizontal and $\text{proj}_{L_{k-1}}(r) \notin L_{k-1}$.

So for any point $p \in S$, there is at least one line segment $L \in \mathscr{L}$ such that $\text{proj}_L(p) \in L$ and the segment $\overline{p\ \text{proj}_L(p)}$ completely lies in $\mathscr{OCP}(S)$ where $\mathscr{L} = \cup_{i=1}^{n} L_i$. From now onwards, we assume that $L_1, \ldots, L_k$ are the segments inserted in $\mathscr{OCP}(S)$ while constructing $\mathscr{H}(\mathscr{OCP}(S))$.



(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

Figure 4.8: Types of the histograms containing b and r in $\mathscr{OCP}(S)$.

Lemma 4.3. *$\mathscr{H}(\mathscr{OCP}(S))$ is a histogram partition of $\mathscr{OCP}(S)$ and it can be constructed in linear time.*

Proof. Let $L_i(S) = \{p \in S : L_i$ can see $p\}$. First we show that $\mathscr{H}(\mathscr{OCP}(S))$ is a histogram partition of $\mathscr{OCP}(S)$, i.e., $\cup_{i=1}^{k} L_i(S) = S$. The segment L_1 sees every point $q \in S$ with the property that $y(q_1) \leqslant y(q) \leqslant y(t)$ as $\mathscr{OCP}(S)$ is an ortho-convex polygon and these points are part of the xy-monotone chains $\{C_{rt}, C_{t\ell}, C_{\ell b}, C_{br}\}$. So $L_1(S)$ consists of all the points in S that lie above L_2. Moreover, all the points above

L_2 are part of the histograms $H^{\ell}_{L_1}$ and $H^{r}_{L_1}$. Now our concern is only about the points of S that are below L_2. Now the segment L_2 can see every point $q \in (S \setminus L_1(S))$ having the property that $x(q_1) \leqslant x(q) \leqslant x(q_2)$. These points are part of the histogram $H^{b}_{L_2}$. Now we can apply the same argument iteratively. This leads to the claim that $\cup_{i=1}^{k} L_i(S) = S$, i.e., $\mathscr{L} = \{L_1, \dots, L_k\}$ can see S. Observe that the segments in $\mathscr{L}$ can be computed by walking around the boundary of $\mathscr{OCP}(S)$ in linear time. We achieve this by using two pointers moving along the border of $\mathscr{OCP}(S)$. One pointer moves counter-clockwise from ℓ to b while the other moves clockwise from t to b. (Note that we do not need to visit the points between t and ℓ as the chain $C_{t\ell}$ entirely belongs to the first histogram $H^{\ell}_{L_1}$.) Hence, the proof. ■

4.3.2 Construction of a planar Manhattan network

Now we describe our construction of a planar Manhattan network $G = (V, E)$ for a convex point set S. We construct the graph by adding, step by step, Steiner points and rectilinear edges into the drawing of $\mathscr{H}(\mathscr{OCP}(S))$. The resulting Manhattan network is straight-line, and crossings occur only between vertical and horizontal edges. We discuss this redrawing in Section 4.3.4 where we argue that the resulting drawing is planar as only horizontal edges remain in the interior of the convex hull and, thus, all crossings are eliminated. In the following, by Algorithm 2 we construct a straight-line drawing of G (Manhattan network for the convex point set S) by adding Steiner points and rectilinear edges into the drawing of $\mathscr{H}(\mathscr{OCP}(S))$. For an illustration of the steps of this algorithm, see Figure 4.9. Recall that $\mathsf{proj}_L(p)$ denotes the orthogonal projection of the point p on the line containing the segment L and further, let $H(q)$ be the set of histograms containing $q \in S$ in $\mathscr{H}(\mathscr{OCP}(S))$. In the histogram partition, there exists a unique histogram containing the point ℓ. But it may happen that $|H(b)| > 1$ (resp. $|H(r)| > 1$). If $|H(\ell)| = |H(b)| = |H(r)| = 1$, we assume that e_1, e_2, and e_3 are the bases of the unique histograms in $H(\ell), H(b)$, and $H(r)$, respectively. In the other case, when $|H(b)| > 1$ (resp. $|H(r)| > 1$), we consider L_{k-1} as e_2 (resp. e_3). First, we draw the segments $e'_1 = \overline{\ell\ \mathsf{proj}_{e_1}(\ell)}, e'_2 = \overline{b\ \mathsf{proj}_{e_2}(b)}$, and $e'_3 = \overline{r\ \mathsf{proj}_{e_3}(r)}$ in $\mathscr{OCP}(S)$. Note that b or r might be a convex vertex. For instance, suppose that in Figure 4.7(a), r is identical with the red point below it. In this special case we draw e'_3 not as a straight line segment but as a curve that is very close to the border but only touches the border in r. Or, more generally, e'_3 is a curve connecting r to $\mathsf{proj}_{e_3}(r)$ within the interior of the respective histogram. A similar kind of argument can be applied when b is convex. Let $\mathscr{L}' = \mathscr{L} \cup \{e'_1, e'_2, e'_3\}$. Next, for each $q \in S$ and each $L \in \mathscr{L}' \cap H$ where $H \in H(q)$, if $\mathsf{proj}_L(q) \in L$, we draw the line segment $\overline{q\ \mathsf{proj}_L(q)}$

in $\mathcal{OCP}(S)$. If L_k is vertical and $\mathsf{proj}_{L_{k-1}}(b) \in L_{k-1}$ (i.e., the configuration of Type 1 holds), we draw the segments $\overline{z\ \mathsf{proj}_{e'_2}(z)}$, for each point $z \in S \cap H^r_{L_k}$. Also if L_k is horizontal and $\mathsf{proj}_{L_{k-1}}(r) \in L_{k-1}$(i.e., the configuration of Type 3 holds), we draw the segments $\overline{w\ \mathsf{proj}_{e'_3}(w)}$, for each point $w \in S \cap H^b_{L_k}$. In this process, we add into E, all the line segments that we draw except $\{e'_1, e'_2, e'_3\}$. Also all the extra points we created to make an orthogonal projection, we add them into the set T of Steiner vertices. Our algorithm ends with removing some specific line segments on e'_1, e'_2, e'_3, that is stated in the Steps 27-33 in Algorithm 2. This step is required only to keep planarity of the network. We illustrate this algorithm in Figure 4.12. Recall that we use $s_{i,i+1}$ to denote the vertex (Steiner point) that is in between p_i and p_{i+1} with respect to a counter-clockwise orientation in the boundary of the ortho-convex polygon $\mathcal{OCP}(S)$. Note that $s_{i,i+1}$ does not exist if $x(p_i) = x(p_{i+1})$ or $y(p_i) = y(p_{i+1})$.

Algorithm 2 Construction of $G = (V = S \cup T, E)$

```
    Input: $\mathcal{H}(\mathcal{OCP}(S))$ of a convex point set $S = \{p_1(=t),\ p_2, \dots,\ p_n\}$.
           Let $\{L_1, \dots, L_k\}$ be the segments and $\{q_i : 1 \leqslant i \leqslant k\}$ be the set of points
           inserted in $\mathcal{OCP}(S)$ during the construction of $\mathcal{H}(\mathcal{OCP}(S))$.
    Output: A planar Manhattan network $G = (V = S \cup T, E)$ of $S$.
 1: $T \leftarrow \{\ s_{i,i+1} :\ 1 \leqslant i \leqslant (n-1)\} \cup \{s_{n,1}\} \cup \{\ q_i :\ 1 \leqslant i \leqslant k\}$;  ▷ The vertex $s_{i,i+1}$ is the
    Steiner point that is in between $p_i$ and $p_{i+1}$ with respect to a counter-clockwise
    orientation in the boundary of the ortho-convex polygon $\mathcal{OCP}(S)$. $s_{i,i+1}$ does not
    exist if $x(p_i) = x(p_{i+1})$ or $y(p_i) = y(p_{i+1})$.
 2: $E \leftarrow \{\overline{p_i s_{i,i+1}} : 1 \leqslant i \leqslant (n-1)\} \cup \{\overline{p_{i+1} s_{i,i+1}} : 1 \leqslant i \leqslant (n-1)\} \cup \overline{p_n s_{n,1}} \cup \overline{p_1 s_{n,1}}$; ▷ Add
    an edge $(p_i, p_{i+1})$ in the case that $s_{i,i+1}$ does not exist. See Figure 4.9(i)
 3: Draw the line segments $e'_1 = \overline{l\ \mathsf{proj}_{e_1}(l)}, e'_2 = \overline{b\ \mathsf{proj}_{e_2}(b)}$, and $e'_3 = \overline{r\ \mathsf{proj}_{e_3}(r)}$  ▷
    $e_1, e_2$, and $e_3$ are the bases of the histograms $H(\ell), H(b)$, and $H(r)$, respectively.
 4: $T = T \cup \{\mathsf{proj}_{e_1}(\ell), \mathsf{proj}_{e_2}(b), \mathsf{proj}_{e_3}(r)\}$
 5: $\mathcal{L}' = \{L_1, \dots, L_k, e'_1, e'_2, e'_3\}$
 6: for each point $q \in S$ do
 7:     for each line segment $L \in \mathcal{L}' \cap H$ where $H \in H(q)$ do
 8:         if $\mathsf{proj}_L(q) \in L$ then
 9:             $T = T \cup \mathsf{proj}_L(q)$;
10:             $E = E \cup \overline{q\ \mathsf{proj}_L(q)}$                  ▷ See Figure 4.9(ii)
11:         end if
12:     end for
13: end for
14: if the configuration of Type 1 holds then
```

```
15:     **for** *each point* $z \in S \cap H^r_{L_k}$ **do**
16:         $T = T \cup \mathsf{proj}_{e'_2}(z)$;
17:         $E = E \cup \overline{z\ \mathsf{proj}_{e'_2}(z)}$                          ▷ See Figure 4.9(iii)
18:     **end for**
19: **end if**
20: **if** the configuration of **Type 3** holds **then**
21:     **for** *each point* $w \in S \cap H^b_{L_k}$ **do**
22:         $T = T \cup \mathsf{proj}_{e'_3}(w)$;
23:         $E = E \cup \overline{w\ \mathsf{proj}_{e'_3}(w)}$                          ▷ See Figure 4.9(iv)
24:     **end for**
25: **end if**
26: **for** *each horizontal line segment* $L \in \mathscr{L}'$ **do**
27:     Let $L$ contain $k_1$ vertices $a_1, \dots, a_{k_1}$ of $T$, where $x(a_i) < x(a_{i+1})$ for $1 \leqslant i < k_1$
28:     **for** $1 \leqslant i \leqslant (k_1 - 1)$ **do**
29:         $E = E \cup \overline{a_i a_{i+1}}$                                       ▷ See Figure 4.9(v)
30:     **end for**
31: **end for**
32: **for** *each vertical line segment* $L \in \mathscr{L}'$ **do**
33:     Let $L$ contain $k_2$ vertices $b_1, \dots, b_{k_2}$ of $T$, where $y(b_i) > y(b_{i+1})$ for $1 \leqslant i < k_2$
34:     **for** $1 \leqslant i \leqslant (k_2 - 1)$ **do**
35:         $E = E \cup \overline{b_i b_{i+1}}$                                       ▷ See Figure 4.9(vii)
36:     **end for**
37: **end for**
38: Delete the following edges if they exist.
39: $(i)$ If there is a vertex (say, $u_1$) on the line segment $e'_1$ such that $u_1 \neq \ell$ and closest
    to $\mathsf{proj}_{e_1}(\ell)$, then we remove the edge $(u_1, \mathsf{proj}_{e_1}(\ell))$ for all such $u_1$.   ▷ See Figure
    4.9(vi)
40: $(ii)$ For **Types 1 or 4**, if there is a vertex (say, $u_2$) on the line segment $e'_2$ such
    that $u_2 \neq b$ and closest to $\mathsf{proj}_{e_2}(b)$, then we remove the edge $(u_2, \mathsf{proj}_{e_2}(b))$ for all
    such $u_2$.                                                                  ▷ See Figure 4.9(viii)
41: $(iii)$ For **Types 2 or 3**, if there is a vertex (say, $u_3$) on the line segment $e'_3$ such
    that $u_3 \neq r$ and closest to $\mathsf{proj}_{e_3}(r)$, then we remove the edge $(u_3, \mathsf{proj}_{e_3}(r))$ for all
    such $u_3$.                                                                  ▷ See Figure 4.9(ix)
42: For **Types 1 or 3**, if there is a vertex (say, $v$) on the line segment $L_k$ where
    $v \notin \{\mathsf{proj}_{e_3}(r), \mathsf{proj}_{e_2}(b)\}$ and $v$ is not a point on the boundary of $\mathscr{OCP}(S)$, then
    remove the single edge $(w, \mathsf{proj}_{L_k}(w))$, where $v = \mathsf{proj}_{L_k}(w)$.     ▷ See Figure 4.9(x)
    and Figure 4.9(xi)
```

```
43: For **Type 1**, if the line segment $e'_3$ contains a vertex other than $r$ and $\mathsf{proj}_{e_3}(r)$,
    then we remove the edge $(r, \mathsf{proj}_{e_3}(r))$.                ▷ See Figure 4.9(xii)
44: For **Type 3**, if the line segment $e'_2$ contains a vertex other than $b$ and $\mathsf{proj}_{e_2}(b)$,
    then we remove the edge $(b, \mathsf{proj}_{e_2}(b))$.                ▷ See Figure 4.9(xiii)
45: **return** $G = (S \cup T, E)$
```

Notice that for each point in S, Algorithm 2 adds at most three Steiner vertices in G. Specifically, $|V(G)| \leqslant 4n$ and $|E(G)| \leqslant 5n$. So both the number of vertices and the number of edges in G are $O(n)$. Now we prove the following lemma.

Lemma 4.4. *For the convex point set S, the graph G can be constructed in $O(n)$ time, assuming the points of S are given in sorted order along their convex hull.*

Proof. The construction of G from S consists of three phases. In phase 1, we construct $\mathscr{OCP}(\mathscr{S})$ from S. For each point $p \in S$, we add exactly one Steiner point and draw two edges. Now for a pair of points creating a Steiner point takes constant time. As $|S| = n$, phase 1 takes $O(n)$ time. In phase 2, we construct a histogram partition $\mathscr{H}(\mathscr{OCP}(S))$ of $\mathscr{OCP}(S)$. In Lemma 4.3, we show that this construction takes $O(n)$ time. In the third and final phase, we apply Algorithm 2 on $\mathscr{H}(\mathscr{OCP}(S))$ to construct our desired graph $G = (V,E) = (S \cup T, E)$. Now we show that Algorithm 2 runs in $O(n)$ time. In this algorithm, Steps 1-3 take linear time. In Steps 6-10, for each point $q \in S$, we perform orthogonal projections at most two times, i.e., we add at most two Steiner vertices and two edges. The points of S are given in sorted order along their convex hull. Also, we have an ordered set of k line segments $L_1, \dots, L_k$ with the ordering based on occurrence while constructing $\mathscr{H}(\mathscr{OCP}(S))$. Now, for any pair of points p_i and p_{i+1}, where $1 \leqslant i \leqslant n$ if the point p_i has an orthogonal projection on L_m for some m then p_{i+1} lies in the same or in a neighboring histogram and, consequently, p_{i+1} can not have an orthogonal projection onto any line segment in $\mathscr{L} \setminus \{L_{m-1}, L_m, L_{m+1}\}$. For example, if p_2 has an orthogonal projection on L_1, then p_3 can not have orthogonal projection onto any line segment in $\{L_3, L_4, \dots, L_k\}$. So we can do the Steps 6-10 by walking along the boundary of $\mathscr{OCP}(S)$ and by traversing $\mathscr{L}$ in its given order $L_1, ..., L_k$. So, overall it takes $O(n+k)$ time to perform all the projections. Steps 11-14 occur only when the configuration of Type 1 holds. We have to make one more projection for each point of $S \cap H^r_{L_k}$ to e'_2. Each projection can be computed in constant time. Therefore, Steps 11-14 take linear time. Similarly, Steps 15-18 take linear time. In Steps 19-26, we add edges to E by looking at each line segment of $\{L_1, \dots, L_k, e'_1, e'_2, e'_3\}$. Here the total number of projections is linear, and for a pair of projection, adding the edge into E takes constant time. So Steps 19-26 takes

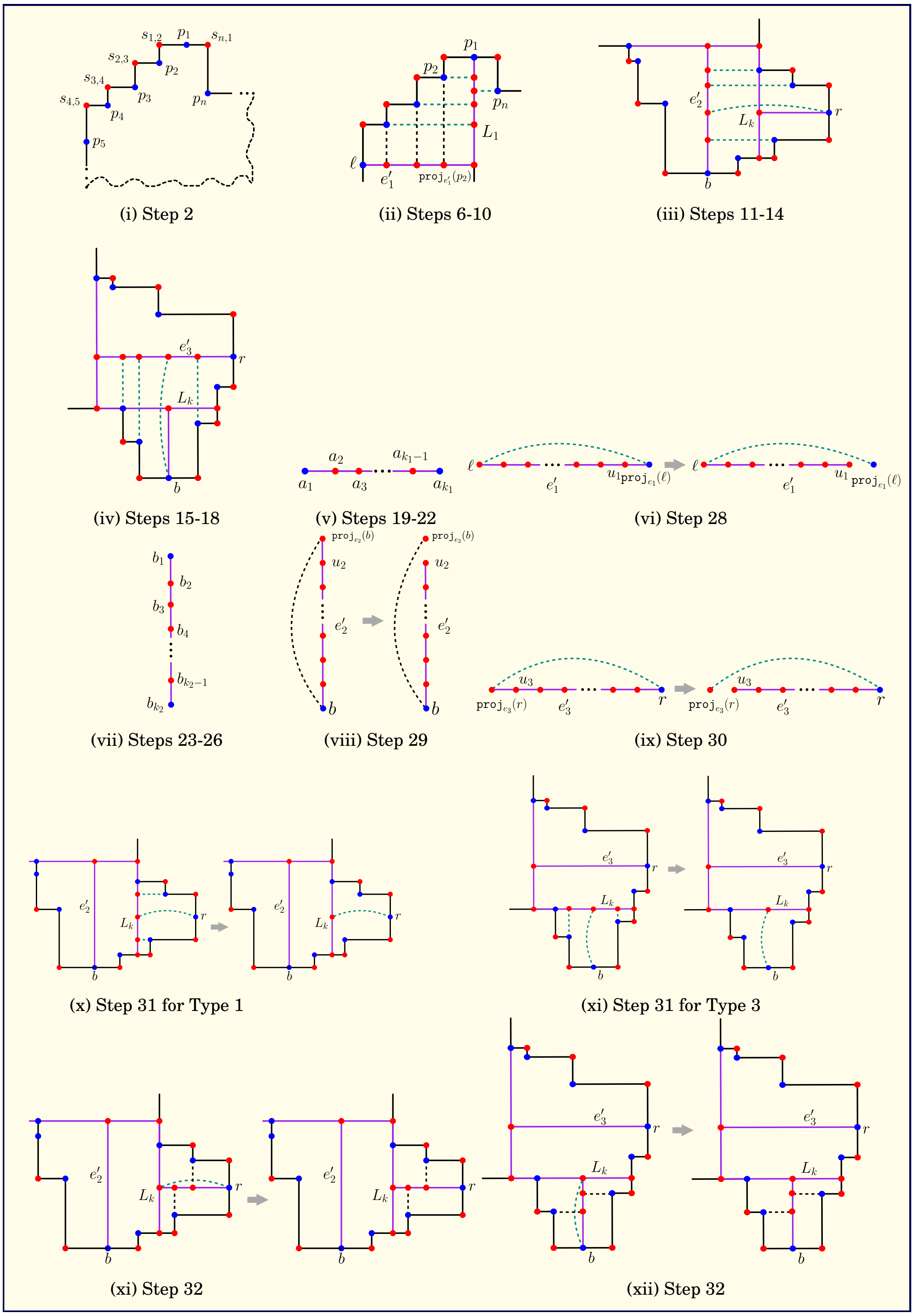


Figure 4.9: Illustration of the steps 2-33 in Algorithm 2. We use purple color for the line segments of the set $\mathscr{L}'$. Blue and red color points identify points from S and Steiner points, respectively. In (iii) and (iv), we only highlight the edges (dashed cyan) that we added in that step.

linear time. In Step 27-33, we delete some vertices and edges from $\{e'_1, e'_2, e'_3, L_k\}$. So the total time complexity is $O(n+k)$. As $k \leqslant n$, Algorithm 2 produces G in $O(n)$ time as claimed. ■

4.3.3 G is a Manhattan network

To show that G is a Manhattan network for the point set S, we have to prove that G contains a shortest L_1-path between every pair of points in S. In a simplified view, the proof is as follows. Recall that we partitioned the ortho-convex polygon of the input points into a sequence of histograms (which are rectilinear polygons that have 'bases' to which all the vertices can be projected orthogonally). Within each histogram, we connected the input points to the bases via vertical and horizontal edges. Since the bases of all histograms form a xy-monotone path, any pair of input points is connected by a shortest L_1-path (either on the boundary or via this xy-monotone path of the bases). In the following, we give a detailed proof that G is a Manhattan network.

Recall that $H(p)$ denotes the set of histograms containing $p \in S$ in $\mathscr{H}(\mathscr{OCP}(S))$ and $\mathscr{L} = \{L_1, \ldots, L_k\}$ denotes the set of k segments inserted in $\mathscr{OCP}(S)$ while constructing $\mathscr{H}(\mathscr{OCP}(S))$. The ortho-convex polygon $\mathscr{OCP}(S)$ consists of four xy-monotone chains $C_{rt}, C_{t\ell}, C_{\ell b}$, and C_{br}. Let p_i and p_j be two arbitrary points of S where $i, j \in \{1, ..., n\}$. We define $\pi_G(x,y)$ as the set of all shortest L_1-paths between a pair of vertices x and y in G. We use $\langle x,y \rangle$ to denote an arbitrary shortest L_1-path between x and y. For any two paths π_1 and π_2, where the two paths intersect exactly once and share a common endpoint, by $\pi_1 \rightsquigarrow \pi_2$ we denote the path that is obtained by concatenating the paths π_1 and π_2. Let $\pi_G(x,y)$ and $\pi_G(y,z)$ be two sets of paths in G where any pair of paths from different sets share exactly one point, which is nothing but y. Then by $\pi_G(x,y) \rightsquigarrow \pi_G(y,z)$, we mean the set of all possible paths that is obtained by concatenating the paths from $\pi_G(x,y)$ and $\pi_G(y,z)$, i.e., $\{\pi_1 \rightsquigarrow \pi_2 : \pi_1 \in \pi_G(x,y) \text{ and } \pi_2 \in \pi_G(x,y)\}$. Now we prove the following lemmas which together imply that G is a Manhattan network for S.

Lemma 4.5. *For each pair of points p_i and p_j of S in a same histogram, i.e., $H(p_i) \cap H(p_j) \neq \emptyset$, there exists a shortest L_1-path between them in G .*

Proof. If both p_i and p_j belong to the same xy-monotone chain then the claim holds as each xy-monotone chain of the ortho-convex polygon $\mathscr{OCP}(S)$ is a Manhattan network for the points it contains. Now we consider the case when p_i and p_j belong to different chains. Let H be the histogram that contains both p_i and p_j. For the special case, where both the points p_i and p_j lie on a same line segment in $\mathscr{L}$, there exists a shortest path between them through that line segment. For the rest of the cases, the proof can be divided into following four cases (we omit the symmetric cases where the roles of p_i and p_j are swapped). For an illustration of these cases see Figure 4.10.

Case A. $\boldsymbol{H \in H(\ell)}$: If $p_i \in L_1$ then the concatenated path $\langle p_i, \mathrm{proj}_{L_1}(p_j)\rangle \rightsquigarrow \langle \mathrm{proj}_{L_1}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$. In other cases, if $p_i \in C_{t\ell}$, $p_j \in C_{\ell b}$ then the concatenated path $\langle p_i, \mathrm{proj}_{e'_1}(p_i)\rangle \rightsquigarrow \pi_G(\mathrm{proj}_{e'_1}(p_i), \mathrm{proj}_{e'_1}(p_j)) \rightsquigarrow \langle \mathrm{proj}_{e'_1}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$.

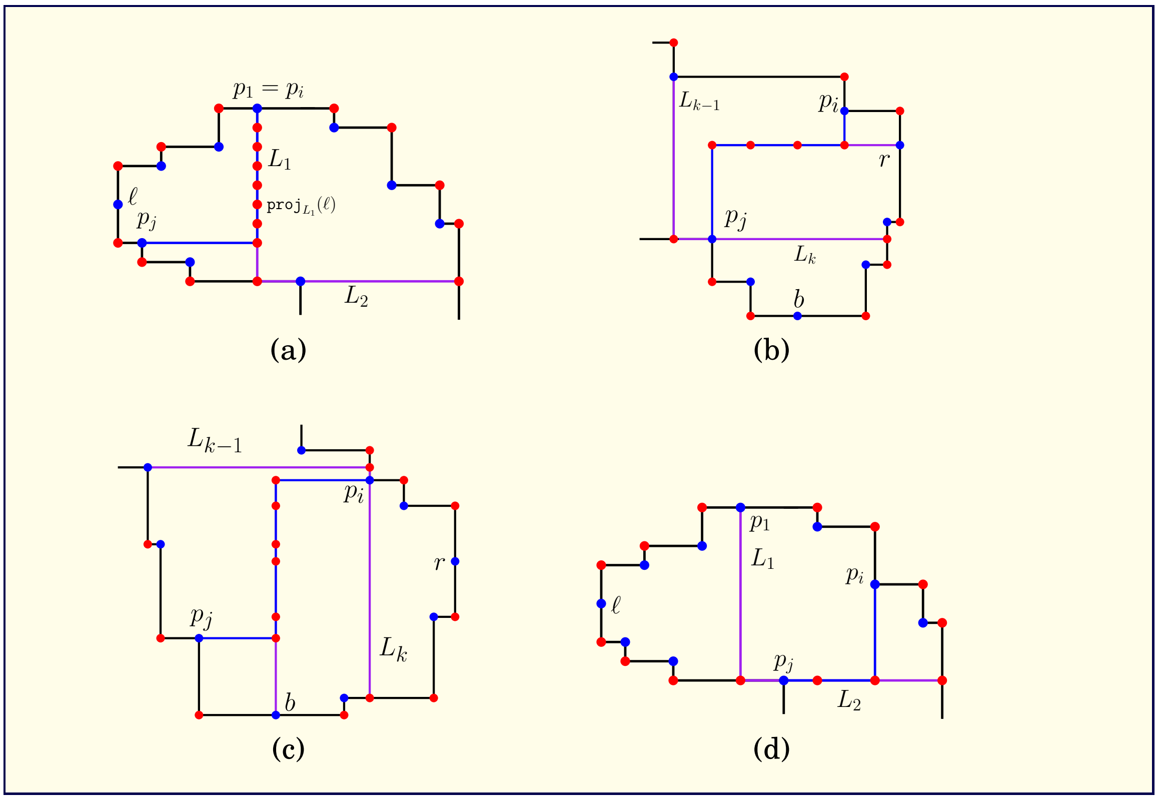


Figure 4.10: Examples of a shortest L_1-path (blue colored) between the pair of points p_i and p_j for the Case A (a), Case B (b), Case C (c), and Case D (d). For the clarity of figures, we do not depict some edges and vertices that are not part of the shortest L_1-path.

Case B. $\boldsymbol{H \in H(r)}$: For Type 1, if $p_i \in L_k$ then the concatenated path $\langle p_i, \mathrm{proj}_{L_k}(r)\rangle \rightsquigarrow \langle \mathrm{proj}_{L_k}(r), \mathrm{proj}_{e'_3}(p_j)\rangle \rightsquigarrow \langle \mathrm{proj}_{e'_3}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$. For Type 2 or 4, if one of p_i and p_j (say, p_i) belongs to L_k then the concatenated path $\langle p_i, \mathrm{proj}_{L_k}(p_j)\rangle$
$\rightsquigarrow \langle \mathrm{proj}_{L_k}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$. We are unable to do the same for Type 1 as in step 31 of Algorithm 2. We might have removed the edge $(\mathrm{proj}_{L_k}(p_j), p_j)$. For Type 3, if one of p_i and p_j (say, p_i) belongs to L_{k-1} then the concatenated

path $\langle p_i, \mathsf{proj}_{L_{k-1}}(p_j)\rangle \rightsquigarrow \langle \mathsf{proj}_{L_{k-1}}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$. In other cases, for Types 1, 2, or 3, if $p_i \in C_{rt}$ and $p_j \in C_{br} \cup C_{\ell b}$, then the concatenated path $\langle p_i, \mathsf{proj}_{e_3'}(p_i)\rangle \rightsquigarrow \pi_G(\mathsf{proj}_{e_3'}(p_i), \mathsf{proj}_{e_3'}(p_j)) \rightsquigarrow \langle \mathsf{proj}_{e_3'}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$. For Type 3, if $p_i \in C_{\ell b}$ and $p_j \in C_{br}$ then the concatenated path $\langle p_i, \mathsf{proj}_{L_k}(b)\rangle \rightsquigarrow \pi_G(\mathsf{proj}_{L_k}(b), p_j)$ belongs to $\pi_G(p_i,p_j)$. For Type 4, if $p_i \in C_{\ell b}$ and $p_j \in C_{br}$ then the concatenated path $\langle p_i, \mathsf{proj}_{e_2'}(p_i)\rangle \rightsquigarrow \pi_G(\mathsf{proj}_{e_2'}(p_i), \mathsf{proj}_{e_2'}(p_j)) \rightsquigarrow \langle \mathsf{proj}_{e_2'}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$.

Case C. $\boldsymbol{H \in H(b)}$: For Type 1, if $p_i \in L_{k-1}$ then the concatenated path $\langle p_i, \mathsf{proj}_{L_{k-1}}(p_j)\rangle \rightsquigarrow \langle \mathsf{proj}_{L_{k-1}}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$. For Type 3, if $p_i \in L_k$ then the concatenated path $\langle p_i, \mathsf{proj}_{L_k}(b)\rangle \rightsquigarrow \langle \mathsf{proj}_{L_k}(b), \mathsf{proj}_{e_2'}(p_j)\rangle \rightsquigarrow \langle \mathsf{proj}_{e_2'}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$. In other cases, for Types 1 or 3, if $p_i \in C_{\ell b}$ and $p_j \in C_{br} \cup C_{rt}$, then the concatenated path $\langle p_i, \mathsf{proj}_{e_2'}(p_i)\rangle \rightsquigarrow \pi_G(\mathsf{proj}_{e_2'}(p_i), \mathsf{proj}_{e_2'}(p_j)) \rightsquigarrow \langle \mathsf{proj}_{e_2'}(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$. For Type 1, if $p_i \in C_{rt}$ and $p_j \in C_{br}$ then the concatenated path $\langle p_i, \mathsf{proj}_{L_k}(r)\rangle \rightsquigarrow \pi_G(\mathsf{proj}_{L_k}(r), p_j)$ belongs to $\pi_G(p_i,p_j)$. For Types 2 or 4, as $H(b) \cap H(r) \neq \emptyset$, it is similar as subcase Case B.

Case D. $\boldsymbol{H \notin H(\ell) \cup H(b) \cup H(r)}$: Let these histograms contain two elements say L and L' of $\mathscr{L}$. Notice that in this case at least one of p_i and p_j already lies on L or L'. Assume that p_i lies on the line segment L. Then the concatenated path $\langle p_i, \mathsf{proj}_L(p_j)\rangle \rightsquigarrow \langle \mathsf{proj}_L(p_j), p_j\rangle$ belongs to $\pi_G(p_i,p_j)$. □

Lemma 4.6. *For any two points w and z in S where no histogram in $\mathscr{H}(\mathscr{OCP}(S))$ contains both w and z, i.e., $H(w) \cap H(z) = \emptyset$, there always exist line segments L and L' in $\mathscr{L}$ such that (i) $\mathsf{proj}_L(w) \in L$, $\mathsf{proj}_{L'}(z) \in L'$ and (ii) w and z lie on different sides of the lines through L and L'.*

Proof. Without loss of generality, $x(w) \leq x(z)$. First we observe that in our construction each histogram H (with the exception of the first and last one) is surrounded by two line segments L_j and L_{j+1} in $\mathscr{L}$ such that L_j is the base of the histogram H. Also for a point p in S, $|H(p)| > 1$ if and only if p belongs to some line segment in $\mathscr{L}$. If w belongs to the first histogram $H^l_{L_1}$, take $L = L_1$. If w belongs to some line segment L_i then $L = L_{i+1}$. Otherwise, let L_i be the base of the unique histogram in $H(w)$. Then we take $L = L_{i+1}$. Similarly, if z belongs to the last histogram $H^r_{L_k}$ or $H^b_{L_k}$, take $L' = L_k$. If z belongs to some line segment L_{j+1} then $L' = L_j$. Otherwise, let L_j be the base of the unique histogram in $H(z)$. Take $L' = L_j$. Observe that the line through L cuts the polygon into two halves with one half containing w and the other half containing all histograms with bases $L_{i+1}, \ldots, L_k$. Similarly, the line through

L' separates z from all histograms with bases $L_1, \ldots, L_{j-1}$ (assuming $j > 1$). Given $x(w) \leq x(z)$ and $H(w) \cap H(z) = \emptyset$, we have either $i = j = 1$ or $i + 1 \leq j$. In either case, w and z lie on different sides of the lines through L and L'. ■

Lemma 4.7. *For each pair of points w and z of S satisfying $H(w) \cap H(z) = \emptyset$, there exists a shortest L_1-path in G between them.*

Proof. Let w and z be a pair of points in S such that no histogram in $\mathscr{H}(\mathscr{OCP}(S))$ contains both w and z. First, we find line segments $L, L' \in \mathscr{L}$ such that (i) L can see w, L' can see z, and (ii) if we draw a line L^* that contains the line segment L (respectively, L') then w and z belong to opposite sides of L^*. By Lemma 4.6, both L and L' exist in $\mathscr{L}$ and it may happen that $L = L'$ (we say "w and z belong to opposite histograms").

Without loss of generality, we assume that $x(w) < x(z)$. If $L = L' \neq L_k$ (for example, $w = p_2$, $z = p_n$, $L = L' = L_1$) then $\pi_G(\mathsf{proj}_L(w), \mathsf{proj}_{L'}(z))$ is along the line segment L and thus the concatenated path $\langle w, \mathsf{proj}_L(w)\rangle \rightsquigarrow \pi_G(\mathsf{proj}_L(w), \mathsf{proj}_{L'}(z)) \rightsquigarrow \langle \mathsf{proj}_{L'}(z), z\rangle$ belongs to $\pi_G(w,z)$. Next we consider $L = L' = L_k$. For Type 2 or 3, the concatenated path $\langle w, \mathsf{proj}_{L_k}(w)\rangle \rightsquigarrow \pi_G(\mathsf{proj}_{L_k}(w), \mathsf{proj}_{L_k}(z)) \rightsquigarrow \langle \mathsf{proj}_{L_k}(z), z\rangle$ belongs to $\pi_G(w,z)$. For the Type 1 or 3, this proposed path might not exist as $\mathsf{proj}_{L_k}(w)$ or $\mathsf{proj}_{L_k}(z)$ might not exist because of Step 31 of the Algorithm 2. We do a case analysis for the Type 1. The case of Type 3 will follow similar kind of arguments (the difference is that in Type 3, L_k is horizontal instead of vertical as the Type 1). If $x(w) \leq x(b)$, then the concatenated path $\langle w, \mathsf{proj}_{e_2'}(w)\rangle \rightsquigarrow \pi_G(\mathsf{proj}_{e_2'}(w), \mathsf{proj}_{e_2'}(z)) \rightsquigarrow \langle \mathsf{proj}_{e_2'}(z), z\rangle$ belongs to $\pi_G(w,z)$. If $x(w) > x(b)$ and $y(z) \leq y(r)$, then w and z belong to the same xy-monotone chain; thus they are connected by a shortest L_1-path. Finally, if $x(w) > x(b)$ and $y(z) > y(r)$, then the concatenated path $\pi_G(w, \mathsf{proj}_{L_k}(r)) \rightsquigarrow \pi_G(\mathsf{proj}_{L_k}(r), \mathsf{proj}_{e_3'}(z)) \rightsquigarrow \langle \mathsf{proj}_{e_3'}(z), z\rangle$ belongs to $\pi_G(w,z)$.

Now we are left with the case when $L \neq L'$. For example, in Figure 4.12, considering l as w and r as z we find $L = L_1$ and $L' = L_k$. The rest of the proof can be divided into two cases. One is for vertical L and the other one is for horizontal L. Recall that $\{L_1, \ldots, L_k\}$ are the segments inserted in $\mathscr{OCP}(S)$ while constructing $\mathscr{H}(\mathscr{OCP}(S))$. The point set $\{q_i : 1 \leqslant i \leqslant k\}$ comes from the construction of $\mathscr{H}(\mathscr{OCP}(S))$. Also, we use $s_{i,i+1}$ to denote the vertex (Steiner point) that is in between p_i and p_{i+1} with respect to a counter-clockwise orientation along the boundary of $\mathscr{OCP}(S)$. Assuming $t = q_0$, L_i is the segment with endpoints q_{i-1} and q_i, where $1 \leqslant i \leqslant k$. We assume that L is vertical, the case in which it is horizontal being similar.
Let $L = L_m$ for some $m, 1 \leqslant m < k$. So $L_m = \overline{q_{m-1}q_m}$. If $w = q_m$ then the horizontal

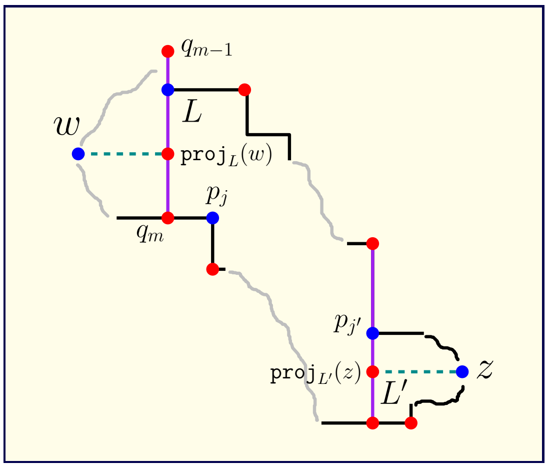


Figure 4.11: Shortest L_1-path between a pair of points from different histograms.

segment L_{m+1} also contains w. In this case we repeat the proof by considering L_{m+1} as L. Else if $w \neq q_m$, then by the construction of $\mathscr{H}(\mathscr{OCP}(S))$, q_m is not only a point on the boundary of $\mathscr{OCP}(S)$ but also there exists a point say p_j in S such that $q_m \in \overline{s_{j-1,j}p_j}$ (in case of collinearities, $q_m \in p_{j-1}p_j$). Since $L \neq L'$, z lies either below L_{m+1} or on L_{m+1}. In both cases, we find a Manhattan path $\langle p_j, z\rangle$ for the pair p_j, z: in the former case, by recursively applying our proof and taking L_{m+1} as the separating line, in the latter case by applying Lemma 5. Also recursively, we obtain a Manhattan path $\langle w, p_j\rangle$. Observe that the concatenation $\pi_G(w,p_j) \rightsquigarrow \pi_G(p_j,z)$ yields a Manhattan path in $\pi_G(w,z)$ and our claim follows. For an illustration, see Figure 4.11.

Lemma 4.5 and Lemma 4.7 together conclude that every pair of vertices of S is connected by a Manhattan path in G. Hence we conclude the following theorem.

Theorem 4.2. *G is a Manhattan network for the point set S.*

4.3.4 Planarity of G

In this section, we show that the graph $G = (V, E)$ is planar by providing a planar embedding. For an illustration, see Figure 4.12.

To show planarity, note that there might be crossings in the current drawing as each input point may contribute to horizontal and vertical edges inside the polygon. Therefore, we remove all vertical or all horizontal edges from every histogram and redraw them in the exterior face. In detail, we start with the first two histograms and draw their common base edge which is vertical in the exterior (not as a straight-line segment but as a curve that goes around the polygon). In the exterior, we also redraw

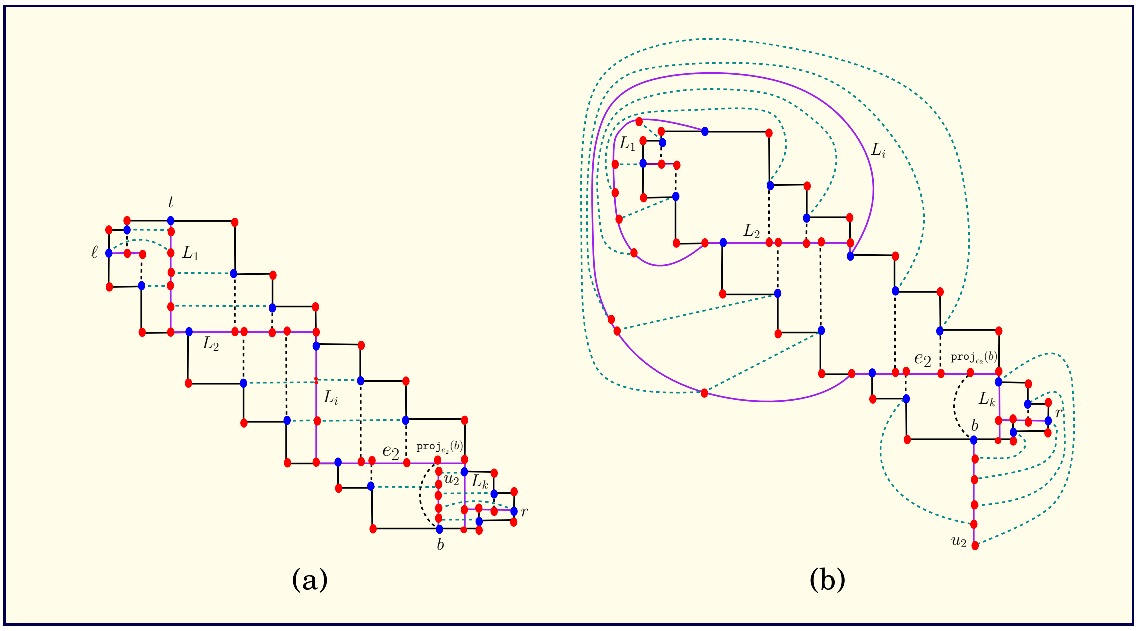


Figure 4.12: (a) The output G of Algorithm 2 for a point set in blue color. (b) A planar embedding of G.

in a crossing-free manner the formerly horizontal edges of the input points that are connected to this base edge. Now in their interior, all crossings are eliminated. In the same manner, we redraw all the remaining histogram pairs and obtain a planar drawing. Now we describe in detail how we obtain a planar embedding. The union of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is defined as the graph $(V_1 \cup V_2, E_1 \cup E_2)$ [W$^+$96]. We will make use of the following theorem.

Theorem 4.3. *[Gib85] A planar embedding of a graph can be transformed into another planar embedding such that any specified face becomes the exterior face.*

Relation to k-plane graphs [GHK$^+$15] A geometric graph $G = (V, E)$ is said to be a k-plane graph for some $k \in N$ if E can be partitioned into k disjoint subsets, $E = E_1 \uplus \cdots \uplus E_k$, such that $G_1 = (V, E_1), \ldots, G_k = (V, E_k)$ are all plane graphs, where $\uplus$ represents the disjoint union. For a finite general point set P in the plane, $\mathscr{G}_k(P)$ denotes the family of k-plane graphs with vertex set P. As per our construction, the graph that we construct to form a Manhattan network for the convex point set is a 2-plane graph because we can partition the edges into horizontal and vertical edge sets and each such set is crossing-free.

Theorem 4.4. *The graph G computed in Algorithm 2 is planar.*

In the following, we prove Theorem 4.4. We decompose G into two subgraphs H and K such that $G = H \cup K$. This decomposition depends on whether the line L_{k-1} is horizontal or vertical. For Types 1 and 2 of our histogram partition $\mathscr{H}(\mathscr{OCP}(S))$, the line segment L_{k-1} is horizontal, for Types 3 and 4, it is vertical. We will define the decomposition later and explain how the decomposition depends on the line L_{k-1}.

Let $V' = V(H) \cap V(K)$. We want to show that G is planar, i.e., there exists a planar embedding $\tilde{G}$ of G. If we can show that (i) there exist two planar embeddings, $\tilde{H}$ for H and $\tilde{K}$ for K, such that V' belongs to the exterior faces of both $\tilde{H}$ and $\tilde{K}$, (ii) there does not exist any edge $(u,v) \in E$ such that $u \in V(H) \setminus V', v \in V(K) \setminus V'$, and (iii) the vertices in V' appear in the same order in the two embeddings in the path on the exterior face they induce, then we can obtain a planar embedding $\tilde{G}$ of G by attaching the embeddings of $\tilde{H}$ with $\tilde{K}$, and this can be done in the following way:

- We transform the planar embedding $\tilde{H}$ of H to another planar embedding $\tilde{\tilde{H}}$ such that exterior face of $\tilde{H}$ becomes an interior face say $\mathscr{F}$ of $\tilde{\tilde{H}}$ (by Theorem 4.3).

- Now V' is contained in both $\mathscr{F}$ and the exterior face of $\tilde{K}$. Because of (ii) and (iii), we can attach the embedding $\tilde{K}$ inside $\mathscr{F}$ of $\tilde{\tilde{H}}$.

Given a planar drawing or embedding of a graph G in the plane, we define its boundary as follows. We can assume that the drawing of G is surrounded by a rectangular box B and let u be an imaginary vertex outside of B. We say a point p in G (either p is a vertex or belongs to an edge) belongs to its boundary if p can be joined with u by a curved line that does not intersect any edges of G. This boundary gives us an outer-face of the drawing. For an illustration of both boundary and outer-face of a drawing, see Figure 4.13.

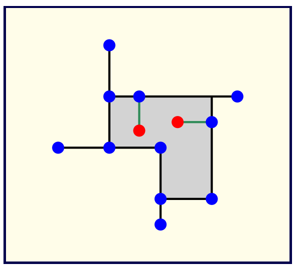


Figure 4.13: Black colored edges form the boundary of the drawing of a graph. All blue vertices are part of the boundary, whereas no red vertex belongs to the boundary. The non-shaded region is the outer-face of the graph.

Here we consider the drawing of G constructed by Algorithm 2 from which we later construct the planar embedding as another drawing. We use V^f to denote the vertices on the outer-boundary of G which are exactly the vertices along the boundary of $\mathscr{OCP}(\mathscr{S})$. We analyze each of the following two cases.

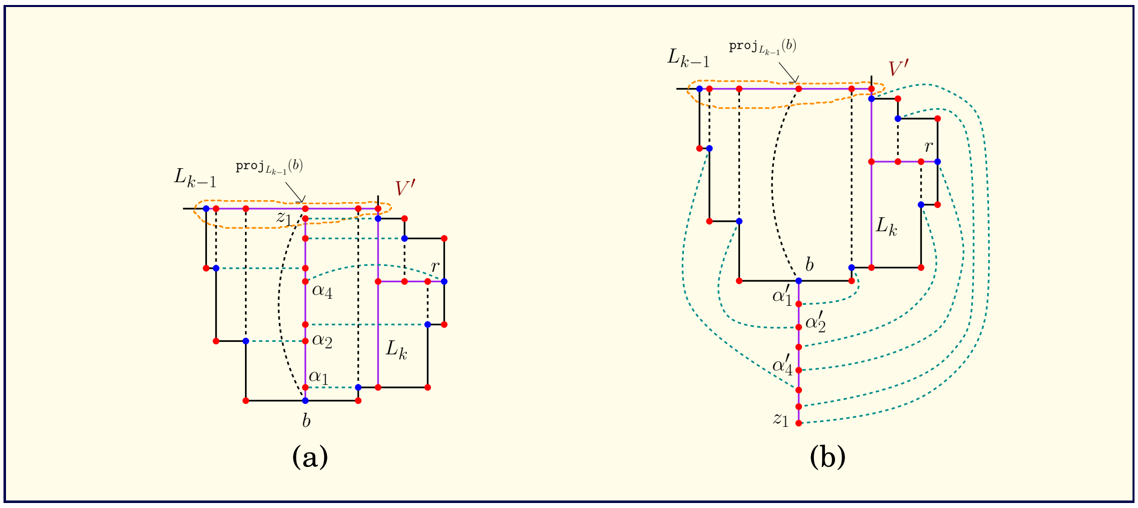


Figure 4.14: (a) The subgraph K of G for Type 1 with the exterior face containing V'. (b) A planar embedding of K. The edges of E_b are shown by dashed cyan segments.

Case 1 (L_{k-1} is horizontal) : H and K are the subgraphs of G induced by the vertices lying above and below, respectively, of the line segment L_{k-1}, i.e., $V(H) = \{v : v \in V,\ y(v) \geqslant y(L_{k-1})\}$. Similarly, $V(K) = \{v : v \in V,\ y(v) \leqslant y(L_{k-1})\}$. Recall that $V' = V(H) \cap V(K)$. We also consider the drawing of K (similarly, H) obtained from the drawing of G by removing the vertices lying strictly above (similarly, below) L_{k-1} (see Figure 4.14).

By our construction, it is easy to verify that there does not exist any edge $(u,v) \in E$ such that $u \in V(H) \setminus V', v \in V(K) \setminus V'$. Now we show that there exist two planar embeddings, $\tilde{H}$ for H and $\tilde{K}$ for K, such that V' belongs to the exterior faces of both $\tilde{H}$ and $\tilde{K}$, and also the vertices in V' appear in the same order in the two embeddings in the path on the exterior face they induce.

Lemma 4.8. *For the case that L_{k-1} is horizontal, the graph K has a planar embedding $\tilde{K}$ such that V' is contained in the exterior face of $\tilde{K}$. The vertices of V' induce a path on the exterior face and are in the same order as on L_{k-1} in the original drawing of G.*

Proof. Let us consider the drawing of K obtained from G. Let $V_K^f \subseteq V$ be the set of vertices in G along the boundary of K. So $V_K^f = (V^f \cap V(K)) \cup V'$. For Type 1, let E_b be the set of horizontal edges that have one endpoint in S and the other endpoint on the segment $e'_2 = \overline{b\ \mathsf{proj}_{L_{k-1}}(b)}$. Let z_1 be the top-most vertex on the segment e'_2 having $y(z_1) < y(\mathsf{proj}_{L_{k-1}}(b))$. In the planar embedding, all the Steiner points on the line segment $\overline{bz_1}$ will go to the exterior of the polygon along with their incident edges. Below we describe this procedure.

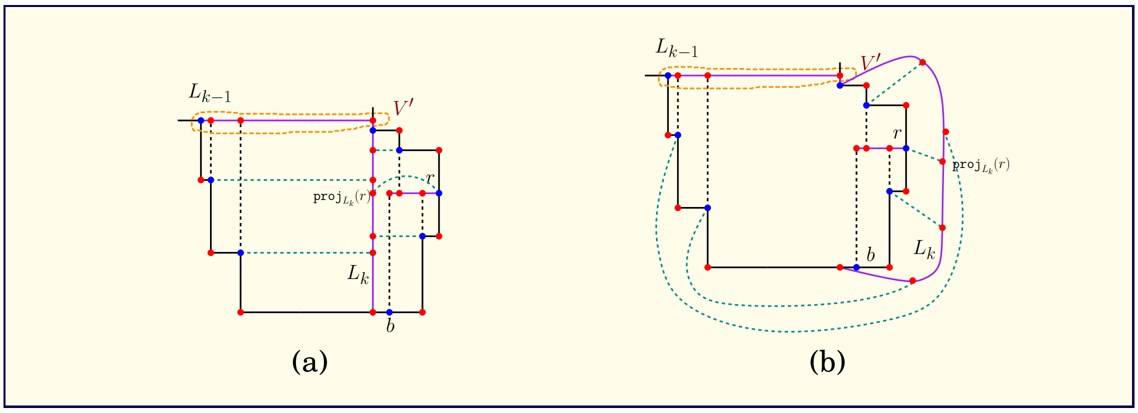


Figure 4.15: (a) The subgraph K of G for Type 2 with the exterior face containing V'. (b) A planar embedding of K. The edges of E_k are shown by dashed cyan segments.

Let the segment $\overline{bz_1}$ contain d Steiner points $\alpha_1, \ldots, \alpha_d$ $(= z_1)$ with $y(\alpha_1) \le \ldots \le y(\alpha_d)$. Also let b_{Left} be the set of those vertices in $S \cap C_{\ell b}$ that are incident to an edge in E_b, i.e., $b_{\mathsf{Left}} = \{v : v \in S \cap C_{\ell b} \text{ and } \mathsf{proj}_{e'_2}(v) \in e'_2\}$. Similarly, let b_{Right} be the set of those vertices in $S \cap (C_{\ell b} \cup C_{rt})$ that are incident to an edge in E_b, i.e., $b_{\mathsf{Right}} = \{v : v \in S \cap (C_{\ell b} \cup C_{rt}) \text{ and } \mathsf{proj}_{e'_2}(v) \in e'_2\}$. Let $\mathscr{F}(K)$ denote the outer-face of K. Now we define two non-intersecting regions $\mathscr{R}_1 = \{p : x(p) < x(b)\} \cap \mathscr{F}(K)$ and $\mathscr{R}_2 = \{p : x(p) > x(b)\} \cap \mathscr{F}(K)$ with respect to the line segment $\overline{bz_1}$. Next we do the following:

- We remove the segment $\overline{bz_1}$ and redraw it below b in the outer-face $\mathscr{F}(K)$.
- In this newly added segment, we add d vertices $\alpha'_1, \ldots, \alpha'_d$ satisfying $y(\alpha_1) \ge \ldots \ge y(\alpha_d)$ that correspond to the vertices $\alpha_1, \ldots, \alpha_d$, respectively.
- We draw the edges in E_b incident to b_{Left} in the region $\mathscr{R}_1$. Similarly, we draw the edges in E_b incident to b_{Right} in the region $\mathscr{R}_2$.

As $\mathscr{R}_1 \cap \mathscr{R}_2 = \emptyset$, no edge in $\mathscr{R}_1$ can intersect an edge in $\mathscr{R}_2$. It is easy to verify that it is possible to draw the edges in E_b incident to b_{Left} (similarly, b_{Right}) in the region $\mathscr{R}_1$ (similarly, $\mathscr{R}_1$) such that there is no pair of crossing edges. For an illustration see Figure 4.14.

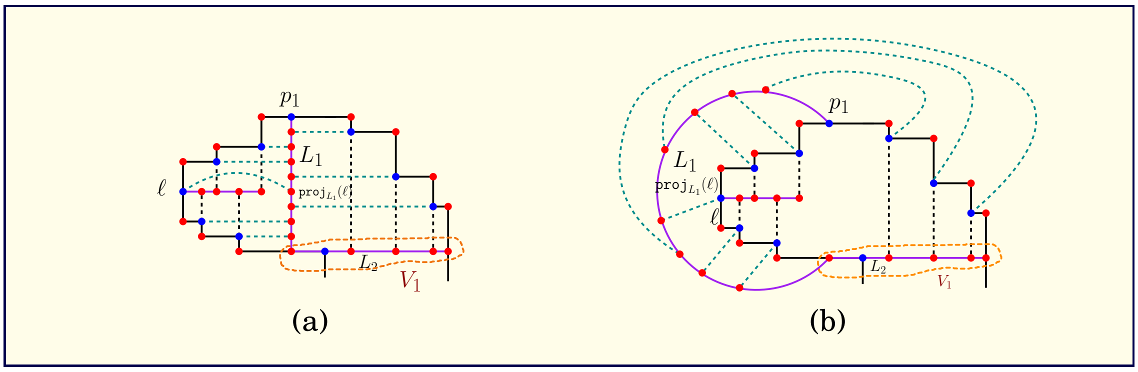


Figure 4.16: (a) The graph G_1, (b) A planar embedding of G_1 with the exterior face containing V_1 as a path (in the same order as in L_2).

For Type 2, let E_k be the set of horizontal edges that have at least one incident vertex on the line L_k. To obtain a planar embedding of K (similar as in the case of Type 1 for $\overline{bz_1}$), we first redraw L_k as L'_k (removing L_k and its vertices) in the outer-face, then place the vertices of L_k in the same order on L'_k and finally draw the edges of E_k without crossings and in such a way that L_{k-1} remains in the outer-face (because it is possible to make the redrawing such that L_{k-1} is not in the outer-face anymore). For an illustration see Figure 4.14. So for both cases of Type 1 and Type 2, V' remains in the exterior face of the planar embedding. Hence, we obtain a planar embedding $\tilde{K}$ of K such that V' is contained in the exterior face of $\tilde{K}$. By our construction, it is easy to verify that V' appears as a path in the same order in K and $\tilde{K}$. ■

Lemma 4.9. *For the case that L_{k-1} is horizontal, the graph H has a planar embedding $\tilde{H}$ such that V' is contained in the exterior face of $\tilde{H}$. The vertices of V' induce a path on the exterior face and are in the same order as on L_{k-1} in the original drawing of G.*

Proof. We prove this by weak induction. As L_{k-1} is horizontal, $(k-1)$ must be even. Let $(k-1) = 2m$ for some $m \in \mathbb{N}$. Let V_i consist of all the vertices in G on the line segment L_{2i} and G_i be the subgraph induced by the vertices lying on or above the line segment L_{2i}, where $i \leqslant m$. So $G_m = H$. By induction, we prove that G_m is planar and it has a planar embedding $\tilde{H}$ such that V' is contained in the exterior face of $\tilde{H}$. Let $P(i)$ be the following statement: G_i is planar and it has a planar embedding $\tilde{G}_i$ such that V_i is contained in the exterior face of $\tilde{G}_i$. Now we need to show that $P(m)$ is true. We prove the statement by induction.

Base Case: $P(1)$ is true: Let us consider the drawing of G_1 obtained from G. We divide the edges of G_1 into three sets E_{11},E_{12}, and E_{13}. The edge set E_{11} consists of all the edges in G_1 that are along the boundary of G_1. The edge set E_{12} consists of all the edges in G_1 that have at least one endpoint on the segment L_1. We define the edge set E_{13} as $E_{13} = E(G_1) \setminus (E_{11} \cup E_{12})$. Let G_{11} be the subgraph of G_1 consisting of the edges $E_{11} \cup E_{12}$, and G_{12} be the subgraph of G_1 consisting of the edges $E_{11} \cup E_{13}$. So $G_1 = G_{11} \cup G_{12}$, where both G_{11} and G_{12} are plane graphs. Now G_{11} (simultaneously, G_{12}) itself is a planar embedding obtained from the drawing of G. In G_{11} there exists an interior face containing V_1. Let V_{12}^f be the set of vertices in the exterior face of G_{12} (and G_{11}, note that both the graphs G_{11} and G_{12} have the same set of vertices in their exterior face). By Theorem 4.3, we can transform the planar embedding G_{12} into another planar embedding $\tilde{G}_{12}$ such that there exists an interior face, say f_1, that contains V_{12}^f. As V_{12}^f is also the set of vertices in the exterior face of G_{11}, and the vertices in V_{12}^f appear in the same order in the two embeddings in the path on the exterior face they induce, we can attach G_{11} to G_{12} in f_1 and obtain a planar embedding G_1' of G_1. In G_1' there exists an interior face containing V_1. Applying Theorem 4.3, we get a planar embedding $\tilde{G}_1$ of G_1 such that V_1 is contained in the exterior face of $\tilde{G}_1$. We illustrate this step in Figure 4.16.

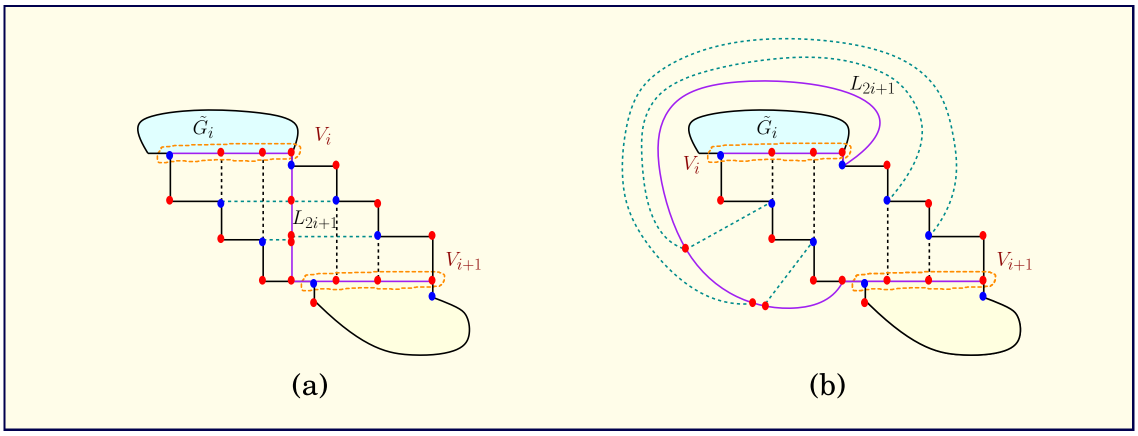


Figure 4.17: (a) The graph G where G_i has the planar embedding $\tilde{G}_i$ where the exterior face of $\tilde{G}_i$ contains V_i, (b) The graph G where G_{i+1} has the planar embedding $\tilde{G}_{i+1}$ where the exterior face of $\tilde{G}_{i+1}$ contains V_{i+1}.

Inductive Case: $P(i)$ is true $\Rightarrow P(i+1)$ is true: Assume that $P(i)$ is true, i.e., G_i has a planar embedding $\tilde{G}_i$ such that V_i is contained in the exterior face of $\tilde{G}_i$ (see Figure 4.17).

Let H_1 be the subgraph of G_{i+1} induced by the vertices lying on or below the line containing L_{2i}. Now $V_i = V(\tilde{G}_i) \cap V(H_1)$. Recall that V_i is contained in the exterior face of $\tilde{G}_i$, and also the vertices in V_i appear as a path in the same order in $\tilde{G}_i$ and H_1 (that is, in the same order as in L_{2i} in G_i). As $G_{i+1} = G_i \cup H_1$ and since there are no edges between $V(G_i) \setminus V_i$ and $V(H_i) \setminus V_i$ so in G_{i+1}, we can replace G_i by its planar embedding $\tilde{G}_i$ within the embedding of G_{i+1}. Here $G_{i+1} = \tilde{G}_i \cup H_1$ with the drawing obtained from G. Now we divide the edges of G_{i+1} into three sets E_1, E_2, E_3. The edge set E_1 consists of all the edges in G_{i+1} that are along the boundary of H_1. The edge set E_2 consists of all the edges in H_1 that have at least one endpoint on the segment L_{2i+1}. $E_3 = E(H_1) \setminus (E_1 \cup E_2)$. Let I_1 be the subgraph of G_{i+1} consisting of the edges $E_1 \cup E_2 \cup E(\tilde{G}_i)$, and I_2 be the subgraph of G_{i+1} consisting of the edges $E_1 \cup E_3 \cup E(\tilde{G}_i)$. So $G_{i+1} = I_1 \cup I_2$, where both I_1 and I_2 are plane graphs. Now I_1 (simultaneously, I_2) itself is a planar embedding obtained from the drawing of G. In I_1 there exists an interior face containing V_{i+1}. Let $V^f_{(i+1)2}$ be the set of vertices in the exterior face of I_2. By Theorem 4.3, we can transform the planar embedding I_2 into another planar embedding such that there exists an interior face, say f, that contains $V^f_{(i+1)2}$. As $V^f_{(i+1)2}$ is also the set of vertices in the exterior face of I_1 and the vertices in $V^f_{(i+1)2}$ appear in the same order in the two embeddings in the path on the exterior face they induce, we can attach I_1 to I_2 in f and obtain a planar embedding G'_{i+1} of G_{i+1}. In G'_{i+1} there exists an interior face containing V_{i+1}. Applying Theorem 4.3, we get our desired planar embedding $\tilde{G}_{(i+1)}$ of G_{i+1} such that V_{i+1} is contained in the exterior face of $\tilde{G}_{(i+1)}$.

Hence by the induction hypothesis, $P(m)$ is true, i.e., G_m is planar and it has a planar embedding $\tilde{G}_m$ such that V_m is contained in the exterior face of $\tilde{G}_m$. Here the set V_m consists of all the vertices on the line L_{2m}. Now $2m = k$ implies that $V_m = V'$. Also $G_m = H$. So H has a planar embedding $\tilde{H}$ $(= \tilde{G}_m)$ such that V' is contained in the exterior face of $\tilde{H}$. By our construction, it is easy to verify that V' appears as a path in the same order in H and $\tilde{H}$ ■

Case 2 (L_{k-1} is vertical):

In this case, we use the same idea as in case 1 to prove that G is planar. When L_{k-1} is vertical, we partition G into H and K as follows: H and K are the subgraphs of G induced by the vertices lying to the left and right, respectively of the line L_{k-1}. Both H and K must include the vertices on L_{k-1}. Now in case 1, when we do a planar embedding for H, we leave the horizontal L_i's inside the polygon $\mathscr{OCP}(\mathscr{S})$ and move the vertical segments $L_1, L_3, \ldots, L_{k-2}$ outside the polygon. Also, we move one more vertical segment (depends on the type of the configuration) outside the polygon but that is the part of planar embedding for K. Here in case 2, we can get a planar embedding for H by moving the horizontal segments $L_2, L_4, \ldots, L_{k-2}$ outside the polygon and leaving the vertical L_i's inside. We will not describe this (planarity for H) as we can do it in a similar way as case 1. Hence the following lemma.

Lemma 4.10. *For the case that L_{k-1} is vertical, the graph H has a planar embedding $\tilde{H}$ such that V' is contained in the exterior face of $\tilde{H}$. The vertices of V' induce a path on the exterior face and are in the same order as on L_{k-1} in the original drawing of G.*

Also, by our construction, it is easy to verify that V' appears as a path in the same order in H and $\tilde{H}$. Now we show planarity for K where we move one more horizontal segment (either L_k or $\overline{z_3 r}$, depends on the type of the configuration) outside the polygon.

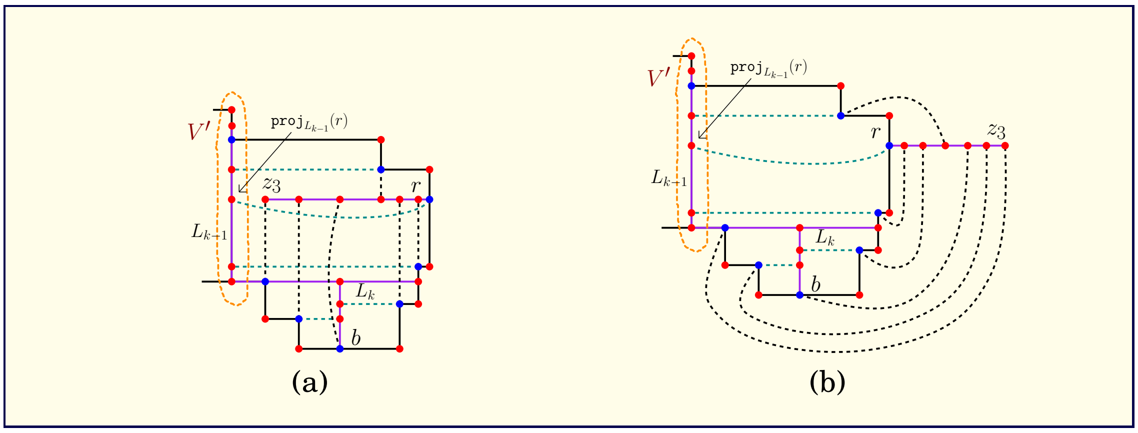


Figure 4.18: (a) The subgraph K of G for Type 3 with the exterior face containing V', (b) a planar embedding of K. The edges of E_r are shown by dashed black segments.

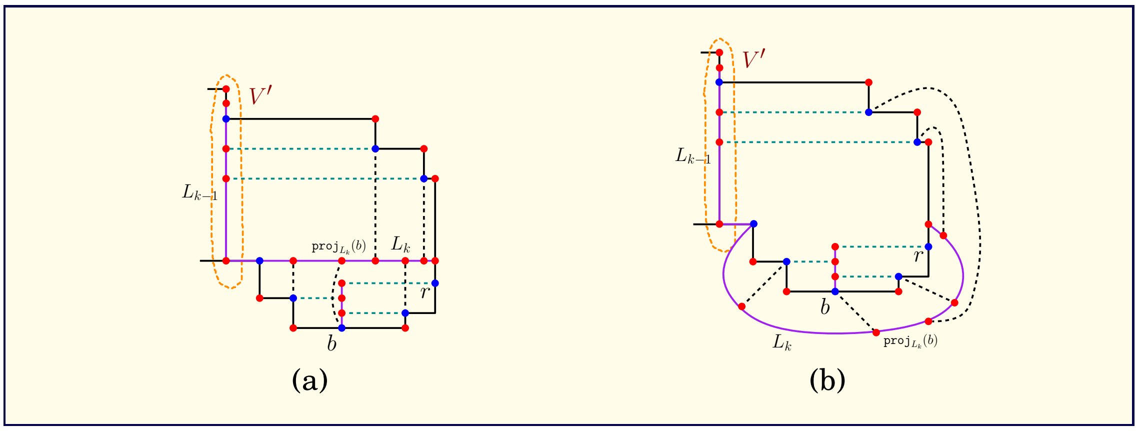


Figure 4.19: (a) The subgraph K of G for Type 4 with the exterior face containing V', (b) a planar embedding of K. The edges of E_k are shown by dashed black segments.

Lemma 4.11. *For the case that L_{k-1} is vertical, the graph K has a planar embedding $\tilde{K}$ such that V' is contained in the exterior face of $\tilde{K}$. The vertices of V' induce a path on the exterior face and are in the same order as on L_{k-1} in the original drawing of G.*

Proof. Let us consider the drawing of K obtained from G. Let $V_K^f \subseteq V$ be the set of vertices in G along the boundary of K. So $V_K^f = (V^f \cap V(K)) \cup V'$. For Type 3, let E_r be the set of vertical edges that have at least one adjacent vertex on the line $e_3' = \overline{r\ \mathrm{proj}_{L_{k-1}}(r)}$. To obtain a planar embedding of K we draw the edges of E_r in the outer-face of K. Let z_3 be the left-most vertex on the segment e_3' having $x(z_3) > x(\mathrm{proj}_{L_{k-1}}(r))$. In the planar embedding, all the Steiner points on the line segment $\overline{rz_3}$ will go to the exterior of the polygon along with its incident edges (see Figure 4.18). For Type 4, let E_k be the set of vertical edges that have at least one adjacent vertex on the line L_k. In this case, we draw the edges E_k in the exterior faces of K in such a way that we obtain a planar embedding of K. In the embedding, all Steiner points on the line segment L_k will go to the polygon exterior along with its adjacent edges (see Figure 4.19). For both types, the procedure for doing the embedding is the same as described in Lemma 4.8. In this planar embedding V' still remains in the exterior face. Hence, we get a planar embedding $\tilde{K}$ of K such that V' is contained in the exterior face of $\tilde{K}$ Also, by our construction, it is easy to verify that V' appears as a path in the same order in K and $\tilde{K}$. ■

This completes the proof of Theorem 4.4.

Summary

In this chapter, we construct a planar Manhattan network G for a given convex point set S of size n in linear time, where G contains $O(n)$ Steiner points. Our construction for convex point sets is optimal with respect to the number of Steiner points as there exist convex points sets requiring $\Omega(n)$ Steiner points for any planar Manhattan networks; for instance $S = \{(1,1),...(n,n)\}$. As a corollary of our construction, for a convex point set, we obtain a $\sqrt{2}$ (≈ 1.41) planar spanner in L_2-norm using $O(n)$ Steiner points.

CHAPTER 5

MAXIMUM BIPARTITE SUBGRAPHS

Contents

Related Publication:

1. Satyabrata Jana, Anil Maheshwari, Saeed Mehrabi, and Sasanka Roy. "Maximum bipartite subgraph of geometric intersection graphs." In *International Workshop on Algorithms and Computation*, pp. 158-169. Springer, 2020.

5.1 Introduction

In this chapter, we study the following problem on geometric objects.

Given a set S of n geometric objects in the plane

➩ compute a maximum-size subset $S' \subseteq S$ such that the intersection graph induced by the objects in S' is bipartite.

We refer to this problem as the Maximum Bipartite Subgraph (MBS) problem. The MBS problem is closely related to the Odd Cycle Transversal (OCT) problem: given a graph G, the objective of the OCT problem is to compute a minimum-cardinality subset of $S \subseteq V(G)$ such that the intersection of S and the vertices of every odd cycle of the graph is non-empty. Notice that MBS and OCT are equivalent for the class of graphs on which OCT is polynomial-time solvable: an exact solution S for OCT gives $V(G) \setminus S$ as an exact solution for MBS within the same time bound. However, an α-approximation algorithm for OCT might not provide any information on the approximability of MBS on the same classes of graphs.

We also study a simpler variant of MBS, called the Maximum Triangle-free Subgraph (MTFS) problem. Let S be a set of n geometric objects in the plane. Then, the objective of the MTFS problem is to compute a maximum-size subset $S' \subseteq S$ such that the intersection graph induced by the objects in S' is triangle-free (as opposed to being bipartite).

The MBS problem is also closely related to the Maximum Independent Set (MIS) problem. Given a graph G, the objective of MIS is to compute a maximum-cardinality subset of vertices such that no two of them are adjacent. Observe that any feasible solution for MIS is also a feasible solution for MBS and, moreover, a feasible solution $S \subseteq V(G)$ for the MBS problem provides a feasible solution of size at least $|S|/2$ for the MIS problem. This gives us the following observation.

Observation 5.1. *An α-approximation algorithm for* MIS *on a class of graphs is a 2α-approximation for* MBS *on the same class with the same time bound.*

In particular, the PTASes for MIS on unit disks and unit squares [HM85b] imply polynomial-time $(2+\epsilon)$-approximation algorithms for MBS on unit disks and unit squares. Also, 2-factor for MIS on unit disks [NPR17] , which runs in $O(n^2)$ time gives an $O(n^2)$ time 4-factor algorithm for the MBS problem on unit disks. Moreover, A $O(\log\log n)$-approximation [CC09] (or, $O(\log \mathsf{OPT})$-approximation [BCK+19]) algorithm is known for MIS on rectangles. As a corollary of Observation 5.1, we can say the following.

Corollary 5.1. *A $f(n)$-time exact algorithm for* MIS *on a class of graphs produces 2-approximation algorithms for* MBS *on the same class with the same time bound.*

There are many graph classes like perfect graphs [KN13], P_6-free graphs, [GKPP19, DFJ+20], line graphs [LM08, DFJ+20] for which MBS is NP-Hard but MIS is polytime solvable; hence it gives us 2-approximation for MBS problem in same polytime.

5.1.1 Our contributions

We have considered the MBS problem on several geometric objects. We show that,

- If the input is a set of n circular-arcs (arcs of the same circle) then we can compute an almost optimal solution $O(n^2)$ time.
- Hardness on the classes of geometric graphs for which the MIS problem is NP-Hard.
- If the input is a set of n unit disks intersecting a horizontal line where all the centers of the disks lie on one side of the line then the problem can be solved in $O(n^2)$-time.
- 2-factor approximate solution when the input is a set of n unit disks intersecting a common line.
- PTAS, $O(\log n)$ and 2-factor approximate solutions when the input is a set of n arbitrary unit disks.
- PTAS when the input is a set of n unit squares.
- 2-factor approximate solution when the input is a set of n unit-height rectangles.
- Hardness of MTFS problem on the intersection graph of axis-parallel rectangles in the plane.

5.2 Algorithmic Result on Circular-arc Graphs

A circular-arc graph is the intersection graph of a set of arcs on a circle. That is, every vertex is represented by an arc, and there is an edge between two vertices if and only if the corresponding arcs intersect. Observe that interval graphs are a proper subclass of circular-arc graphs. Let $G = (V, E)$ be a circular-arc graph with $|V| = n$. Narasimhan [Nar89] showed that this problem can be solved optimally with running time $O(n^3)$ assuming that a corresponding family of arcs is given. Now we present a quadratic time almost optimal solution for the MBS problem on circular-arc graphs. We say a solution S to a maximization problem is *almost optimal* with respect to its optimum solution X if $|X| \leq |S| + 1$. Given a circular graph, its intersection model or circular-arc representation can be obtained using Tucker's $O(n^3)$ algorithm [Tuc80]. For the rest of this section we assume that a geometric representation of G (i.e., a set of n arcs on a circle $\mathscr{C}$) is given as part of the input, First, we prove the following lemmas.

Lemma 5.1. *If G is triangle-free, then it can have at most one cycle.*

Proof. Suppose for the sake of contradiction that G has more than one cycle. Let A_1 and A_2 be two cycles of G. Now, since G is a triangle-free circular-arc graph, the corresponding arcs of the vertices of any cycle in G together cover the circle $\mathscr{C}$. So, there must exist three distinct vertices $v \in A_1, u \in A_1$ and $w \in A_2$ such that v, u, w are pairwise adjacent. Which is a contradiction to the fact that G is triangle-free. ■

Lemma 5.2. *If B and T are optimal solutions for the* MBS *and* MTFS *problems on G, respectively, then $|T| - 1 \leq |B| \leq |T|$.*

Proof. Since a bipartite subgraph contains no triangle, $|B| \leq |T|$. Now, if $G[T]$ (i.e., the subgraph of G induced by T) is odd-cycle free, then it induces a bipartite subgraph. Otherwise, $G[T]$ can have at most one cycle by Lemma 5.1. If this cycle is odd, then by removing any single vertex form the cycle, we obtain a bipartite subgraph of G with size at least $|T| - 1$. ■

Since $G[T]$ contains at most one cycle, the following lemma trivially holds.

Lemma 5.3. *If H is a maximum-size induced forest in G, then $|V(H)| \geq |T| - 1$.*

By the above lemmas, our goal now is to find a maximum acyclic subgraph H of G. So instead of solving MBS problem, we actually solve *maximum induced forest* problem (maximum-cardinality subset of vertices such that the subgraph induced by them is acyclic) in circular-arc graphs. Notice that there must be a clique K ($|K| \geq 1$) in G that is not in H. Now, for each arc u in the circular-arc representation of G, let $l(u)$ and $r(u)$ denote the two endpoints of u in the clockwise order of the endpoints. Then, we consider two vertex sets $S_u^1 = \{w : w \in V, l(u) \notin (l(w), r(w)]\}$ and $S_u^2 = \{z : z \in V, r(u) \notin [l(z), r(z))\}$. Both S_u^1 and S_u^2 are interval graphs. Since there are n vertices in G, we compute $2n$ interval graphs in total. Then, for each of these interval graphs, we apply the Narasimhan's $O(n)$ algorithm [Nar89] to compute an optimal solution for MBS, and will return the one with maximum size as the final solution. So the total time to find H is $O(n^2)$; hence the following theorems.

Theorem 5.1. *Given a circular-arc graph G with its corresponding family of arcs, in quadratic time we can solve the maximum acyclic subgraph problem on G.*

Theorem 5.2. *Given a circular-arc graph G with its corresponding family of n arcs, in $O(n^2)$ time we can compute an induced bipartite subgraph H of G such that $|V(H)| \geq |\mathsf{MBS}(G)| - 1$, where $\mathsf{MBS}(G)$ denotes an optimal solution of MBS problem in G.*

5.3 Hardness

In this section, we show that the MBS problem is NP-Complete on the classes of geometric intersection graphs for which MIS is NP-Complete. The MIS problem is known to be NP-Complete on a wide range of geometric intersection graphs, even restricted to unit disks and unit squares [CCJ90], 1-string graphs [KN90], and B_1-VPG graphs [LMS15]. Let $G = (V, E)$ be an intersection graph induced by a set S of n geometric objects in the plane. We construct a new graph G' from the disjoint union of two copies of G by adding edges as follows. For each vertex in V, we add an edge from each vertex in one copy of G to the corresponding vertex in the other copy. For each edge $(u, v) \in E$, we add four edges $(u, v), (u', v'), (u, v')$, and (v, u') to G', where u' and v' are the corresponding vertices of u and v, respectively in the other copy. Graph G' is the intersection graph of $2n$ geometric objects S, where each object has occurred twice in the same position. See Figure 5.1 for an illustration.

Clearly, the number of vertices and edges in G' are polynomial in the number of vertices of G; hence, the construction can be done in polynomial time.

Lemma 5.4. *G has an independent set of size at least k if and only if G' has a bipartite subgraph of size at least $2k$.*

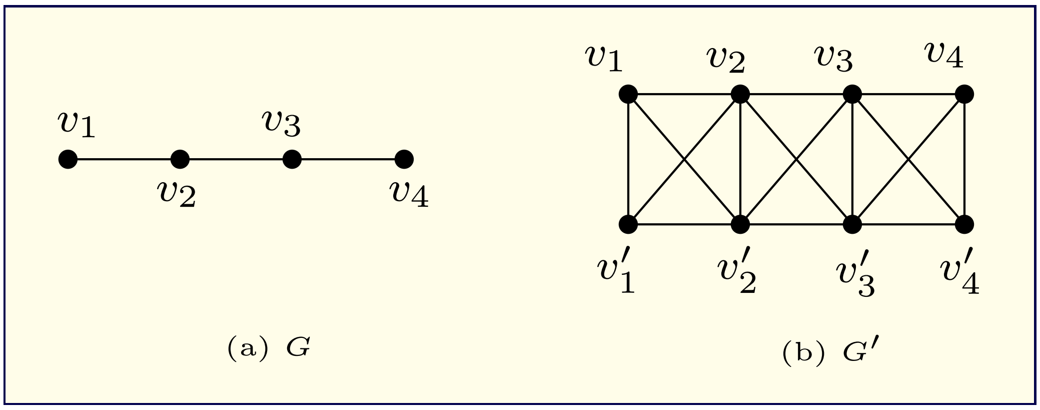


Figure 5.1: (a) A graph G, (b) The graph G' constructed from G.

Proof. Let U be an independent set of G with $|U| \geq k$. Let H be the subgraph of G' induced by U along with all the corresponding vertices of U in the other copy. Then, H is a bipartite subgraph with size at least $2k$. Conversely, let G' has a bipartite subgraph of size at least $2k$. Now for a pair of integers i and j, if any pair of vertices from $\{v_i, v_j, v_i', v_j'\}$ is non-adjacent in G' then v_i must be non-adjacent to v_j in G. So if G' has an independent set of size at least $2k$, then G must have an independent set of size at least k. ■

By Lemma 5.4, we have the following theorem.

Theorem 5.3. *The* MBS *problem is* NP-Complete *on the classes of geometric intersection graphs for which* MIS *is* NP-Complete.

Recall that, MBS is NP-Hard but MIS is polynomial time solvable for many graph classes, e.g., perfect graphs [KN13], P_6-free graphs, [GKPP19, DFJ+20], line graphs [LM08, DFJ+20].

Remark. By the definition of parameterized reduction we have the following result.

Corollary 5.2. *The* MBS *problem is* W[1]*-complete on the classes of geometric intersection graphs for which* MIS *is* W[1]*-complete.*

Marx [Mar05, Mar06] proved that MIS is W[1]-complete on unit squares, unit disks, and even unit line segments. As such, by Corollary 5.2, the MBS problem is W[1]-complete on all these geometric intersection graphs.

5.4 MBS on Unit Disks intersecting a Line

In this section, we first look at the MBS problem where the inputs are a set of unit disks that intersect a horizontal line and all centers lie on one side of the line. Next, we give a 2-approximation algorithm for the MBS problem on the unit disks intersecting a horizontal line (i.e., without requiring the centers to lie on the same side of the line). Our 2-approximation algorithm is based on (i) partitioning the input disks into two sets, depending on which side of the line their centers lie, and (ii) solving the MIS problems independently in these two sets. Then we will discuss how to obtain the factor-2 approximation.

5.4.1 Unit disks intersecting a line and centers lie on one side

Here, we are given n unit disks $\mathscr{D} = \{D_1, \cdots, D_n\}$ intersecting a straight line L, where the center of every disk in $\mathscr{D}$ lies on one side of L. We assume w.l.o.g. that all the disks intersect the x-axis and all the centers have non-negative y-coordinates. So, $\mathscr{D}$ is a set of n unit disks in the plane that are intersected by the X-axis and all the centers of the disks have non-negative y-coordinate. We use following notations in this section. $G_{\mathscr{D}}$ denotes the intersection graph of disks in $\mathscr{D}$. That means each disk in $\mathscr{D}$ corresponds to a vertex in $G_{\mathscr{D}}$ and there is an edge between vertices if the corresponding disks intersect. In the rest of this section, we use "disks" and "vertices" in $G_{\mathscr{D}}$ interchangeably. We also use $x(p)$ and $y(p)$ to denote, respectively, the x- and y-coordinates of a point p. Let $d(p,q)$ denote the Euclidean distance between two points p and q. Let c_i denote the center of the disk D_i for all $1 \le i \le n$. W.l.o.g., we assume that $x(c_1) \le x(c_2) \le \cdots \le x(c_n)$. By $\square(p_1, p_2, p_3, p_4)$, we mean the rectangle with corner points p_1, p_2, p_3, p_4 in clockwise order. In this section, we give an $O(n^4)$ time algorithm to solve the MBS problem on $G_{\mathscr{D}}$; we assume that the optimal solution has size greater than 2 (if it is at most 2, then we can find it easily).

We first show that the MBS problem and the MTFS problem are equivalent on $G_{\mathscr{D}}$. To this end, we need the following lemmas.

Lemma 5.5. *[NPR17] Let D_i, D_k be a pair of disks in $\mathscr{D}$ with centers c_i and c_k, respectively, where $x(c_i) \le x(c_k)$. If $D_i \cap D_k \neq \emptyset$, then for any j where $i < j < k$ either $D_i \cap D_j \neq \emptyset$ or $D_j \cap D_k \neq \emptyset$.*

Proof. We prove this by contradiction. Suppose that $D_i \cap D_k \neq \emptyset$. Let R be the rectangle such that $R = \square((x(c_i), 0), (x(c_i), 1), (x(c_k), 1), (x(c_k), 0))$. Since $x(c_i) \le x(c_j) \le x(c_k)$, c_j must belong to R. Let $m = (x(c_i) + x(c_k))/2$. We partition the rectangle R into two rectangles A and B in the following way:

$$A = \square((x(c_i),0),(x(c_i),1),(m,1),(m,0)),$$

$$B = \square((m,0),(m,1),(x(c_k),1),(x(c_k),0)).$$

Since $D_i \cap D_k \neq \emptyset$, we have $d(c_i,c_k) \leq 2$. Then, $d((x(c_i),0),(m,0)) \leq 1$ and $d((m,0),(x(c_k),0)) \leq$ 1. Now, if $c_j \in A$ then $d(c_i,c_j) \leq \sqrt{2}$ and so $D_i \cap D_j \neq \emptyset$. Otherwise, if $c_j \in B$, then $d(c_j,c_k) \leq \sqrt{2}$ and so $D_j \cap D_k \neq \emptyset$. See Figure 5.2 for an illustration. Since both cases lead to a contradiction, the lemma holds. ■

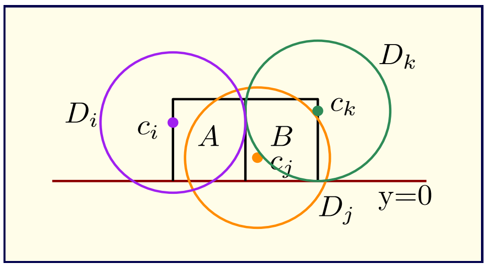


Figure 5.2: An illustration in supporting the proof of Lemma 5.5.

Lemma 5.6. *Let D_i, D_k be a pair of disks in $\mathscr{D}$ with centers c_i and c_k, respectively, where $x(c_i) \leq x(c_k)$. If $D_i \cap D_k \neq \emptyset$, then for any pair j, j' where $i < j < j' < k$ either $D_j \cap D_{j'} \neq \emptyset$ or $D_i \cap D_j \neq \emptyset$, $D_{j'} \cap D_k \neq \emptyset$. Moreover, there always exists a set Z of at least $\lceil (k-i)/2 \rceil$ pairwise adjacent disks in $\{D_{i+1},\ldots,D_{k-1}\}$ such that either $D_i \cap z \neq \emptyset$, for all $z \in Z$, or $D_k \cap z \neq \emptyset$, for all $z \in Z$.*

Proof. It is given that D_i intersects the disk D_k. Let R be the rectangle such that $R = \square((x(c_i),0),(x(c_i),1),(x(c_k),1),(x(c_k),0))$. Since $x(c_i) \leq x(c_k)$, each c_j, where $i < j < k$ must belong to R. Let $m = (x(c_i)+x(c_k))/2$. We partition the rectangle R into two rectangles A and B in the following way:

$$A = \square((x(c_i),0),(x(c_i),1),(m,1),(m,0)),$$

$$B = \square((m,0),(m,1),(x(c_k),1),(x(c_k),0)).$$

From the construction of the partition, we can say that if there is a pair of centers lying in the same ractangle, then the corresponding pair of disks should have non-empty intersection. Now for any pair of disks D_j and $D_{j'}$, if c_j and $c_{j'}$ belongs to the same rectangle of A, B then $D_j \cap D_{j'} \neq \emptyset$. Else $c_j \in A$ and $c_{j'} \in B$ together imply $D_i \cap D_j \neq \emptyset$, $D_{j'} \cap D_k \neq \emptyset$. By pigeonhole principle, at least one of A and B contain the centers of $\lceil k - i/2 \rceil$ disks from $\{D_{i+1},\ldots,D_{k-1}\}$. So there always exists a set Z of at least $\lceil (k-i)/2 \rceil$ pairwise adjacent disks in $\{D_{i+1},\ldots,D_{k-1}\}$ such that either $D_i \cap z \neq \emptyset$, for all $z \in Z$, or $D_k \cap z \neq \emptyset$, for all $z \in Z$. ■

Relation to co-comparability graph [Mou18]. A graph G is said to be a co-comparability graph if and only if the edge set of its complement graph G^c admits a transitive orientation. That means it is possible to orient the edges in $E(G^c)$ such that for every triple u,v,w in $V(G^c)$, if the edge (u,v) is oriented $u \to v$ and the edge (v,w) is oriented $v \to w$, then the edge (u,w) should exists and have orientation $u \to w$. To prove $G_{\mathscr{D}}$ is a co-comparability graph, we give a transitive orientation on the edges of $G^c_{\mathscr{D}}$. For any pair of non-intersecting disks D_i, D_j in $\mathscr{D}$ where $i < j$, we make use of the orientation as follows: $D_i \to D_j$. The correctness of this orientation directly follows from Lemma 5.12. Hence the following lemma holds.

Lemma 5.7. *Let $\mathscr{D}$ be a set of unit-disks intersecting a straight line where all the centers lie on one side of the line. Then $\mathscr{D}$ induces a co-comparability graph.*

It is easy to observe that $G_{\mathscr{D}}$ can contain an induced 4-cycle; see Figure 5.3 for an example.

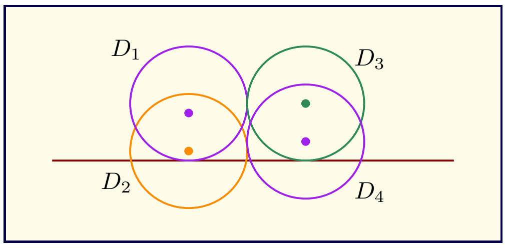


Figure 5.3: An example of a 4-cycle.

Now we prove the following lemma.

Lemma 5.8. *There is no induced cycle of length at least 5 in $G_{\mathscr{D}}$.*

Proof. We prove this lemma by contradiction. Let $G_{\mathscr{D}}$ contains an induced cycle $C = \langle D_1, D_2, \ldots, D_r, \ldots, D_k \rangle$ of length $k \geq 5$. W.l.o.g. we assume that $\forall i,\ 2 \leq i \leq k,\ x(c_1) \leq x(c_i)$. Let D_r be the right-most disk according to the increasing order of x-coordinates of the centers of disks in the cycle C. Now there are two disjoint paths π_1 and π_2 in C between the vertices corresponding to D_1 and D_r. Further we assume that the number of disks in π_1 is at most the number of disks in π_2. The other case is analogous. Our proof is based on the number of vertices in π_1.

First consider the case that $D_1 \cap D_r \neq \emptyset$. In this case π_1 consists of disks $\{D_1, D_2 (= D_r)\}$ and π_2 consists of disks $\{D_2, D_3, \ldots, D_k, D_1\}$. Note that the number of disks in π_2 is at least 5. So there must be a disk $D_i \in \pi_2$ such that $D_i \cap D_1 = \emptyset$ and $D_i \cap D_2 = \emptyset$. As $x(c_1) \leq x(c_i) \leq x(c_2)$, by Lemma 5.5, $D_1 \cap D_2 = \emptyset$. This contradicts our assumption.

Now we consider the case that $\pi_1 = (D_1, D_2, \ldots, D_r)$, where r is at least 3 and $\pi_2 = (D_1, D_k, \ldots, D_{r+1}, D_r)$. Note that the number of disks in π_1 is at least three and the number of disks in π_2 is at least four. Now we consider following three subcases based on location of centers of $D_2, \ldots, D_{r-1}$.

Case 1: $x(c_k) \leq x(c_j) \leq x(c_{k-1})$ holds for at least one j $\{2, \ldots, r-1\}$. As $D_k \cap D_j = \emptyset$ and $D_j \cap D_{k-1} = \emptyset$, so by Lemma 5.5, D_k and D_{k-1} should not intersect each other. This contradicts our assumption that $D_k \cap D_{k-1} \neq \emptyset$.

Case 2: $x(c_i) \leq x(c_k)$, $\forall i$, $2 \leq i \leq r-1$. As $D_{r-1} \cap D_k = \emptyset$ and $D_k \cap D_r = \emptyset$, so by Lemma 5.5, D_{r-1} and D_r should not intersect each other. This contradicts our assumption that $D_{r-1} \cap D_r \neq \emptyset$.

Case 3: $x(c_{k-1}) \leq x(c_i)$, $\forall i$, $2 \leq i \leq r-1$. As $D_1 \cap D_{k-1} = \emptyset$ and $D_{k-1} \cap D_2 = \emptyset$, so by Lemma 5.5, D_1 and D_2 should not intersect each other. This contradicts our assumption that $D_1 \cap D_2 \neq \emptyset$. ■

By Lemma 5.8, the MBS and MTFS problems are equivalent on $G_{\mathscr{D}}$. Therefore, in the following, we focus on solving the MTFS problem on $G_{\mathscr{D}}$.

A dynamic-programming algorithm. Let $\mathscr{D}[i] = \{D_i, D_{i+1}, \ldots, D_n\}$. For each triple (i,j,k), where $1 \leq i < j < k \leq n$, we define $B[i,j,k]$ to be the size of a maximum induced triangle-free subgraph containing D_i, D_j and $D_k \in D_i \cup D_j \cup \mathscr{D}[k]$. Let $D[i,j,k] = \{D_l : D_l \in \mathscr{D}[k+1]$ and $\{D_i, D_j, D_k, D_\ell\}$ induces a K_3-free subgraph$\}$. We now describe how to compute $B[i,j,k]$ for each triple (i,j,k). If D_i, D_j and D_k form a K_3, then clearly $B[i,j,k] = 0$ as there is no bipartite subgraph containing D_i, D_j and D_k. If D_i, D_j and D_k do not form a K_3 and $D[i,j,k] = \emptyset$, then $B[i,j,k] = 3$ and in that case D_i, D_j and D_k are the disks that induce maximum induced triangle-free subgraph in $D_i \cup D_j \cup \mathscr{D}[k]$. On the other hand, if D_i, D_j, D_k do not form a K_3 and $D[i,j,k] \neq \emptyset$, then $B[i,j,k]$ is one more than the size of a maximum induced triangle-free subgraph containing D_j, D_k, D_l among all $D_l \in D[i,j,k]$. Hence, we obtain the following recurrence for $B[i,j,k]$.

$$B[i,j,k] = \begin{cases} 0, & D_i, D_j, D_k \text{ form a } K_3, \\ 3, & D_i, D_j, D_k \text{ do not form a } K_3 \text{ and } D[i,j,k] = \emptyset, \\ 1 + \max\limits_{\forall D_l \in D[i,j,k]} B[j,k,l], & \text{otherwise.} \end{cases}$$

The size of an optimal solution is the maximum value in the table B. To show the correctness of our algorithm, we prove the following.

Lemma 5.9. *Let $B[i,j,k] > 3$, and $D_\ell \in D[i,j,k]$. Then, no two disks corresponding to $B[j,k,\ell]$ form a triangle with D_i.*

We need several helper lemmas before proving Lemma 5.9.

Lemma 5.10. *Let $D_i, D_j, D_k, D_\ell \in \mathscr{D}$ be four disks with $x(c_i) \le x(c_j) \le x(c_k) \le x(c_\ell)$. If $D_i \cap D_\ell \neq \emptyset$, then the subgraph induced by $\{D_i, D_j, D_k, D_\ell\}$ is $K_{1,3}$-free.*

Proof. We prove the lemma by contradiction. Suppose that the subgraph induced by $\{D_i, D_j, D_k, D_\ell\}$ is isomorphic to $K_{1,3}$. Then either (i) D_i intersects both D_j and D_k, where D_j, D_k, D_ℓ are pairwise independent, or (ii) D_ℓ intersects both D_j and D_k, where D_i, D_j, D_k are pairwise independent. Now by Lemma 5.6 either $D_j \cap D_k \neq \emptyset$ or $D_i \cap D_j \neq \emptyset$, $D_k \cap D_\ell \neq \emptyset$. Since both the cases lead to a contradiction, the lemma holds. ■

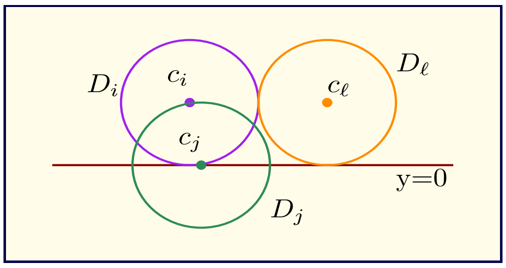


Figure 5.4: Existence of $K_{1,2}$ in one side.

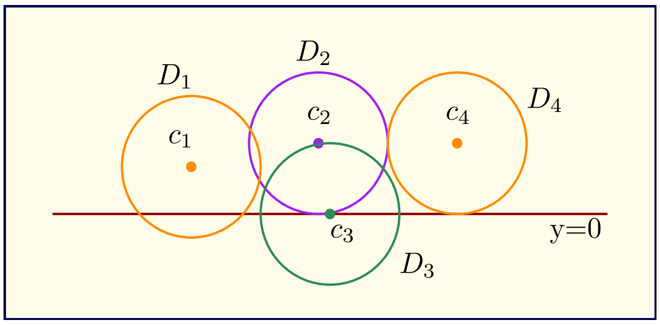


Figure 5.5: Existence of $K_{1,3}$ in $G_{\mathscr{D}}$.

Lemma 5.11. *The graph $G_{\mathscr{D}}$ is $K_{1,4}$-free.*

Proof. We prove the lemma again by contradiction. Suppose there is an induced subgraph $K_{1,4}$ with vertex set $\{D_i : 1 \le i \le 5\}$. Assume w.l.o.g. that $x(c_1) \le x(c_2) \le x(c_3) \le x(c_4) \le x(c_5)$. In $\{D_i : 1 \le i \le 5\}$, there must be four pair-wise independent disks and one disk (say, D) that intersects each of that four disks. If $D = D_1$ or D_4. Then the subgraph induced by $\{D_i : 1 \le i \le 4\}$ is isomorphic to $K_{1,3}$. This contradicts Lemma 5.10. Similarly if $D = D_2$ or D_5. Then the subgraph induced by $\{D_j : 2 \le j \le 5\}$ is isomorphic to $K_{1,3}$. This contradicts Lemma 5.10. Hence only possibility is that $D = D_3$ and the edge set in $K_{1,4}$ is $\{(D_1,D_3),(D_2,D_3),(D_3,D_4),(D_3,D_5)\}$.

Now, consider the rectangle $R = \Box((x(c_3)-2,0),(x(c_3)-2,1),(x(c_3)+2,1),(x(c_3)+2,0)))$. We partition R into three rectangles A,B and C as follows:

$$A = \Box((x(c_3)-2,0),(x(c_3)-2,1),(x(c_3)-0.5,1),(x(c_3)-0.5,0)),$$

$$B = \Box((x(c_3)-0.5,0),(x(c_3)-0.5,1),(x(c_3)+0.5,1),(x(c_3)+0.5,0)),$$

$$C = \Box((x(c_3)+0.5,0),(x(c_3)+0.5,1),(x(c_3)+2,1),(x(c_3)+2,0)).$$

Since c_i belongs to R, for $1 \le i \le 5$, by the pigeonhole principle, one of A, B and C must contain two of $\{c_1, c_2, c_4, c_5\}$. The distance between any two points in the rectangles A, B or C is ≤ 2. Therefore, $\{D_1,D_2,D_4,D_5\}$ are not pairwise independent. Hence, $G_{\mathscr{D}}$ is $K_{1,4}$-free. ■

Lemma 5.12. *Let D_1,D_2,D_3,D_4,D_5 be five disks in $\mathscr{D}$ with $x(c_i) \le x(c_{i+1})$ where $1 \le i \le 4$. If $D_1 \cap D_5 \neq \emptyset$, then the disks D_2,D_3,D_4 do not form an independent set.*

Proof. The proof directly follows from Lemma 5.6 as $D_1 \cap D_5 \neq \emptyset$ implies there always exist a pair of adjacent disks in $\{D_2,D_3,D_4\}$. ■

We are now ready to prove Lemma 5.9.

Proof of Lemma 5.9 By our assumption, we consider that $B[i,j,k] > 3$ and $D_\ell \in \{D_{k+1},\ldots,D_n\}$ where the subgraph induced by $\{D_i,D_j,D_k,D_\ell\}$ is triangle-free. Let $\Psi_{j,k,\ell}$ be the set of disks that defines $B[j,k,\ell]$. Since $B[i,j,k] > 3$, $\Psi_{j,k,\ell} \neq \emptyset$. We need to show that for any pair of disks $D,D' \in \Psi_{j,k,\ell}$, the subgraph induced by $\{D_i,D,D'\}$ is triangle-free. We prove this by contradiction. Suppose for a contradiction that D and D' be two disks corresponding to $B[j,k,\ell]$ such that D_i,D and D' form a triangle. As the subgraph induced by $\{D_j,D_k,D_\ell,D,D'\}$ is triangle-free. By the definition of $D[i,j,k]$, at most one of D and D' belongs to $\{D_j,D_k,D_\ell\}$.

Case 1: $\{D,D'\} \cap \{D_j,D_k,D_\ell\} \neq \emptyset$. Assume w.l.o.g. that $D' \in \{D_j,D_k,D_\ell\}$. We consider several cases.

- $D' = D_j$. Since $D_i \cap D \neq \emptyset$, by Lemma 5.5, (i) either $D_i \cap D_k \neq \emptyset$ or $D_k \cap D \neq \emptyset$ as well as (ii) either $D_i \cap D_\ell \neq \emptyset$ or $D_\ell \cap D \neq \emptyset$. Similarly, since $D_j \cap D \neq \emptyset$, by

Lemma 5.5, (i) either $D_j \cap D_\ell \neq \emptyset$ or $D_\ell \cap D \neq \emptyset$. Suppose that $D_k \cap D \neq \emptyset$. Now, if $D_\ell \cap D \neq \emptyset$, then $\{D_j, D_k, D_l, D\}$ forms a $K_{1,3}$. This contradicts Lemma 5.10. Therefore, $D_\ell \cap D = \emptyset$; consequently, $D_j \cap D_\ell \neq \emptyset$ and $D_i \cap D_\ell \neq \emptyset$. Then, the disks D_i, D_j, D_ℓ form a triangle. A contradiction. So, D_k and D should not intersect each other. Then, $D_i \cap D_k \neq \emptyset$ and $D_j \cap D_k \neq \emptyset$. Consequently, the disks D_i, D_j, D_k form a triangle and so $B[i,j,k] = 0$. This contradicts our assumption that $B[i,j,k] > 3$.

- $D' = D_k$. Since $D_i \cap D \neq \emptyset$, by Lemma 5.5, either $D_i \cap D_j \neq \emptyset$ or $D_j \cap D \neq \emptyset$. Suppose that $D_j \cap D \neq \emptyset$. Since $D_k \cap D \neq \emptyset$, either $D_k \cap D_\ell \neq \emptyset$ or $D_\ell \cap D \neq \emptyset$ (again by Lemma 5.5). Let $D_\ell \cap D \neq \emptyset$. Then, $\{D_j, D_k, D_\ell, D\}$ forms a $K_{1,3}$. This contradicts Lemma 5.10. So, $D_\ell \cap D = \emptyset$, that means that $D_k \cap D_\ell \neq \emptyset$ and $D_i \cap D_\ell \neq \emptyset$. Consequently, the disks D_i, D_k, D_ℓ form a triangle. A contradiction. This means that the disks D_j and D do not intersect each other and so $D_i \cap D_j \neq \emptyset$. Now, if $D_i \cap D_\ell \neq \emptyset$, then $\{D_i, D_j, D_k, D_\ell\}$ forms a $K_{1,3}$. This contradicts Lemma 5.10. Thus, the disks D_i and D_ℓ do not intersect each other and so $D_\ell \cap D \neq \emptyset$. We know that previously $D_i \cap D \neq \emptyset$ and $D_i \cap D_j \neq \emptyset$. Since $D_j \cap D_k = \emptyset$ and $D_k \cap D_\ell = \emptyset$, we have $D_j \cap D_\ell = \emptyset$. Thus, the disks D_j, D_k, D_ℓ are now pairwise independent. In this case, the disks D_i, D_j, D_k, D_ℓ, D together contradict Lemma 5.12.

- $D' = D_\ell$. Proof is analogous to the case $D' = D_j$. Since $D_i \cap D \neq \emptyset$, by Lemma 5.5, (i) either $D_i \cap D_j \neq \emptyset$ or $D_j \cap D \neq \emptyset$ as well as (ii) either $D_i \cap D_k \neq \emptyset$ or $D_k \cap D \neq \emptyset$. Similarly, since $D_i \cap D_\ell \neq \emptyset$, by Lemma 5.5, (i) either $D_i \cap D_j \neq \emptyset$ or $D_j \cap D_\ell \neq \emptyset$. Suppose that $D_i \cap D_k \neq \emptyset$. Now, if $D_i \cap D_j \neq \emptyset$, then $\{D_i, D_j, D_k, D_\ell\}$ forms a $K_{1,3}$. A contradiction to Lemma 5.10. Therefore, $D_i \cap D_j = \emptyset$; consequently, $D_j \cap D_\ell \neq \emptyset$ and $D_\ell \cap D \neq \emptyset$. Then, the disks D_j, D_ℓ, D form a triangle. A contradiction. So, D_i and D_j should not intersect each other. Then, $D_j \cap D_\ell \neq \emptyset$ and $D_\ell \cap D \neq \emptyset$. Consequently, the disks D_j, D_ℓ, D form a triangle. This contradicts the fact the subgraph induced by $\Psi_{j,k,\ell}$ is triangle-free.

Case 2: $\{D, D'\} \cap \{D_j, D_k, D_\ell\} = \emptyset$. Since $D_i \cap D \neq \emptyset$, by Lemma 5.5, either $D_i \cap D_j \neq \emptyset$ or $D_j \cap D \neq \emptyset$. Assume that $D_j \cap D \neq \emptyset$. If D_j and D' intersects, then we get a triangle formed by D_j, D, D'. This contradicts the definition of $B[j,k,\ell]$. But, since $D_i \cap D' \neq \emptyset$, by Lemma 5.5, either $D_i \cap D_j \neq \emptyset$ or $D_j \cap D' \neq \emptyset$. We have already shown that $D_j \cap D' = \emptyset$ and so $D_i \cap D_j \neq \emptyset$. Then, the disks $\{D_i, D_j, D\}$ forms a triangle. Observe that this is Case 1 and so $D_j \cap D = \emptyset$. By a similar argument, one can show that $D_k \cap D = \emptyset$ and $D_\ell \cap D = \emptyset$. By Lemma 5.5, since $D_i \cap D \neq \emptyset$,

$D_i \cap D_j \neq \emptyset, D_i \cap D_k \neq \emptyset, D_i \cap D_\ell \neq \emptyset$. This implies that $\{D_j, D_k, D_\ell\}$ are pairwise independent. Then $\{D_i, D_j, D_k, D_\ell\}$ forms a $K_{1,3}$. This contradicts to Lemma 5.10. ■

Complexity. For each triple (i,j,k) we find $B[i,j,k]$. Then we output one that maximizes the size. To compute $B[i,j,k]$, we have to compare $O(n)$ subproblems. So the total time complexity is $O(n^4)$. Since table B is of size $O(n^3)$, the total space complexity is $O(n^3)$. Therefore, we have the following theorem.

Theorem 5.4. *Let $\mathscr{D}$ be a set of n unit-disks intersecting a straight line where all the centers lie on one side of the line. In $O(n^4)$ time and $O(n^3)$ space, we can find maximum-sized $\mathscr{D}' \subseteq \mathscr{D}$ such that $\mathscr{D}'$ induces a bipartite subgraph.*

5.4.2 A 2-approximation for MBS on unit disks intersecting a line

We now consider the MBS problem when the set of disks in $\mathscr{D}$ intersect the x-axis (denoted by L afterward) from both sides; that is, the centers can lie on both sides of the x-axis. Observe that here the corresponding graph $G_{\mathscr{D}}$ might have induced cycles of arbitrary length (see Figure 5.6 for an example).

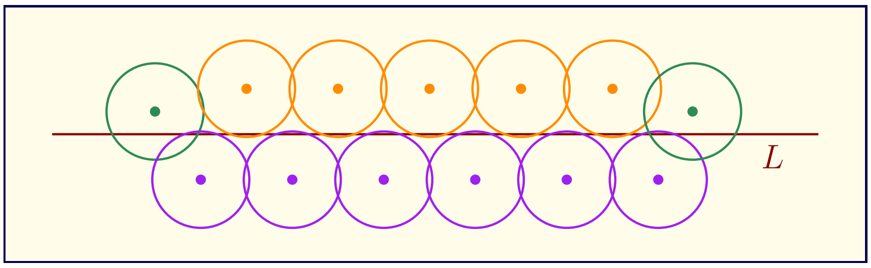


Figure 5.6: An induced cycle of 13 disks intersecting the line L.

We give an $O(n + |E(G_{\mathscr{D}})|)$-time 2-approximation algorithm for the MBS problem, where $E(G_{\mathscr{D}})$ is the set of all edges in $G_{\mathscr{D}}$. To this end, we partition the set $\mathscr{D}$ into two sets $\mathscr{D}_a$ and $\mathscr{D}_b$, where $\mathscr{D}_a$ is the set of all disks in $\mathscr{D}$ whose centers lie above or on L and $\mathscr{D}_b$ is the set of all disks in $\mathscr{D}$ whose centers are strictly below L. Let $G_{\mathscr{D}_a}$ (resp. $G_{\mathscr{D}_b}$) be the intersection graph of disks in $\mathscr{D}_a$ (resp. $\mathscr{D}_b$). A co-comparability graph is a graph whose complement admits a transitive orientation. Now, using Lemma 5.7, we can claim that both graphs $G_{\mathscr{D}_a}$ and $G_{\mathscr{D}_b}$ are co-comparability graphs. Kohler et al. [KM16] gave an $O(|V| + |E|)$-time algorithm to compute a maximum-weighted independent set on a co-comparability graph $G = (V, E)$. Let MIS($G_{\mathscr{D}_a}$) and MIS($G_{\mathscr{D}_b}$) be the maximum independent sets on $G_{\mathscr{D}_a}$ and $G_{\mathscr{D}_b}$, respectively. Clearly, the vertices in $\mathsf{MIS}(G_{\mathscr{D}_a}) \cup \mathsf{MIS}(G_{\mathscr{D}_b})$ induce a bipartite subgraph of $G_{\mathscr{D}}$. Notice that $|\mathsf{MIS}(G_{\mathscr{D}})| \leq |\mathsf{MIS}(G_{\mathscr{D}_a})| + |\mathsf{MIS}(G_{\mathscr{D}_b})|$. Let $\mathsf{MBS}(G_{\mathscr{D}})$ be an optimal solution for the MBS problem on $G_{\mathscr{D}}$. Using Corollary 5.1, we obtain $|\mathsf{MBS}(G_{\mathscr{D}})| \leq 2(|\mathsf{MIS}(G_{\mathscr{D}_a})| + |\mathsf{MIS}(G_{\mathscr{D}_b})|)$. Hence, $\mathsf{MIS}(G_{\mathscr{D}_a}) \cup \mathsf{MIS}(G_{\mathscr{D}_b})$ gives a factor-2 for the MBS problem on $G_{\mathscr{D}}$. Therefore, we have the following theorem.

Theorem 5.5. *Let $\mathscr{D}$ be a set of n unit disks intersecting a line in the plane and let $G_{\mathscr{D}}$ be the intersection graph of $\mathscr{D}$. An induced bipartite subgraph of size at least $|\mathsf{MBS}(G_{\mathscr{D}})|/2$ can be computed in $O(n + |E(G_{\mathscr{D}})|)$ time, where $\mathsf{MBS}(G_{\mathscr{D}})$ is an optimal solution for the MBS problem on $G_{\mathscr{D}}$.*

Matsui [Mat98] showed a polynomial algorithm to solve the MIS problem for families of unit disks contained in a slab of fixed height. Since a family of unit disks intersecting a line is contained in a slab of height four, Matsui's result together with Observation 5.1 generalize the above theorem.

5.5 MBS on arbitrary Unit Disks

Recall that since MIS is NP-Complete on unit disks, the MBS problem is NP-Complete on these graphs by Theorem 5.3. Also, Nandy et al. [NPR17] designed a factor 2-approximation algorithm for the MIS problem on unit disks, which runs in $O(n^2)$ time. Consequently, by Observation 5.1 we obtain an $O(n^2)$ time 4-approximation algorithm for the MBS problem on unit disks. In this section, we first give PTASes for MBS, and will then consider the problem of finding several approximation algorithms. More specifically, we give an $O(n^4)$ time 3-approximation and an $O(n \log n)$ time $O(\log n)$-approximation algorithm for MBS on unit disks.

5.5.1 PTAS

We first show the PTAS for MBS on unit disks, and will then discuss it for the *weighted* MBS problem. Let S be a set of n unit disks in the plane, and let $k > 1$ be a fixed integer. A PTAS running in $O(n^{O(1)} \cdot n^{O(1/\epsilon^2)})$ time, for any $\epsilon > 0$, is straightforward using the shifting technique of Hochbaum and Maass [HM85b] and the following packing argument: for an instance of the MBS problem, where the unit disks lie inside a $k \times k$ square, an optimal solution cannot have more than k^2 unit disks. Hence, we can compute an exact solution for such an instance of the problem in $O(n^{O(1)} \cdot n^{O(k^2)})$ time. Consequently, by setting $k = 1/\epsilon$, we obtain a PTAS running in time $O(n^{O(1)} \cdot n^{O(1/\epsilon^2)})$.

To improve the running time to $O(n^{O(1)} \cdot n^{O(1/\epsilon)})$, we rely on the shifting technique again, but instead of applying the plane partitioning twice, we only partition the plane into horizontal slabs and solve the MBS problem for each of them exactly. This gives us the desired running time for our PTAS. We next describe the details of how to solve MBS exactly for a slab.

Algorithm for a slab. Let H be a horizontal slab of height k and let $D \subseteq S$ be the set of disks that lie entirely inside H. The idea is to build a vertex-weighted directed acyclic graph G such that finding a maximum-weight path from the source vertex to the target vertex corresponds to an exact solution for the MBS problem [Mat98]. To this end, let a and b $(a < b)$ be two integers such that every disk in D lies inside the rectangle R bounded by H and the vertical lines $x = a$ and $x = b$. Partition R vertically into width 2 boxes B_i, where the left side of B_i has the x-coordinate $a + i$, for all integers $0 \le i < b - a$; let $D_i \subseteq D$ denote the set of disks whose centers lie inside B_i. Since B_i has height k and width 2, we can compute all feasible (not necessarily exact) solutions for the MBS problem on D_i in $O(n^{O(1)} \cdot n^{O(k)})$ time; let $\mathcal{M}_i$ be the set of all such feasible solutions. We now build a directed vertex-weighted acyclic graph G as follows. Each feasible solution of $\mathcal{M}_i$ corresponds to some vertices. Note that each feasible solution is a bipartite graph so there can be many way of bipartitioning. We creat vertices for each possible bipartition. For a solution M, we denote each such vertices as $(M, V_1(M), V_2(M))$. The vertex set of $V(G)$ is $V \cup \{s, t\}$, where V has vertices for each solution in $\mathcal{M}_i$, for all i. Moreover, the weight of each vertex is the number of disks in the corresponding bipartite graph. For every pair i, j, where $1 \le i < j < n$, consider two solutions $(M, V_1(M), V_2(M)) \in \mathcal{M}_i$ and $(M', V_1(M'), V_2(M')) \in \mathcal{M}_j$. Then, there exists an edge from the vertex of M to that of M' in G if the intersection graph induced by the disks in $M \cup M'$ is bipartite with one of the following two bipartition: $(V_1(M) \cup V_1(M'), V_2(M) \cup V_2(M'))$ and $(V_1(M) \cup V_2(M'), V_2(M) \cup V_1(M'))$. Finally, for all i and for all $M \in \mathcal{M}_i$: there exists an edge from s to $(M, V_1(M), V_2(M))$, and there exists an edge from $(M, V_1(M), V_2(M))$ to t. The weights of vertices s and t are zero.

Lemma 5.13. *The* MBS *problem has a feasible solution of size k on G if and only if there exists a directed path from s to t with the total weight k.*

Proof. For a given directed st-path with total weight k, let X be the union of all the disks corresponding to the internal vertices of this path. Then, the intersection graph of X is bipartite because the disks in $X \cap \mathcal{M}_i$ are disjoint from the disks in $S \cap \mathcal{M}_j$ when $j > i + 1$. Moreover, when $j = i + 1$, the disks in $X \cap (\mathcal{M}_i \cup \mathcal{M}_j)$ must form an induced bipartite graph by the definition of an edge in G. Since the total weight of the vertices on the path is k, we have $|X| = k$. On the other hand, let Y be a feasible solution of size k for the MBS problem on G. Then, the intersection graph of disks in $Y \cap D_i$ is bipartite, for all i. Hence, selecting the vertices corresponding to $Y \cap D_i$ for all i gives us a path with total weight k from s to t. ■

By Lemma 5.13, the MBS problem for H reduces to the problem of finding the maximum-weighted path from s to t on G. The number of vertices of G that correspond to feasible solutions for the MBS problem on disks in $S \cap D_i$ is bounded by $O(n^{O(k)})$, which is the bound on the number of vertices of G that correspond to these feasible solutions. Hence, we can compute the edge set of G in $O(n^{O(1)} \cdot n^{O(k)})$ time (we can check whether a subset of disks form a bipartite graph in $O(n^{O(1)})$ time). Since G is directed and acyclic, the maximum-weighted st-path problem can be solved in linear time. Therefore, by setting $k = 1/\epsilon$, we have the following theorem.

Theorem 5.6. *There exists a* PTAS *for* MBS *on unit disks that runs in* $O(n^{O(1)} \cdot n^{O(1/\epsilon)})$ *time, for any* $\epsilon > 0$.

5.5.2 2-approximation algorithm

Here we give an $O(n^4)$ time 2-approximation algorithm. Our approach is motivated by the technique of Agarwal et al. [AvKS98], where they gave a $O(n \log n)$-time 2-approximation algorithm for the maximum independent set problem on a set of n unit-height rectangles in the plane. Consider a set $\mathscr{D}$ of n unit-disks in the plane. We first place a set of horizontal lines, two distance apart, such that each disk is intersected by a line which is positioned below the centre of the disk. Note that such a placement is always possible.

Suppose there are $k \leq n$ such horizontal lines that stab all the disks. Let $\{L_1, L_2, \ldots, L_k\}$ be the set of these lines and they partition the set $\mathscr{D}$ into subsets $\mathscr{D}_1, \mathscr{D}_2, \ldots, \mathscr{D}_k$, where $\mathscr{D}_i \subseteq \mathscr{D}$ is the set of those disks that are intersected by the line L_i and centers are lying above or on L_i. Now we have the following observation.

Observation 5.2. *Let* $D \in \mathscr{D}_i$ *and* $D' \in \mathscr{D}_j$ *be two disks in* $\mathscr{D}$*, where* $1 \leq i, j \leq k$*. If* $|i - j| \geq 2$ *then* D *and* D' *do not intersect.*

We now apply the algorithm to solve the MBS problem on each $\mathscr{D}_i$ ($1 \leq i \leq k$). Thus, we compute a maximum bipartite subgraph B_i on each $\mathscr{D}_i$ in $O(|\mathscr{D}_i|^4)$ time. Consider the two bipartite subgraphs $\{B_1 \cup B_3, \ldots, B_{2\lceil k/2 \rceil - 1}\}, \{B_2 \cup B_4, \ldots, B_{2\lfloor k/2 \rfloor}\}$. Let $\mathsf{MBS}(G_{\mathscr{D}})$ denote a maximum bipartite subgraph on the graph induced by the objects in $\mathscr{D}$. It is easy to observe that $\sum_{i=1}^{k}|B_i| \geq |\mathsf{MBS}(G_{\mathcal{D}})|$. So, the largest set among these two must have size at least $|\mathsf{MBS}(G_{\mathcal{D}})|/2$. Thus, we obtain a 2-approximation algorithm. To find the lines L_i (and form the corresponding partition), we need to do a single pass through the disks after sorting them by the y-coordinates of their centers. Thus, running time of the algorithm is bounded by $O(n^4)$. We hence have the following theorem.

Theorem 5.7. *Let $\mathcal{D}$ be a set of n unit disks in the plane and let $G_{\mathcal{D}}$ be the intersection graph of $\mathcal{D}$. An induced bipartite subgraph of size at least $|\mathsf{MBS}(G_{\mathcal{D}})|/2$ can be computed in $O(n^4)$ time, where $\mathsf{MBS}(G_{\mathcal{D}})$ is a maximum induced bipartite subgraph of $G_{\mathcal{D}}$.*

5.5.3 $O(\log n)$-approximation algorithm

Here, we describe an $O(\log n)$-approximation algorithm for MBS problem in unit disk graphs that runs in $O(n \log n)$ time. This algorithm is also motivated by Agarwal et al. [AvKS98] who gave an $O(n \log n)$-time $O(\log n)$-approximation algorithm for the maximum independent set problem for a set of n axis-parallel rectangles in the plane. Let $\mathcal{D} = \{D_1, D_2, \ldots D_n\}$ and let c_i be the centers of the disks $D_i, 1 \leq i \leq n$. We sort the centers of the disks in $\mathcal{D}$ by their x-coordinates. This takes $O(n \log n)$ time. If $n \leq 2$, we solve problem in constant time. Otherwise, we do the following.

1. Let c_{med} be the center having the median x-coordinate among all the centers.

2. Draw the vertical line $x = x(c_{med})$.

3. Partition the disks $\mathcal{D}$ into three sets $\mathcal{D}_\ell$, $\mathcal{D}_{med}$, and $\mathcal{D}_r$ defined by

$$\mathcal{D}_\ell = \{D_i : D_i \in \mathcal{D}, x(c_{med}) - x(c_i) > 1\},$$
$$\mathcal{D}_{med} = \{D_i : D_i \in \mathcal{D}, |x(c_{med}) - x(c_i)| \leq 1\},$$
$$\mathcal{D}_r = \{D_i : D_i \in \mathcal{D}, x(c_i) - x(c_{med}) > 1\}.$$

4. Compute B_{med}, a 2-approximation solution for the MBS problem on disks in $\mathcal{D}_{med}$ using Theorem 5.5.

5. Recursively compute B_ℓ and B_r, the approximate maximum bipartite subgraph induced by $\mathcal{D}_\ell$ and $\mathcal{D}_r$, respectively.

6. Return B_{med} if $|B_{med}| \geq |B_\ell \cup B_r|$; otherwise, return $B_\ell \cup B_r$.

Observe that for every pair of disks $D \in \mathscr{D}_\ell$ and $D' \in \mathscr{D}_r$, $D \cap D' = \emptyset$. This implies that $B_\ell \cup B_r$ induces a bipartite subgraph of $G_{\mathscr{D}}$, intersection graph of $\mathscr{D}$. By Theorem 5.5, we can obtain a 2-approximation solution for the problem on $\mathscr{D}_{med}$ in linear time. Since $|\mathscr{D}_\ell| \le n/2$ and $|\mathscr{D}_r| \le n/2$, the overall running time is $O(n\log n + |E|)$, where E is the set of edges in $G_{\mathscr{D}}$.

Next, we show that our algorithm computes a bipartite subgraph of size at least $|\mathsf{MBS}(G_{\mathscr{D}})|/\max(1, 2\log n)$, where $\mathsf{MBS}(G_{\mathscr{D}})$ is a maximum bipartite subgraph in $G_{\mathscr{D}}$. The proof is by induction on n. For the base case, the claim is true for $n \le 2$ since we can compute a maximum bipartite subgraph in constant time. For inductive step, suppose the claim is true for all $k < n$. Let B^*_ℓ, B^*_r and B^*_{med} denote the maximum bipartite subgraph on $G_{\mathscr{D}_\ell}, G_{\mathscr{D}_r}$ and $G_{\mathscr{D}_{med}}$, respectively. Since we compute a 2-approximation solution for the problem on $G_{\mathscr{D}_{med}}$, we have

$$|B_{med}| \ge \frac{|B^*_{med}|}{2} \ge \frac{|\mathsf{MBS}(G_{\mathscr{D}}) \cap \mathscr{D}_{med}|}{2}.$$

By the induction hypothesis,

$$|B_\ell| \ge \frac{|B^*_\ell|}{2\log(n/2)} \ge \frac{|\mathsf{MBS}(G_{\mathscr{D}}) \cap \mathscr{D}_\ell|}{2(\log n - 1)} \text{ and similarly } |B_r| \ge \frac{|\mathsf{MBS}(G_{\mathscr{D}}) \cap \mathscr{D}_r|}{2(\log n - 1)}.$$

Therefore,

$$\begin{aligned} |B_\ell| + |B_r| &\ge \frac{|\mathsf{MBS}(G_{\mathscr{D}}) \cap \mathscr{D}_\ell| + |\mathsf{MBS}(G_{\mathscr{D}}) \cap \mathscr{D}_r|}{2(\log n - 1)} \\ &= \frac{|\mathsf{MBS}(G_{\mathscr{D}})| - |\mathsf{MBS}(G_{\mathscr{D}}) \cap \mathscr{D}_{med}|}{2(\log n - 1)} \end{aligned}$$

If $|\mathsf{MBS}(G_{\mathscr{D}}) \cap \mathscr{D}_{med}| \ge |\mathsf{MBS}(G_{\mathscr{D}})|/\log n$, then $|B_{med}| \ge \dfrac{|\mathsf{MBS}(G_{\mathscr{D}})|}{2\log n}$ and we are done. Otherwise, $|\mathsf{MBS}(G_{\mathscr{D}}) \cap \mathscr{D}_{med}| < |\mathsf{MBS}(G_{\mathscr{D}})|/\log n$ in which case

$$\begin{aligned} |B_\ell| + |B_r| &\ge \frac{|\mathsf{MBS}(G_{\mathscr{D}})| - |\mathsf{MBS}(G_{\mathscr{D}}) \cap \mathscr{D}_{med}|}{2(\log n - 1)} \\ &\ge \frac{|\mathsf{MBS}(G_{\mathscr{D}})| - |\mathsf{MBS}(G_{\mathscr{D}})|/\log n}{2(\log n - 1)} \\ &= \frac{|\mathsf{MBS}(G_{\mathscr{D}})|}{2\log n} \end{aligned}$$

Therefore, $\max\{|B_{med}|, |B_\ell| + |B_r|\} \ge |\mathsf{MBS}(G_{\mathscr{D}})|/(2\log n)$; hence, proving the induction step.

Theorem 5.8. *Let $\mathscr{D}$ be a set of n unit disks in the plane and let $G_{\mathscr{D}} = (V, E)$ be the intersection graph of $\mathscr{D}$. An induced bipartite subgraph of size at least $|\mathsf{MBS}(G_{\mathscr{D}})|/(2\log n)$*

can be computed in $O(n \log n + |E|)$ time, where $\mathsf{MBS}(G_{\mathcal{D}})$ *denotes a maximum induced bipartite subgraph in $G_{\mathcal{D}}$.*

5.6 MBS on Unit Squares and Unit-height Rectangles

In this section we give approximation algoritms for MBS problem on unit Squares and unit-height rectangles.

5.6.1 PTAS on unit squares

Since MIS is NP-Complete on unit square graphs, the MBS problem is NP-Complete on this class of graphs by Theorem 5.3. Here, we give a PTAS for MBS on a set of n unit squares. One can verify that the algorithm to obtain Theorem 5.6 can be applied to obtain a PTAS for MBS on a set of n unit squares, as well. Moreover, the algorithm extends to the weighted MBS problem on unit disks and unit squares. The only modification is, instead of assigning the number of disks (resp. squares) in a solution as the weight of the corresponding vertex, we assign the total weight of the disks (resp. squares) in the solution as the vertex weight.

Theorem 5.9. *There exists a* PTAS *for the* MBS *problem on unit squares running in $O(n^{O(1)} \cdot n^{O(1/\epsilon)})$ time, for any $\epsilon > 0$. Moreover, the weighted* MBS *problem also admits a* PTAS *running within the same time bound on unit disks and unit squares.*

5.6.2 2-approximation for unit-height rectangles

Here, we give an $O(n \log n)$ time 2-approximation algorithm for MBS on a set of n unit-height rectangles. The technique used in this section is similar to the technique of Agarwal et al. [AvKS98]. To this end, suppose that the bottom side of the bottommost rectangle has y-coordinate a and the top side of the topmost rectangle has y-coordinate b. Consider the set of horizontal lines $y := a + i + \epsilon$ for all $i = 0, \ldots, b$, where $\epsilon > 0$ is a small constant; we may assume w.l.o.g. that each rectangle intersects exactly one line. Ordering the lines from bottom to top, let S_i be the set of rectangles that intersect the horizontal line i. We now run BIPARTITEINTERVAL(S), once for when $S = S_1 \cup S_3 \cup S_5 \ldots$ and once for when $S = S_2 \cup S_4 \cup S_6 \ldots$, and will then return the largest of these two solutions.

Algorithm 3 BIPARTITEINTERVAL(S)

```
let initially $M = \emptyset$;
$x := -\infty$ and $y := -\infty$;
for $i := 1$ to $n$ do
    if left($I_i$) $> y$ then
        $M := M \cup I_i$ and $y :=$ right($I_i$);
    else if $x <$ left($I_i$) $< y$ then
        $M := M \cup I_i$, $x := y$, and $y :=$ right($I_i$);
    end if
end for
return $M$;
```

We perform an initial sorting that takes $O(n \log n)$ time, and BIPARTITEINTERVAL(S) runs in $O(n)$ time. This gives us the following theorem.

Theorem 5.10. *There exists an $O(n \log n)$ time 2-approximation algorithm for the* MBS *problem on a set of n unit-height rectangles in the plane.*

5.7 Hardness of MTFS

In this section, we show that MTFS problem is NP-Hard when geometric objects are axis-parallel rectangles. We give a reduction from the independent set problem on 3-regular planar graphs, which is known to be NP-Complete [GJ79b].

Rim et al. [RN95] proved that MIS is NP-Hard for planar rectangle intersection graphs with degree at most 3. They also gave a reduction from the independent set problem on 3-regular planar graphs. Given a 3-regular planar graph $G = (V,E)$, they construct an instance $H = (V',E')$ of the MIS problem on rectangle intersection graphs. First we outline their construction of H from G. For any cubic planar graph G, it is always possible [RN95] to get a rectilinear planar embedding of G such that each vertex $v \in V$ is drawn as a point p_v, and each edge $e = (u,v) \in E$ is drawn as a rectilinear path, connecting the points p_u and p_v, having at most four bends, and thus consisting of at most five straight line segments. They [RN95] construct a family of rectangles B in the following way. For each point p_{v_i} where $v_i \in V$, a rectangle b_i is placed surrounding the point p_{v_i}. In each rectilinear path connecting p_{v_i} and p_{v_j}, they place six rectangles $b_{ij}^1, b_{ij}^2, \ldots, b_{ij}^6$ such that i) b_i intersects b_{ij}^1, ii) b_j intersects b_{ij}^6, iii) b_{ij}^k intersects b_{ij}^{k+1} for $k = 1,2,\ldots,5$ iv) $b_{ij}^1, b_{ij}^2, \ldots, b_{ij}^6$ do not intersect any other rectangles in B. For an illustration see Figure 5.7.

Clearly, $H(V',E')$ is an axis-parallel rectangle intersection graph with degree at most 3 where $|V'| = |V| + 6|E|$ and $|E'| = 7|E|$. In their reduction, the following lemma holds.

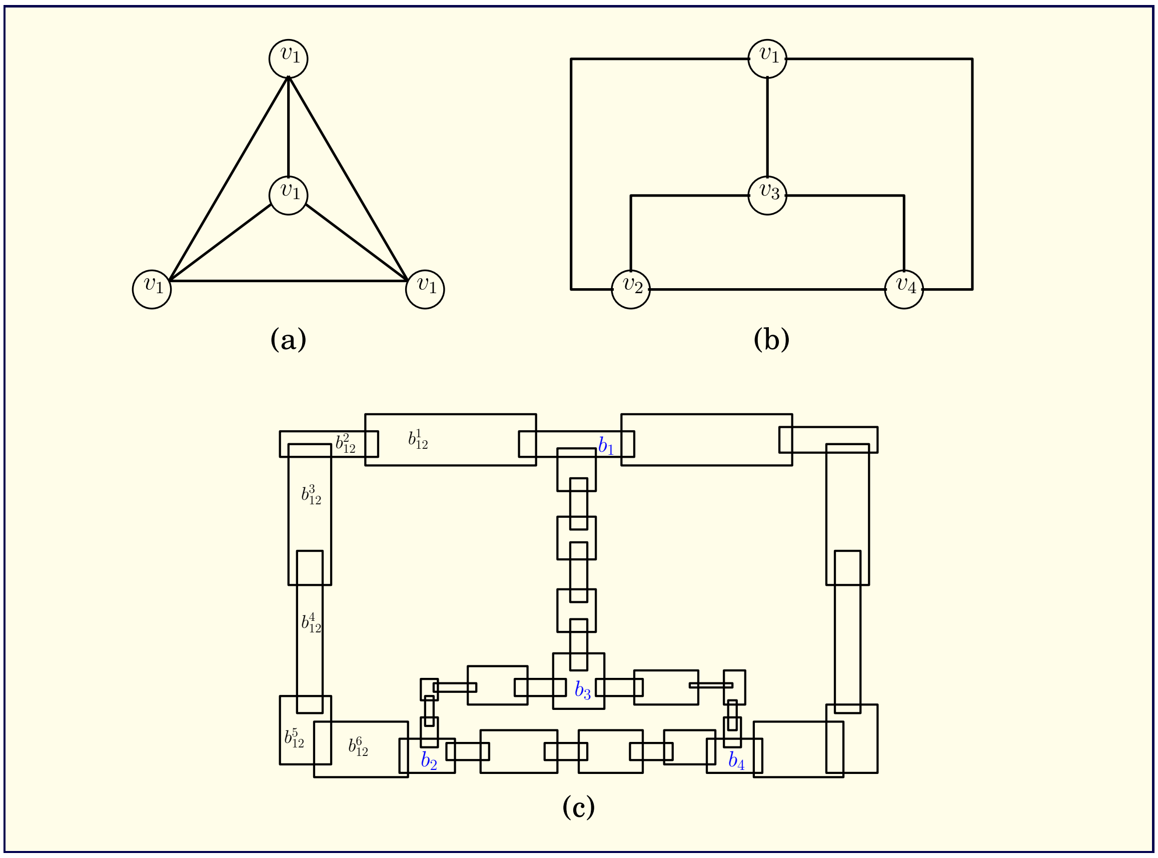


Figure 5.7: (a) A cubic planar graph G, (b) A rectilinear embedding of G, (c) Family B of rectangles.

Lemma 5.14. *[RN95] $G = (V, E)$ has an independent set of size $\geq m$ if and only if H has an independent set of size $m + 3|E|$.*

Given H, we construct an instance G_H of MTFS problem in axis-parallel rectangles intersection graphs. For the sake of understanding, let all rectangles corresponding to vertices in H, i.e., all rectangles in B, be colored black. To get G_H, we insert a family R of red rectangles in the following way. For each pair of adjacent rectangles b and b' in B, we place a red rectangle $R_{b,b'}$ such that i) $R_{b,b'}$ intersects both b and b', ii) $R_{b,b'}$ does not intersect any other rectangles in $B \cup R$. As per the construction of H, it is always possible to place such a rectangle for each pair of adjacent rectangles in B. See Figure 5.8 for an illustration of this transformation. This completes the construction of our instance of the MTFS problem on axis-parallel rectangles intersection graphs. Since R contains $7|E|$ rectangles, the above transformation can be done in polynomial time. Now G_H is an axis-parallel rectangle intersection graph with underlying geometric objects $B \cup R$. Clearly, the number of vertices in G_H is $(|V| + 13|E|)$. We now prove the following lemma.

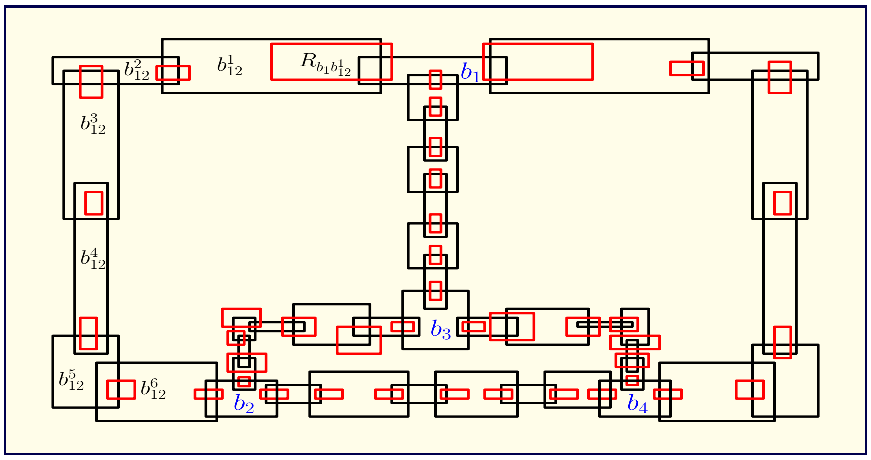


Figure 5.8: Construction of an instance of MTFS problem from B.

Lemma 5.15. *H has an independent set of size $\geq k$ if and only if G_H has a triangle-free subgraph on $\geq k+7|E|$ vertices.*

Proof. Let H have an independent set of k vertices. Let $B' \subseteq B$ be the set of rectangles corresponding to these k vertices. Note that the rectangles in R are independent. We take the subgraph H' of G_H induced by the vertices corresponding to the rectangles $B' \cup R$. Now H' is triangle-free. Because if H' is not triangle-free then there is an intersection between two black rectangles in B', which leads to a contradiction. As $|R| = 7|E|$, so the claim holds.

Now we show the other direction. Let G_H have a triangle-free subgraph H' on $\geq k+7|E|$ vertices. Let $X(H') = B_1 \cup R_1$ be the set of rectangles corresponding to the vertices of H', where $B_1 \subseteq B$ and $R_1 \subseteq R$. Also, let $R_2 \subseteq (R \setminus R_1)$ be the set of those red rectangles that has at most one adjacent rectangle in B_1. Then clearly the graph with underlying rectangles $B_1 \cup R_1 \cup R_2$ has no triangles. So each rectangle in $R \setminus (R_1 \cup R_2)$ has exactly two neighbours in B_1. Let $E(B_1)$ denote the set of edges in the subgraph induced by the vertices corresponding to the rectangles in B_1. Note that if there is a pair of adjacent rectangles b_i and b_j in B_1 then the rectangle $R_{i,j}$ should be part of R_3. It implies that $|E(B_1)| = |R_3|$ so $|E(B_1)| = |R| - |R_1 \cup R_2|$. Now $|B_1| + |R_1 \cup R_2| \geq k+7|E|$, so $|B_1| - (7|E| - |R_1 \cup R_2|) \geq k$. This implies $|B_1| - (|R| - |R_1 \cup R_2|) \geq k$, hence $|B_1| - |E(B_1)| \geq k$. Now in B_1 we repeatedly remove the rectangles from B_1 to get an independent set of rectangles. Finally, within at most $|E(B_1)|$ steps, we are left with a set of independent rectangles in B_1 with size $\geq k$. So there exists an independent set of size $\geq k$ in H. ■

By Lemma 5.14 and Lemma 5.15, we have the following.

Lemma 5.16. *$G = (V, E)$ has an independent set of size $\geq m$ if and only if G_H has a triangle-free subgraph on $m + 10|E|$ vertices.*

We can now conclude the following theorem.

Theorem 5.11. *The* MTFS *problem is* NP-Complete *on axis-parallel rectangle intersection graphs.*

Summary

In this chapter, we studied the problem of computing a maximum-size bipartite subgraph on geometric intersection graphs. We showed that the problem is NP-Hard on the geometric graphs for which the maximum independent set problem is NP-Hard. On the positive side, we gave polynomial-time algorithms for solving the problem almost optimally on circular-arc graphs. We also produce some approximation algorithms for the problem on unit squares, variants of unit disks, and unit-height rectangles.

CHAPTER 6

COVERING AND PACKING OF RECTILINEAR SUBDIVISION

Contents

Related Publications:

1. Satyabrata Jana, Supantha Pandit. "Covering and packing of rectilinear subdivision." *Theoretical Computer Science* 840 (2020): 166-176.

2. Satyabrata Jana, Supantha Pandit. "Covering and packing of rectilinear subdivision." In *International Workshop on Algorithms and Computation*, pp. 381-393. Springer, 2019

6.1 Introduction

The Set Cover and Independent Set problems are two well-studied problems in many fields. In the geometric Set Cover problem, we are given a set of points and a set of geometric objects such that their union contains the set of points, and the goal is to find a minimum cardinality collection of objects that covers all of the given set of points. In the Independent Set problem, we are given a set of objects and seek a maximum cardinality subset of objects that are pairwise non-intersecting. The Dominating Set problem is a variation of the Set Cover problem in which we are given a set of objects and seek a minimum cardinality subset of objects such that every object has a non-empty intersection with one of the chosen objects.

In this chapter, we study variations of the Set Cover, Independent Set, and Dominating Set problems. We are given m axis-parallel line segments that induce a planar subdivision $\mathscr{P}$ with a set F of n bounded rectilinear faces. Further, we consider each bounded face to be a closed region, i.e. including the boundary. We formally define these problems as follows.

A planar subdivision with a set F of n bounded faces in $\mathbb{R}^2$, is given as input.

- **(P1) Stabbing-Subdivision:** find a minimum cardinality set of points in the plane such that each face in F is stabbed (intersected) by one of the selected points.
- **(P2) Independent-Subdivision:** find a maximum cardinality subset $F' \subseteq F$ of faces such that any pair of faces in F' is non-intersecting.
- **(P3) Dominating-Subdivision:** find a minimum cardinality subset $F' \subseteq F$ of faces such that any face in $F \setminus F'$ has a non-empty intersection with a face in F'.

A special case of the Stabbing-Subdivision problem has an application to the art gallery problem [CRCS+94]. Suppose a rectangular art gallery is given. The gallery is subdivided into rectangular rooms. The art gallery problem seeks to find the fewest guards (points) so that every room (face) is protected (stabbed) by a guard point. This problem is precisely the Stabbing-Subdivision problem in which the input faces are all rectangular. More generally, we consider the case of rectilinear rooms (the original input of the Stabbing-Subdivision problem), not just rectangular rooms, and ask the same question, to find the fewest guards to protect all of the rectilinear rooms. We say a guard protects a rectilinear room if and only if guard belongs to the room.

In this chapter, we sometimes use the term “rectangle” and “rectangular face” (of a subdivision) interchangeably.

6.1.1 Our contributions

We show that each of the problems of STABBING-SUBDIVISION, INDEPENDENT-SUBDIVISION, and DOMINATING-SUBDIVISION is NP-Hard. We even prove that these problems remains NP-Hard when we concentrate only on the rectangular faces of the subdivision. In addition, we provide a 2.083-approximation and a PTAS for the STABBING-SUBDIVISION problem.

6.2 STABBING-SUBDIVISION

6.2.1 NP-hardness

We first prove that the STABBING-SUBDIVISION problem is NP-Hard when we are only required to stab the rectangular faces of the subdivision. Next, we modify the construction to show that the STABBING-SUBDIVISION problem is NP-Hard. We give a reduction from the Rectilinear Planar 3SAT (RP3SAT) Problem. Lichtenstein [Lic82] proved that the Planar 3SAT problem is NP-Complete. Later, Knuth and Raghunathan [KR92] showed that every Planar 3SAT problem can be expressed as an RP3SAT problem. We define the RP3SAT problem as follows. We are given a *3-SAT* formula ϕ with n variables $x_1, x_2, \dots, x_n$ and m clauses $C_1, C_2, \dots, C_m$ where each clause contains exactly 3 literals. For each variable or clause, we take a rectangle. The variable rectangles are placed on a horizontal line such that no two of them intersect each other. The clause rectangles are placed above and below this horizontal line such that they together form a nested structure. The clause rectangles connect to the variable rectangles by vertical lines such that no two lines intersect. The objective is to decide whether there is a truth assignment to the variables that satisfies ϕ. See Figure 6.1 for an instance of the RP3SAT problem.

Variable gadget: The gadget of x_i consists of $8m+4$ vertical and 4 horizontal line segments. See Figure 6.2 for the construction of the gadget. The 4 segments v_1, v_4, h_1, and h_4 together form a rectangular region R. Next, the 2 vertical segments v_2 and v_3 partition R vertically into 3 rectangles R_1, R_2, and R_3. Further, two horizontal segments h_2 and h_3 partition R_2 horizontally into three rectangles R_4, R_5, and R_6. Fi-

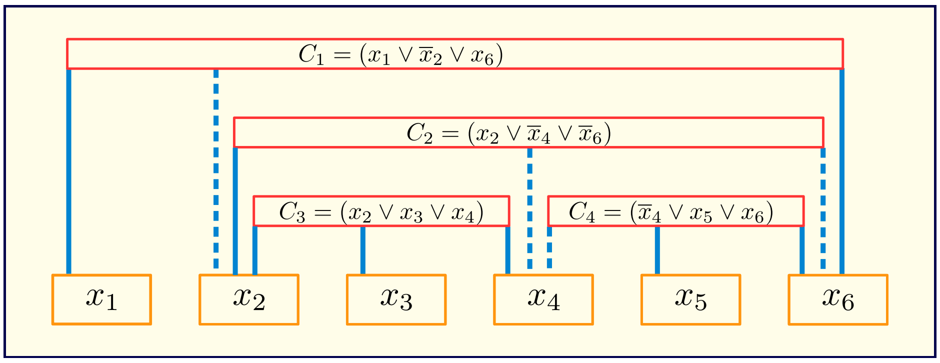

$C_1 = (x_1 \vee \overline{x}_2 \vee x_6)$
$C_2 = (x_2 \vee \overline{x}_4 \vee \overline{x}_6)$
$C_3 = (x_2 \vee x_3 \vee x_4)$
$C_4 = (\overline{x}_4 \vee x_5 \vee x_6)$
x_1
x_2
x_3
x_4
x_5
x_6

Figure 6.1: An instance of the RP3SAT problem. We only show the clauses that connect to the variables from above. The solid (resp. dotted) lines represent that the variable is positively (resp. negatively) present in the corresponding clauses.

nally, the $4m$ vertical segments $\ell_1, \ell_2, \ldots, \ell_{4m}$ partition R_4 vertically into $4m+1$ small rectangles $r_1, r_2, \ldots, r_{4m+1}$. Similarly, the $4m$ vertical segments $\ell_{4m+1}, \ell_{4m+2}, \ldots, \ell_{8m}$ partition R_6 vertically into $4m+1$ small rectangles $r_{4m+2}, r_{4m+3}, \ldots, r_{8m+2}$. Finally we have the total of $8m+5$ rectangles $R_1, R_3, R_5, r_1, r_2, \ldots, r_{8m+2}$ inside R. Clearly, these rectangles except R_5 form a cycle of size $8m+4$. Observe that any point along the cycle can stab at most two consecutive regions. Therefore, all solutions can be represented, canonically, by exactly one of the two optimal solutions $P_1^i = \{p_1, p_3, \ldots, p_{8m+3}\}$ and $P_2^i = \{p_2, p_4, \ldots, p_{8m+4}\}$ each of size $4m+2$ (Note that these points are not as a part of the input, they are one set of canonical points.). These two canonical solutions are corresponding to the truth value of the variable x_i.

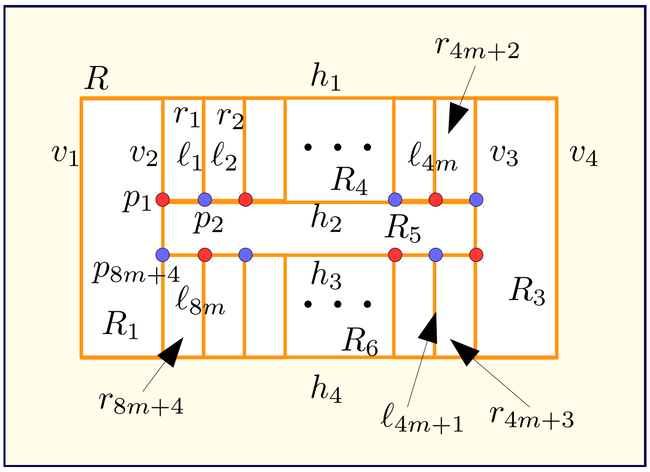


Figure 6.2: Structure of a variable gadget.

Clause gadget: The gadget for clause C_α consists of a single rectangle r_α that is formed by four line segments. The rectangle r_α can be interpreted as the same rectangle as C_α in the RP3SAT problem instance.

Interaction: Now we describe how the clause gadgets interact with the variable gadgets. Observe that the description for the clauses which connect to the variables from above are independent with the clauses which connect to the variables from below. Therefore, we only describe the construction for the clauses which connect to the variables from above. Let $C^i_1, C^i_2, \dots, C^i_\tau$ be the left to right order of the clauses which connect to x_i from above. Then we say that C^i_k is the k^{th} clause for x_i. For example, C_3, C_2, and C_4 are the 1st, 2nd, and 3rd clause for the variable x_4 in Figure 6.1. Let C_α be a clause containing the variable x_i, x_j, x_t. We say that clause C_α is the k_1, k_2, and ${k_3}^{\text{th}}$ clause for variable x_i, x_j, and x_t, respectively based on the above ordering. For example, C_3 is the 3rd, 1st, and 1st clause for variable x_2, x_3, and x_4, respectively in Figure 6.1. Let r_α be the rectangle corresponding to C_α. Now we have the following cases.

- If x_i appears as a positive literal in C_α, then extend the 3 segments ℓ_{4k_1-3}, ℓ_{4k_1-2}, and ℓ_{4k_1-1} vertically upward such that it touches the bottom boundary of r_α. Move p_{4k_1-1} vertically upward to the bottom boundary of r_α.
- If x_i appears as a negative literal in C_α, then extend the 3 segments ℓ_{4k_1-2}, ℓ_{4k_1-1}, and ℓ_{4k_1} vertically upward such that it touches the bottom boundary of r_α. Move p_{4k_1} vertically upward to the bottom boundary of r_α.

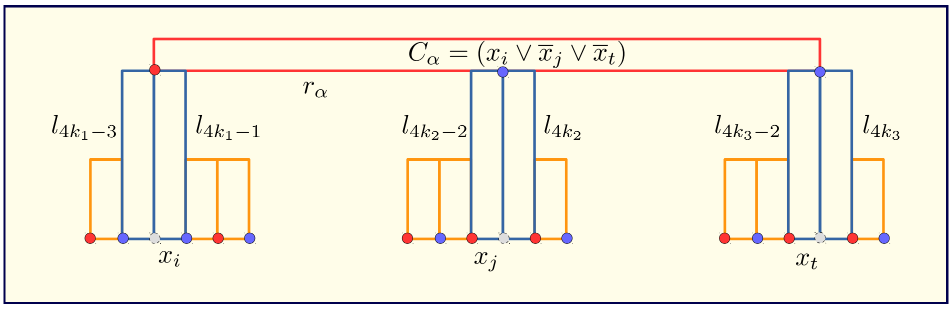


Figure 6.3: Variable clause interaction.

The similar constructions can be done for x_j and x_t by replacing k_1 with k_2 and k_3, respectively. The construction for the clause C_α is shown in Figure 6.3. Note that, we break the horizontal segment h_1 in the variable gadgets into smaller intervals and shifted the intervals vertically along with the extension of the vertical lines. We also demonstrate the complete construction in Figure 6.4 for the formula shown in Figure 6.1. This completes the construction and clearly, it can be done in polynomial time.

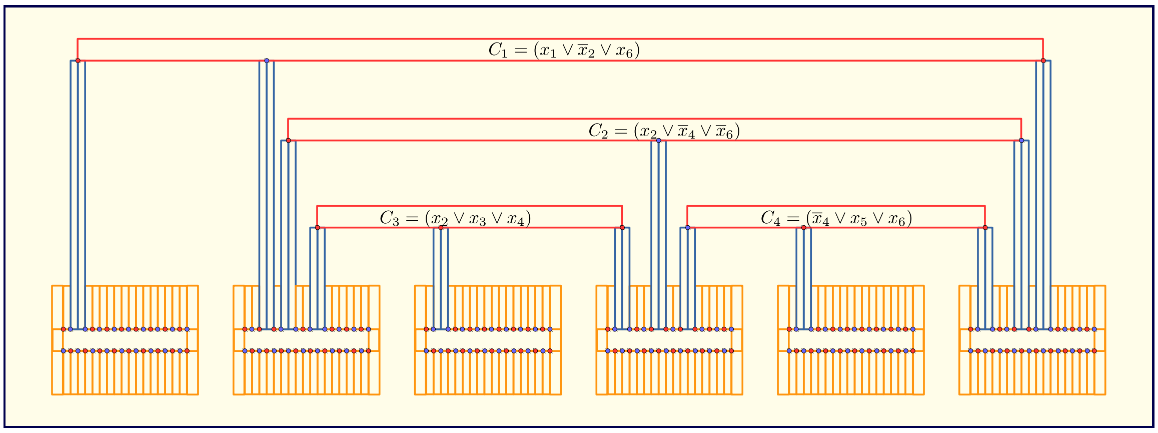

$C_1 = (x_1 \vee \overline{x}_2 \vee x_6)$
$C_2 = (x_2 \vee \overline{x}_4 \vee \overline{x}_6)$
$C_3 = (x_2 \vee x_3 \vee x_4)$
$C_4 = (\overline{x}_4 \vee x_5 \vee x_6)$

Figure 6.4: Complete construction for the formula in Figure 6.1.

Correctness: The correctness follows from the following lemma.

Lemma 6.1. *ϕ is satisfiable if and only if there is a solution to the* Stabbing-Subdivision *problem to stab only rectangular faces with $n(4m+2)$ points.*

Proof. Assume that ϕ is satisfiable, i.e., we have a truth assignment of the variables in ϕ. Now consider a variable x_i. If x_i is true, we select the set P_1^i, otherwise we select the set P_2^i. Clearly, the $n(4m+2)$ selected points corresponding to all variable gadgets stab all the rectangular faces of the construction.

On the other hand, assume that the Stabbing-Subdivision problem has a solution with $n(4m+2)$ points. Observe that at least $(4m+2)$ points are needed to stab all the faces of a variable gadget. Since the rectangular faces of variable gadgets are disjoint from each other, exactly $(4m+2)$ points must be selected from each variable gadget. Now there are exactly two canonical solutions of size $(4m+2)$, either P_1^i or P_2^i. Therefore, we set variable x_i to be true if P_1^i is selected from the gadget of x_i, otherwise we set x_i to be false. Note that for each clause C_α the six faces corresponding to three literals it contains, touches the rectangle r_α. Since r_α is stabbed, at least one of the selected points must be chosen in the solution. Such a point is either in one of the sets P_1^i or P_2^i of the corresponding variable gadget based on whether the variable is positively or negatively present in that clause. Hence, the above assignment is a satisfying assignment. ■

Theorem 6.1. *The* Stabbing-Subdivision *problem is* NP-Hard *for stabbing only rectangular faces of a subdivision.*

The Stabbing-Subdivision problem for stabbing all rectilinear faces:

We now prove that the STABBING-SUBDIVISION problem (i.e., stabbing all rectilinear faces of a subdivision) is NP-Hard. Note that, after embedding the gadgets (i.e., the complete construction) on the plane, the subdivision creates three types of faces, (i) variable faces: the faces that are interior to the variable gadgets (ii) clause faces: the faces that are associated with the clause gadgets, and (iii) outer faces: the faces other than types (i) and (ii). See Figure 6.5 for these types of faces. Observe that the variable and clause faces are all rectangular. The only non-rectangular faces are the outer faces. We prove that Figure 6.1 is true even if we consider the outer faces as well. Notice that each outer face shares a boundary with a variable face on either h_1 (coincides with a top boundary of a variable face), or h_4 (coincides with a bottom boundary of a variable face), or both h_1 and h_4. See Figure 6.5 for an illustration of this fact.

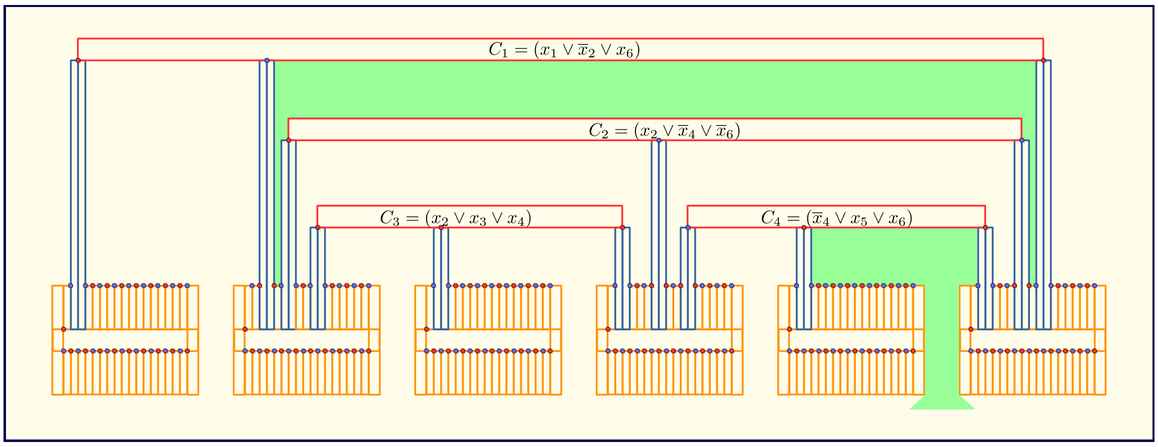

$C_1 = (x_1 \vee \overline{x}_2 \vee x_6)$
$C_2 = (x_2 \vee \overline{x}_4 \vee \overline{x}_6)$
$C_3 = (x_2 \vee x_3 \vee x_4)$
$C_4 = (\overline{x}_4 \vee x_5 \vee x_6)$

Figure 6.5: Complete construction of the STABBING-SUBDIVISION problem for the formula in Figure 6.1. Only two outer faces are highlighted in green color.

Note that, in the proof of the Lemma 6.1, we assume that the canonical points are on the lines h_2 and h_3 (see Figure 6.2). At a later stage some of them are moved vertically to stab clause rectangles. To get our result for the STABBING-SUBDIVISION problem, we assume a different position of the canonical point-set in the variable gadgets. In each variable gadget, to stab this rectangle R_5, we keep only one point either on h_2 (say p_1) or on h_3 (say point $p_{8m}+4$ and we shift the remaining points vertically upward from h_2 to h_1 and vertically downward from h_3 to h_4. Since each outer face shares a boundary with a variable face either top (coincides with line the h_1) or bottom (coincides with the line h_4), or both boundaries. This implies that each clause face is stabbed by at least two *consecutive* canonical points either on h_1 or on h_4 (see Figure 6.5) . Now to stab the variable rectangles we choose either P_1^i or P_2^i from the i-th variable gadget. The chosen solution is also stabs the clause gadgets. Therefore the

Lemma 6.1 is true even when we are intended to stab all the faces (rectilinear) of the subdivision. Finally we have the following theorem.

Theorem 6.2. *The* STABBING-SUBDIVISION *problem is* NP-Hard.

6.2.2 Approximation algorithms

6.2.2.1 Factor 2.083 approximation

We are given m axis-parallel line segments that induce a planar subdivision $\mathscr{P}$ with a set F of n bounded rectilinear faces. To provide the approximation algorithm, we transform any instance of the STABBING-SUBDIVISION problem into an instance of the Set Cover problem where the size of each set is at most 4. Observe that, there exists an optimal solution to the STABBING-SUBDIVISION problem that only contains vertices of $\mathscr{P}$ (we can call them corner points of F). Also, any corner point of F can stab at most 4 rectilinear faces in $\mathscr{P}$.

We now create an instance of the Set Cover problem as follows. The set of elements is the set of all faces and the collection is all sets of faces corresponding to the corner points of F. Note that each set in the collection is of size at most 4, since any corner point can stab at most 4 faces. This Set Cover instance admits a 2.083 (H_4, i.e., the harmonic series sum of the first 4 terms) factor approximation (see Exercise 2.8 on page 24 of [Vaz01]). Hence we have the following theorem.

Theorem 6.3. *There exists a* 2.083 *factor approximation algorithm for* STABBING-SUBDIVISION *problem in a planar subdivision by rectilinear line segments.*

6.2.3 PTAS via local search

In this section, we show that a local search framework [MR10] leads to a PTAS for the STABBING-SUBDIVISION problem. We are given a planar subdivision with a set F of n bounded faces. Note that, we can choose points only from the vertex set V of the subdivision. Therefore, $\mathscr{R} = (V, F)$ be the given range space. Clearly V is a feasible solution to the STABBING-SUBDIVISION problem. We apply the k-level local search [MR10] (k is a given parameter) as follows.

1. Let X be some feasible solution to the STABBING-SUBDIVISION problem (initially take X as V).

2. Do the following:

a) Search for $X' \subseteq X$ and Y such that $|Y| \subseteq V$, $|Y| < |X'| \leqslant k$ and $(X \setminus X') \cup Y$ is a feasible solution.

b) If such X' and Y exist, update X with $(X \setminus X') \cup Y$ and repeat the above step. Otherwise, return X and stop.

It is easy to see that, for a fixed k, the running time of the algorithm is polynomial in n. Further, the local search algorithm always returns a local optimum solution. A feasible solution X is said to be a local optimum if no X' exists in Step 2(a) in the above algorithm. We show that given any $\epsilon > 0$, a $O(1/\epsilon^2)$-level local search returns a hitting set of size at most $(1+\epsilon)$ times an optimal hitting set for $\mathscr{R}$.

Locality condition ([MR10]): A range space $\mathscr{R} = (V, F)$ satisfies the locality condition if for any two disjoint subsets $R, B \subseteq V$, it is possible to construct a planar bipartite graph $G = (R \cup B, E)$ with all edges going between R and B such that for any $f \in F$, there exist two vertices $u \in f \cap R$ and $v \in f \cap B$ such that edge $(u,v) \in E$.

Theorem 6.4. *[MR10] Let $\mathscr{R} = (V, F)$ be a range space satisfying the locality condition. Let $R \subseteq V$ be an optimal hitting set for F, and $B \subseteq V$ be the hitting set returned by a k-level local search. Furthermore, assume $R \cap B = \emptyset$. Then there exists a planar bipartite graph $G = (R \cup B, E)$ such that for every subset $B' \subseteq B$ of size at most k, $|N_G(B')| \geq |B'|$ where $N_G(W)$ denotes the set of all neighbours of the vertices of W in G.*

The following lemma implies that given any $\epsilon > 0$, a k-level local search with $\epsilon = \dfrac{c}{\sqrt{k}}$ gives a $(1+\epsilon)$-approximation for the STABBING-SUBDIVISION problem.

Lemma 6.2. *[MR10] Let $G = (R \cup B, E)$ be a bipartite planar graph on red and blue vertex sets R and B, $|R| \geq 2$, such that for every subset $B' \subseteq B$ of size at most k, where k is a large enough number, $|N_G(B')| \geq |B'|$. Then $|B| \leq (1 + \dfrac{c}{\sqrt{k}})|R|$, where c is a constant.*

PTAS **for the STABBING-SUBDIVISION problem:** Let R (red) and B (blue) be disjoint subsets of the vertices in planar subdivision $\mathscr{P}$ where R and B be an optimum solution and the solution returned by the k-level local search, respectively. For simplicity, we assume that $R \cap B = \emptyset$. Otherwise, we can remove the common elements from each of R and B, and then do a similar analysis. As we remove the same number of elements from both R and B, the approximation ratio of the original instance is at most the approximation ratio of the restricted one. We construct the required graph G on the vertices $R \cup B$ in the following way. Since R and B are feasible solutions of the STABBING-SUBDIVISION problem, every face $f \in F$ must contain at least one red and one blue point. We simply join exactly one pair of red and blue points by an edge for each face $f \in F$. Clearly, the edge for a face $f \in F$ lies completely inside f. Therefore G becomes a planar bipartite graph and hence $\mathscr{R}$ satisfies the locality condition. Therefore, from Theorem 6.4 and Lemma 6.2, we say that the STABBING-SUBDIVISION problem admits a PTAS.

6.3 INDEPENDENT-SUBDIVISION

In this section, we prove that the INDEPENDENT-SUBDIVISION problem is NP-Hard by giving a reduction from the RP3SAT problem. The reduction follows the same line of the reduction presented in Section 6.2. We construct an instance I of the INDEPENDENT-SUBDIVISION problem from an instance ϕ of the RP3SAT problem and prove that the construction is correct. Here our main result is that it remains NP-Hard when considering the rectangular faces only.

Variable gadget: The variable gadget is similar to the variable gadget that is described in the Section 6.2. See Figure 6.6 for the construction of a variable gadget. The difference of this variable gadget from the gadget in the Section 6.2 is that we partition R_4 into $4m-2$ smaller rectangles $r_1, r_2, \ldots, r_{4m-2}$ and R_6 into $4m-2$ smaller rectangles $r_{4m-1}, r_{4m}, \ldots, r_{8m-4}$. Finally, we have the total of $8m-1$ rectangles $R_1, R_3, R_5, r_1, r_2, \ldots, r_{8m-4}$ inside R. Notice that, these rectangles except R_5 form a cycle of size $8m-2$. Therefore, all solutions can be represented, canonically, by exactly one of the two optimal solutions $S_1^i = \{R_3, r_1, r_3, \ldots, r_{4m-3}, r_{4m}, r_{4m+2}, \ldots, r_{8m-4}\}$ and $S_2^i = \{R_1, r_2, r_4, \ldots, r_{4m-2}, r_{4m-1}, r_{4m+1}, \ldots, r_{8m-5}\}$, each with size $4m-1$. These two canonical solutions are corresponding to the truth values of the variable x_i.

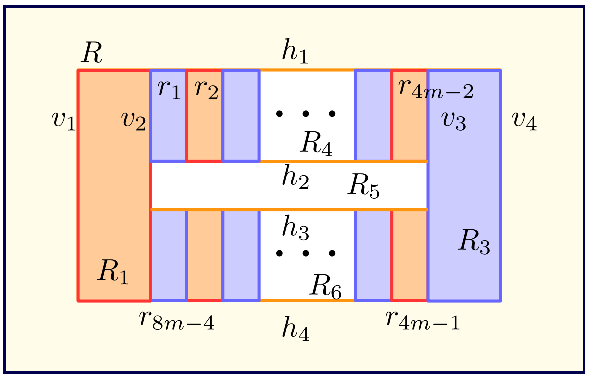


Figure 6.6: Structure of a variable gadget.

Clause gadget: The gadget of the clause C_α includes 9 rectangles $r_\alpha^1, r_\alpha^2, \ldots, r_\alpha^9$ (see green rectangles in Figure 6.7. The six rectangles $r_\alpha^4, r_\alpha^5, \ldots, r_\alpha^9$ are placed inside the rectangle of C_α in the RP3SAT problem instance and the other three rectangles $r^1_\alpha, r^2_\alpha, r^3_\alpha$ are corresponding to the three vertical legs between C_α and the three variables it contains. Note that there is another rectangle present in the clause gadget bounded by the above 9 rectangles. However, this rectangle has no effect in the reduction, since picking this rectangle makes other 9 rectangles invalid (cannot be selected).

Interaction: Here also we describe the construction for the clauses that connect to the variables from above, since the construction is similar and independent from the clauses that connect to the variables from below. Let C_α be a clause containing the variables x_i, x_j, x_t. Also assume that this is the left to right order of these variables in which they appear in ϕ. Using the similar way as before (Section 6.2), we say that the clause C_α is the k_1, k_2, and k_3th clause for the variables x_i, x_j, and x_t, respectively.

- If x_i appears as a positive literal in the clause C_α, then attach the rectangle r^1_α to the rectangle r_{4k_1-3}.
- If x_i appears as a negative literal in clause C_α, then attach the rectangle r^1_α to the rectangle r_{4k_1-2}.

The similar constructions can be done for x_j by replacing r^1_α and k_1 with r^2_α and k_2, respectively and for x_t by replacing r^1_α and k_1 with r^3_α and k_3, respectively. The construction for the clause C_α is depicted in Figure 6.7. The complete construction for the formula in Figure 6.1 is shown in Figure 6.8. Clearly, the construction can be done in polynomial time. We now prove the correctness of the construction.

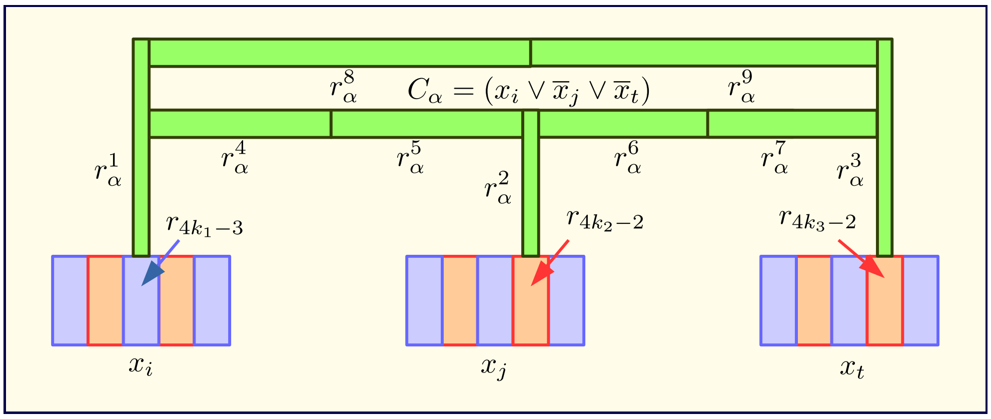


Figure 6.7: Variable clause interaction.

Lemma 6.3. *ϕ is satisfiable if and only if there is a solution of size $n(4m-1)+4m$ to* INDEPENDENT-SUBDIVISION *problem while considering only rectangular faces.*

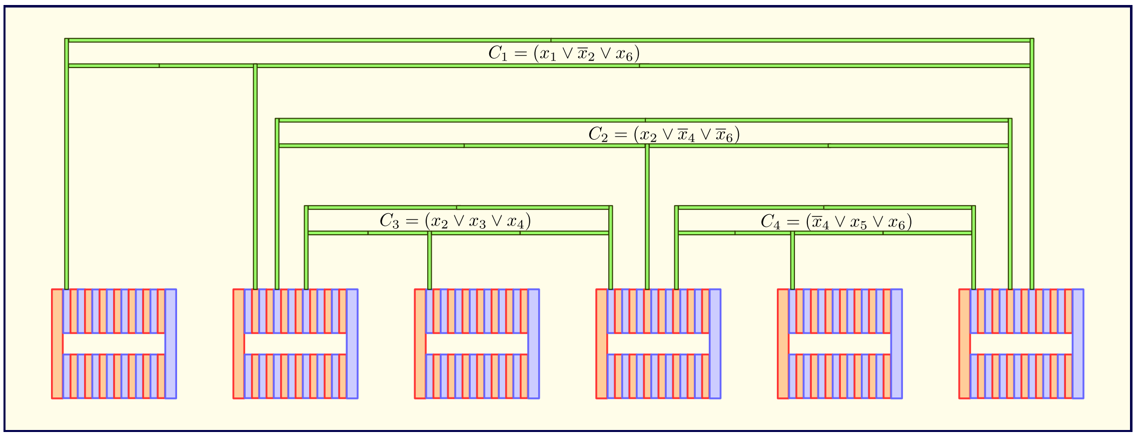

$C_1 = (x_1 \vee \overline{x}_2 \vee x_6)$
$C_2 = (x_2 \vee \overline{x}_4 \vee \overline{x}_6)$
$C_3 = (x_2 \vee x_3 \vee x_4)$
$C_4 = (\overline{x}_4 \vee x_5 \vee x_6)$

Figure 6.8: Complete construction for the formula in Figure 6.1.

Proof. Assume that ϕ has a satisfying assignment. For the variable x_i, if x_i is true, select the set S_2^i, otherwise select the set S_1^i. Since each set is of cardinality $(4m-1)$, clearly we select $n(4m-1)$ independent rectangles across all variable gadgets. Now let C_α be a clause containing variables x_i, x_j, x_t. Since C_α is satisfiable at least one of the three rectangles $r_\alpha^1, r_\alpha^2, r_\alpha^3$ is free to choose in a solution. This implies we can select exactly 4 rectangles from the gadget of C_α. We can picked 4 rectangles independently from each clause gadget. Hence, in total we can select $n(4m-1)+4m$ rectangles.

On the other hand, assume that the INDEPENDENT-SUBDIVISION problem has a solution S with $n(4m-1)+4m$ rectangles. Note that for each variable gadget the size of an optimal independent set is $(4m-1)$, either the set S_1^i or S_2^i. We set the variable x_i to be true if S_2^i is selected from the gadget of x_i, otherwise we set x_i to be false. Now we have to show that this assignment is a satisfying assignment for ϕ, i.e., each clause of ϕ is satisfied. Since the variable gadgets are independent, there are at most $n(4m-1)$ rectangles from the variable gadgets belongs to S. Also since the size of the solution is $n(4m-1)+4m$, from each clause gadget exactly 4 rectangles are is in S. Let C_α be a clause containing variables x_i, x_j, x_t. As there are 4 independent rectangles from the set $\{r_\alpha^1, r_\alpha^2, \ldots, r_\alpha^9\}$, so one must be from the set $\{r_\alpha^1, r_\alpha^2, r_\alpha^3\}$ that is in the given solution. Without loss of generality, let r_α^1 be present, then surely x_i is a true variable as our assignment. Hence the above assignment is a satisfying assignment. ■

Theorem 6.5. *The* INDEPENDENT-SUBDIVISION *problem is* NP-Hard *by considering only rectangular faces of a subdivision.*

The INDEPENDENT-SUBDIVISION problem for all rectilinear faces:

We now prove that it is also NP-Hard to find a maximum independent set of rectilinear faces in a planar subdivision. Similar to the STABBING-SUBDIVISION problem, after embedding the construction on the plane, the subdivision creates three types of faces, (i) variable faces: the faces that are interior to the variable gadgets (note that all variable faces are rectangular), (ii) clause faces: the faces that are associated with the clause gadgets (each clause gadget consists of nine rectangles), and (iii) outer faces: any other faces that are not included in any of (i) and (ii). We now show that with the presence of outer faces also, Lemma 6.3 remains true.

Visualize that we are attaching each clause gadget one by one with the variable gadgets from inner level to outer level (i.e., first C_3, then, C_4, then C_2, and then finally C_1 in Figure 6.8). Then each clause gadget creates two additional rectilinear faces (outer faces), on both sides of the rectangle corresponding to the middle leg (see Figure 6.9). Note that, each such face is adjacent with at least 4 clause rectangles from a single clause gadget and at least 4 variable rectangles, 2 consecutive rectangles from 2 variable gadgets each. Therefore, picking one of these new faces to the optimal solution makes the solution size strictly less than the original. Therefore, even if we consider all rectilinear faces, Lemma 6.3 holds and so Theorem 6.5. Therefore we have the following theorem.

Theorem 6.6. *The* INDEPENDENT-SUBDIVISION *problem is* NP-Hard.

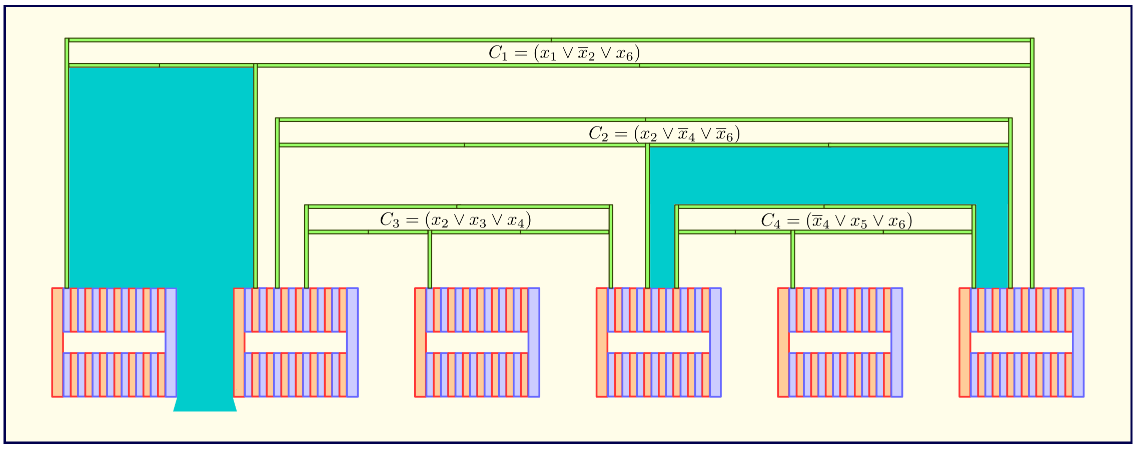

$C_1 = (x_1 \vee \overline{x}_2 \vee x_6)$
$C_2 = (x_2 \vee \overline{x}_4 \vee \overline{x}_6)$
$C_3 = (x_2 \vee x_3 \vee x_4)$
$C_4 = (\overline{x}_4 \vee x_5 \vee x_6)$

Figure 6.9: Complete construction of the INDEPENDENT-SUBDIVISION problem for the formula in Figure 6.1. Only two outer faces are highlighted.

6.4 DOMINATING-SUBDIVISION

In this section, we prove that the DOMINATING-SUBDIVISION problem is NP-Hard. We give a reduction from the RP3SAT problem similar to Section 6.3. Here also our main result is that it remains NP-Hard when considering the rectangular faces only.

We construct an instance I of the DOMINATING-SUBDIVISION problem from an instance ϕ of the RP3SAT problem and prove that the construction is correct.

Variable gadget: Variable gadgets are similar to the variable gadgets that are described in Section 6.2. The difference between this variable gadget and that of in Section 6.2 is as follows. We partition R_4 into $3m+1$ small rectangles $r_1, r_2, \ldots, r_{3m+1}$ and R_6 into $3m+1$ small rectangles $r_{3m+4}, r_{3m+5}, \ldots, r_{6m+4}$. We partition R_1 into two rectangles r_{6m+6}, r_{6m+5} and R_3 into r_{3m+2}, r_{3m+3}. Next we take $2m+2$ mutually independent rectangles $s_1, s_2, \ldots, s_{2m+2}$ inside R_5 such that rectangle s_i touches the two regions r_{3i-2} and r_{3i-1}, for $1 \le i \le 2m+2$. Finally we have a total of $8m+8$ rectangles $r_1, r_2, \ldots, r_{6m+6}, s_1, s_2, \ldots, s_{2m+2}$ inside R. Figure 6.10 illustrate the construction of a variable gadget just described.

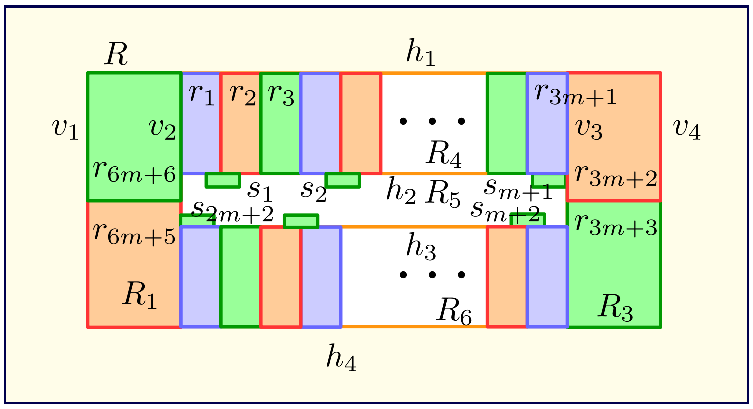


Figure 6.10: Structure of a variable gadget.

Lemma 6.4. *There exists exactly two optimal dominating sets of rectangles,* $D_1^i = \{r_1, r_4, \ldots, r_{6m+4}\}$ *and* $D_2^i = \{r_2, r_5, \ldots, r_{6m+5}\}$*, for the gadget of* x_i*.*

Proof. There is no rectangle that can dominate more than 4 rectangles. Since there are in total $(8m+8)$ rectangles, any dominating set cannot have size less than $(2m+2)$. Further, both D_1^i and D_2^i, each of size $(2m+2)$, dominate all the faces of the subdivision and hence they are optimal solutions. Now we show that there is no other optimal solution.

Clearly, no rectangle of the form r_{3k} or s_k where $1 \le k \le (2m+2)$ can be a part of an optimal solution, since each of them dominates exactly 3 rectangles. As a result, any optimal solution contains only rectangles of the form r_{3k-1} or r_{3k-2}, for $1 \le k \le (2m+2)$. Also, two rectangles, one of the form r_{3k-1} and other of the form r_{3k-2}, together cannot be a part of an optimal solution. ■

Clause gadget: The gadget for the clause C_α is a rectangle r_α (Figure 6.11).

Interaction: Here we describe the construction for the clauses that connect to the variables from above. A similar construction can be done for the clauses that connect to the variables from below. As before, we interpret C_α that contains variables x_i, x_j, and x_t as the k_1, k_2, and k_3th clause for the variables x_i, x_j, and x_t, respectively.

- If x_i appears as a positive literal in the clause C_α, then we extend the rectangle r_{3k_1-1} vertically upward such that it touches the rectangle r_α.
- If x_i appears as a negative literal in the clause C_α, then we extend the rectangle r_{3k_1-2} vertically upward such that it touches the rectangle r_α.

We make the similar constructions for x_j and x_t by replacing k_1 with k_2 and k_3, respectively. The construction for the clause C_α is depicted in Figure 6.11. The complete construction for the formula in Figure 6.1 is shown in Figure 6.12. Clearly, the construction can be done in polynomial time. We now prove the correctness.

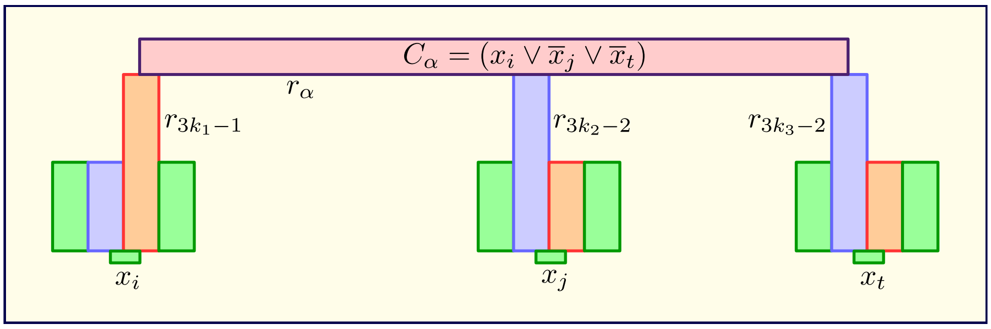


Figure 6.11: Variable clause interaction.

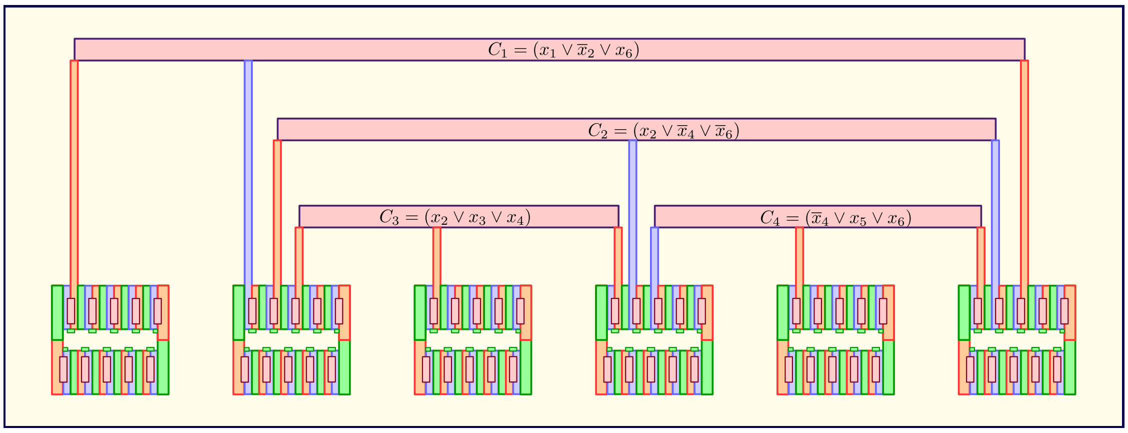

$C_1 = (x_1 \vee \overline{x}_2 \vee x_6)$
$C_2 = (x_2 \vee \overline{x}_4 \vee \overline{x}_6)$
$C_3 = (x_2 \vee x_3 \vee x_4)$
$C_4 = (\overline{x}_4 \vee x_5 \vee x_6)$

Figure 6.12: Complete construction of for the formula in Figure 6.1.

Lemma 6.5. *ϕ is satisfiable if and only if there is a solution of size $n(2m+2)$ to the* DOMINATING-SUBDIVISION *problem while considering only rectangular faces.*

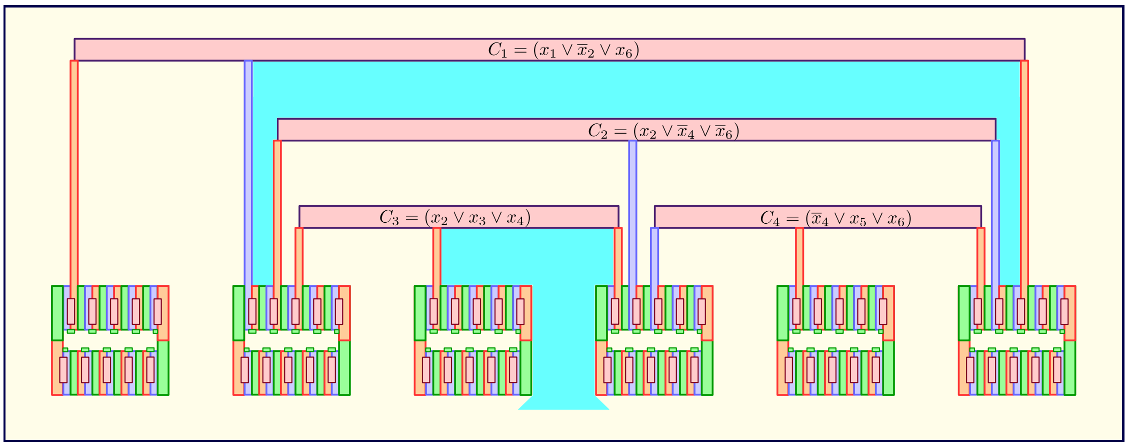
Proof. Assume that ϕ is satisfiable, i.e., we have a truth assignment to the variables of ϕ. For the variable x_i, if x_i is true we select the set D_2^i, otherwise we select the set D_1^i. Clearly, the $n(2m+2)$ selected rectangles corresponding to all the variable gadgets dominate all the rectangular faces of the subdivision.

On the other hand, assume that the DOMINATING-SUBDIVISION problem has a solution with $n(2m+2)$ rectangles. Observe that at least $(2m+2)$ rectangles are needed to dominate all the rectangular faces of a variable gadget. Since the rectangular faces of variable gadgets are disjoint from each other and the size of the solution is $n(2m+2)$, from each variable gadget exactly $(2m+2)$ rectangles must be selected. Therefore, we set variable x_i to be true if D_2^i is selected from the gadget of x_i, otherwise we set x_i to be false. Note that for each clause C_α the three rectangles corresponding to the three literals it contains attach to the rectangle r_α. Since r_α is dominated, at least one of these three rectangles is chosen in the solution. Such a rectangle is either in D_2^i or D_1^i of the corresponding variable gadget based on whether the variable is positively or negatively present in that clause. Hence, the above assignment is a satisfying assignment. ■

Theorem 6.7. *The* DOMINATING-SUBDIVISION *problem is* NP-Hard *when we are constrained to dominate all the rectangular faces of a subdivision.*

The DOMINATING-SUBDIVISION problem for all rectilinear faces:

We prove that the DOMINATING-SUBDIVISION problem is NP-Hard. Notice that, in a variable gadget, the rectilinear face R_5 dominates all the faces of that gadget. Hence the size of the optimal solution is 1. So we modify each variable gadget so that there are exactly two optimal solutions D_1^i and D_2^i of size $(2m+2)$ each while considering all (rectilinear) faces. Now if we embed the complete construction on the plane, similar to STABBING-SUBDIVISION problem, the subdivision creates three types of faces, (i) variable faces: the faces that are interior to the variable gadgets, (ii) clause faces: the faces that are associated with the clause gadgets, and (iii) outer faces: the faces other than types (i) and (ii). See Figure 6.13 for an illustration of these three types of faces. We keep the rest of the construction (clause gadget and the interaction) and proofs (Lemma 6.5) unaltered except we now show that using the new variable gadget (described later) the outer faces are also dominated by the faces chosen from the variable gadgets (i.e., the proof of Lemma 6.5 considering all three types of faces).



$C_1 = (x_1 \vee \overline{x}_2 \vee x_6)$
$C_2 = (x_2 \vee \overline{x}_4 \vee \overline{x}_6)$
$C_3 = (x_2 \vee x_3 \vee x_4)$
$C_4 = (\overline{x}_4 \vee x_5 \vee x_6)$

Figure 6.13: Complete construction of the DOMINATING-SUBDIVISION problem for the formula in Figure 6.1. Only two outer faces are highlighted.

Modified variable gadget: We take $2m+2$ rectangles $b_1, b_2, \ldots, b_{2m+2}$. We place the rectangle b_i in between the rectangles r_{3i-2} and r_{3i-1}, for $1 \leq i \leq 2m+2$ of the variable gadget shown in Figure 6.10 (see Figure 6.14). Now if we choose R_5 in the optimal solution, then we need $(2m+3)$ rectilinear faces to dominate all the faces in a variable gadget. So the additional rectangles $\{b_i; 1 \leq i \leq (2m+2)\}$ enforce not to choose R_5 in an optimal solution. Also any additional rectangles dominates exactly 3 faces. So selecting any additional rectangle actually increases the size of the dominating set. Hence none of these additional rectangles belongs to an optimal solution. Now it is easy to convince that Lemma 6.4 remains true for this modified gadget even when we consider all the rectilinear faces of the subdivision.

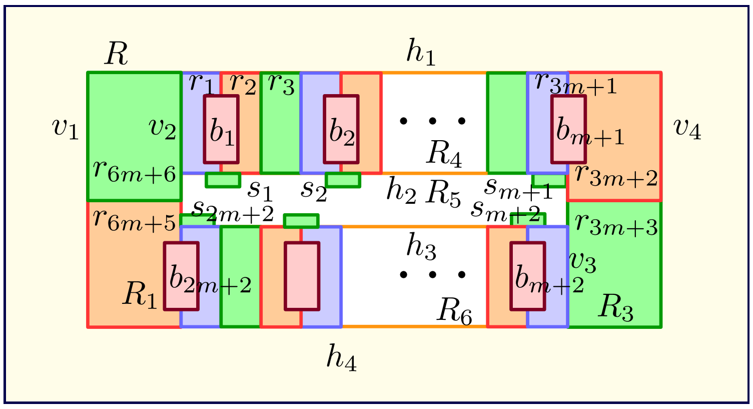


Figure 6.14: Modified variable gadget.

We now show that Lemma 6.5 is true when considering outer faces also. Notice that, similar to the STABBING-SUBDIVISION problem, each outer face shares two consecutive variable faces from a single variable gadget on either h_1, or h_4, or both

h_1 and h_4. See Figure 6.13 for an illustration of this fact. Hence, selecting D_i^i or D_i^2 will stab all the outer faces that shares boundary with the faces of the gadget of x_i. Hence, using the solution of size $n(2m+2)$, all the outer faces are also dominated. Thus Lemma 6.5 remains true even when we consider all faces. Hence we have the following theorem.

Theorem 6.8. *The* DOMINATING-SUBDIVISION *problem is* NP-Hard.

Summary

In this chapter, we study three problems, the STABBING-SUBDIVISION, INDEPENDENT-SUBDIVISION, and DOMINATING-SUBDIVISION on a given planar rectilinear subdivision. We prove that all these three problems are NP-Hard. We also show that these problem remains NP-Hard when we consider only rectangular faces of the subdivision. Further, we show that a local search produces a PTAS for the STABBING-SUBDIVISION problem. We also provide a constant factor approximation algorithm for the STABBING-SUBDIVISION problem. Note that both the Independent Set and the Dominating Set problems have PTAS es [Bak94]. Since our input is a planar subdivision, we can easily get its corresponding planar graph. As a result, we obtain PTAS es for both the INDEPENDENT-SUBDIVISION and DOMINATING-SUBDIVISION problems.

CHAPTER 7

Uniquely Restricted Matching

Contents

Related Publication:

1. Mathew C. Francis, Dalu Jacob, and Satyabrata Jana. "Uniquely restricted matchings in interval graphs." *SIAM Journal on Discrete Mathematics* 32.1 (2018): 148-172.

7.1 Introduction

Let $G = (V, E)$ be a graph. A set of edges $M \subseteq E(G)$ is said to be a matching if no two edges of M share a common vertex. The set of vertices in V that have an edge of M incident on them are called the vertices matched by M. A matching M is said to be uniquely restricted if there is no other matching that matches the same set of vertices as M. This problem is known to be NP-Complete for split graphs, bipartite graphs whereas is polynomially solvable for threshold graphs, cacti and block graphs [GHL01].

7.1.1 Our contributions

We propose linear-time dynamic programming algorithms for computing a maximum cardinality uniquely restricted matching in proper interval graphs and bipartite permutation graphs respectively. We show that the linear-time algorithm described for the problem for proper interval graphs in [GHL01] does not appear to work in all cases.

7.2 Preliminaries

Let M be a matching in a graph G. An even cycle in G is said to be an alternating cycle with respect to M if every second edge of the cycle belongs to M [GHL01]. The following theorem characterizes uniquely restricted matchings in terms of alternating cycles.

Theorem 7.1 ([GHL01])**.** *Let $G = (V,E)$ be a graph. A matching M in G is uniquely restricted if and only if there is no alternating cycle with respect to M in G.*

Proof. Let the input graph G contains an alternating cycle with respect to M, say $C = \langle (e_1,e'1),(e'1,e_2),(e_2,e'_2),\ldots,(e_m,e'_m),(e'_m,e_1)\rangle$ such that $(e_i,e'i) \in M$ for $1 \le i \le l$. Then $M' = C \setminus M$ is a different matching on the same vertices as M, and so, M is not uniquely restricted.

Conversely, let M is not uniquely restricted and let M' be a different matching on the same vertices as M. Now consider the subgraph $G[M \triangle M']$. The degree of any vertex in this subgraph must be exactly 2; therefore by Euler's theorem, a cycle must be formed in $M \triangle M'$ which is nothing but an alternating cycle with respect to M. ■

Note that an alternating cycle with respect to a matching M may not contain all the edges in M. But clearly, if M contains only two edges, then any alternating cycle with respect to M has to contain both the edges in M. Furthermore, it can be easily seen that if M is any matching and $M' \subseteq M$, then any alternating cycle with respect to M' is also an alternating cycle with respect to M. It therefore follows from Theorem 7.1 that any subset of a uniquely restricted matching is also a uniquely restricted matching. The following observation is easy to see.

Observation 7.1. *Let $G = (V,E)$ be any graph and let M be a matching in it. There is an alternating cycle of length 4 with respect to M if and only if there exist $e,e' \in M$ such that $\{e,e'\}$ is not a uniquely restricted matching.*

Proof. Any alternating cycle of length 4 with respect to M contains exactly two edges, say e and e', from M. Clearly, this cycle is also an alternating cycle with respect to the matching $\{e,e'\}$ and therefore by Theorem 7.1, $\{e,e'\}$ is not a uniquely restricted matching. On the other hand, if M contains two edges e and e' such that $\{e,e'\}$ is not a uniquely restricted matching, then by Theorem 7.1, there is an alternating cycle with respect to $\{e,e'\}$ and it is clear that this is an alternating cycle of length 4 with respect to M. ■

Lemma 7.1. *Let $G=(V,E)$ be any graph and let $\{uv,u'v'\}$ be a matching in it. The following statements are equivalent:*

(i) There is an alternating cycle with respect to $\{uv,u'v'\}$ in G.

(ii) Each of u,v has at least one neighbour in $\{u',v'\}$ and each of u',v' has at least one neighbour in $\{u,v\}$.

Proof. If there is an alternating cycle with respect to the matching $\{uv,u'v'\}$, then it is either $uvu'v'u$ or $uvv'u'u$. In any case, it is clear that each of u,v has at least one neighbour in $\{u',v'\}$ and each of u',v' has at least one neighbour in $\{u,v\}$.

Now suppose that each of u,v has at least one neighbour in $\{u',v'\}$ and each of u',v' has at least one neighbour in $\{u,v\}$. Then, $(vu' \notin E(G)$ or $v'u \notin E(G)) \Rightarrow (uu' \in E(G)$ and $vv' \in E(G))$. This means that if $vu' \notin E(G)$ or $v'u \notin E(G)$, then $uvv'u'u$ is an alternating cycle with respect to $\{uv,u'v'\}$ in G. If on the other hand both $vu',v'u \in E(G)$, then $uvu'v'u$ is an alternating cycle with respect to $\{uv,u'v'\}$ in G. ■

7.3 Proper Interval Graphs

By "interval" we shall mean a closed interval on the real line. An interval is denoted as $[a,b]$ where $a,b \in \mathbb{R}$ and $a \le b$, and is the set $\{x \in \mathbb{R} \colon a \le x \le b\}$. Given a graph G, a collection $\{I_u\}_{u \in V(G)}$ of intervals is said to be an *interval representation* of G if for distinct $u,v \in V(G)$, we have $uv \in E(G)$ if and only if $I_u \cap I_v \neq \emptyset$. Graphs which have interval representations are called *interval graphs*. The following theorem about uniquely restricted matchings in interval graphs is from [GHL01].

Theorem 7.2 ([GHL01]). *Let $G=(V,E)$ be an interval graph. Let M be a matching in G. Then the following statements are equivalent:*

(i) M is a uniquely restricted matching in G

(ii) There is no alternating cycle of length 4 with respect to M in G

(iii) For any two edges $e, e' \in M$, $\{e, e'\}$ is a uniquely restricted matching in G

Proof. We shall first show *(i)⇔(ii)*. By Theorem 7.1, if M is uniquely restricted, then there is no alternating cycle of any length with respect to M in G. So it suffices to show that when M is not uniquely restricted, there is an alternating cycle of length 4 with respect to M in G. Let $<$ be an ordering of the vertices of G according to a non-decreasing order of the left endpoints of their intervals in an interval representation of G. It is easy to see (and folklore) that the ordering $<$ has the property that for any $u,v,w \in V(G)$ such that $u < v < w$, $uw \in E(G) \Rightarrow uv \in E(G)$. Suppose that the matching M is not uniquely restricted. Then, by Theorem 7.1, there exists some alternating cycle with respect to M in G. Let $C = u_1u_2u_3 \ldots u_ku_1$ be an alternating cycle with respect to M of smallest possible length in G. Clearly, k is even and $4 \le k \le |V(G)|$. If $k = 4$, then this cycle is an alternating cycle of length 4 with respect to M and we are done. So let us suppose for the sake of contradiction that $k > 4$. If $u_iu_j \in E(G)$ for some two vertices u_i and u_j that are not consecutive on the cycle C (i.e., u_iu_j is a "chord" of C) and i and j are of different parity, then one of the two cycles into which the chord u_iu_j splits C will be an alternating cycle with respect to M of length smaller than k. As this is a contradiction to the assumption that C is the alternating cycle with respect to M of smallest possible length, we can assume that such chords are "forbidden", or in other words, there are no such chords for C. We shall also assume without loss of generality that $u_1 = \max_<\{u_1, u_2, \ldots, u_k\}$ and that $u_k < u_2$ (otherwise we can relabel the vertices of the cycle C to satisfy both these conditions). From the special property of the ordering $<$ and the fact that $u_ku_1 \in E(G)$, it can be seen that if $u_k < u_3$, then $u_ku_3 \in E(G)$. On the other hand, if $u_3 < u_k$, we again have $u_ku_3 \in E(G)$ as $u_3u_2 \in E(G)$. As u_ku_3 is a forbidden chord, we have a contradiction.

The fact that *(ii)⇔(iii)* has already been noted in Observation 7.1. ■

A *proper interval representation* is an interval representation in which no interval strictly contains another interval. *Proper interval graphs* are the graphs which have proper interval representations.

Definition 7.1. For a graph $G = (V, E)$, an ordering $<$ of $V(G)$ is said to be a *proper vertex ordering* if for $u,v,w \in V(G)$ such that $u < v < w$, $uw \in E(G) \Rightarrow uv, vw \in E(G)$.

Proper vertex orderings are called "proper orderings" in [GHL01]. Note that in a proper interval representation of a graph G, the ordering of the vertices of G

according to the left endpoints of the intervals corresponding to them is the same as their ordering according to the right endpoints of their intervals, since no interval is contained in another. It is easy to see that this ordering is a proper vertex ordering of G. In fact, it is folklore that a graph is a proper interval graph if and only if it has a proper vertex ordering.

We can assume that G is connected as if it is not, we can easily run the algorithm separately in each component to find a maximum cardinality uniquely restricted matching in each of them and then take a union of those to obtain a maximum cardinality uniquely restricted matching in G. We can also assume that a proper interval representation of the graph G, and therefore a proper vertex ordering of G, is available, as there are well known linear-time algorithms that can generate the proper interval representation of a graph, given its adjacency list [Cor04]. From here onwards, we will see a proper interval graph G exclusively in terms of a proper vertex ordering of it and will not be concerned with any interval representation of G at all.

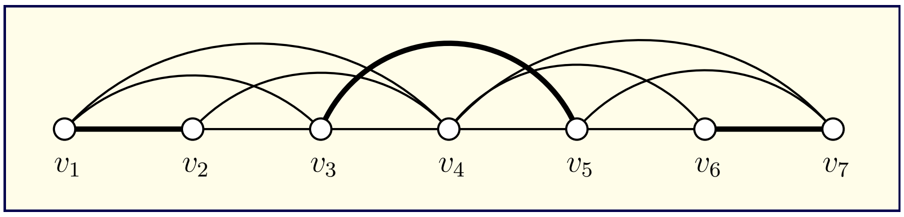


Figure 7.1: A proper interval graph with vertices arranged according to a proper vertex ordering. The bold edges represent a maximum cardinality uniquely restricted matching.

Algorithm by Golumbic et al. [GHL01]

Here we show that the algorithm described for the problem for proper interval graphs in [GHL01] does not appear to work in all cases. Below, we mention their algorithm. Let $R(v_i)$ denote the rightmost vertex v_j in the proper ordering such that $(v_j, v_i) \in E$. If v_i is an isolated vertex, then $R(v_i) = v_i$.

Algorithm 4

```
1: choose the first edge $(v_l, v_{l+1})$ that exists and put it in the matching $M$.
2: set $u \leftarrow v_l$ and $w \leftarrow v_{l+1}$
3: until we reach the end of the ordering
4:     (a) if $R(u) = R(w) = v_l$ then
```

```
5:          (i) set $u \leftarrow v_l$ and $w \leftarrow v_{l+1}$
6:          (ii) add $(u,w)$ to the matching $M$.
7:      (b) if $R(u) = v_l \neq R(w)$ then
8:          (i) set $u \leftarrow v_{l+1}$ and $w \leftarrow v_{l+2}$
9:          (ii) add $(u,w)$ to the matching $M$.
10:         return $M$.
```

In Figure 7.1, the bold edges represent a maximum cardinality uniquely restricted matching. The algorithm 4 from [GHL01], given a proper vertex ordering as input, always produces a uniquely restricted matching consisting of edges between consecutive vertices. But every such matching in this graph, given this particular vertex ordering, has at most two edges. So the Algorithm 4 described in [GHL01] for proper interval graph does not work in all inputs.

Now we describe our algorithm. Before that we define some notations. Let $<$ be a proper vertex ordering of a graph G. For an edge $e = uv \in E(G)$, we define $l_<(e) = \min_<\{u,v\}$ and $r_<(e) = \max_<\{u,v\}$. We shorten $l_<(e)$ and $r_<(e)$ to just $l(e)$ and $r(e)$ when the proper vertex ordering $<$ is clear from the context.

Lemma 7.2 ([GHL01]). *Let G be a proper interval graph with a proper vertex ordering $<$. If $\{e,e'\}$ is a uniquely restricted matching in G, then either $r(e) < l(e')$ or $r(e') < l(e)$.*

Proof. As $\{e,e'\}$ is a matching, we can be sure that $l(e), r(e), l(e'), r(e')$ are all distinct vertices. Suppose that $l(e') < r(e)$ and $l(e) < r(e')$. Let us assume without loss of generality that $l(e) < l(e')$. Since $l(e)r(e) \in E(G)$ and $l(e) < l(e') < r(e)$, by Definition 7.1, we have $l(e)l(e'), r(e)l(e') \in E(G)$. Now if $r(e) < r(e')$, then as $l(e') < r(e) < r(e')$ and $l(e')r(e') \in E(G)$, we have by Definition 7.1 that $l(e')r(e), r(e')r(e) \in E(G)$. On the other hand, if $r(e') < r(e)$, then as $l(e) < r(e') < r(e)$ and $l(e)r(e) \in E(G)$, Definition 7.1 gives us $l(e)r(e'), r(e)r(e') \in E(G)$. Thus, in any case, each of $l(e), r(e)$ has a neighbour in $\{l(e'), r(e')\}$ and each of $l(e'), r(e')$ has a neighbour in $\{l(e), r(e)\}$. We can now use Lemma 7.1 to conclude that there is an alternating cycle with respect to $\{e,e'\}$ in G. But as $\{e,e'\}$ is a uniquely restricted matching, this contradicts Theorem 7.1. ■

Lemma 7.3. *Let G be a proper interval graph with a proper vertex ordering $<$ and let $e, e' \in E(G)$. Then $\{e,e'\}$ is a uniquely restricted matching in G if and only if $l(e), l(e')$ are distinct and nonadjacent or $r(e), r(e')$ are distinct and nonadjacent.*

Proof. Suppose that $\{e,e'\}$ is a uniquely restricted matching. Then clearly $l(e)$, $r(e)$, $l(e')$, $r(e')$ are all distinct vertices. If $l(e)l(e') \in E(G)$ and $r(e)r(e') \in E(G)$, then we have the alternating cycle $l(e)l(e')r(e')r(e)\,l(e)$ with respect to $\{e,e'\}$ in G, which is a contradiction to Theorem 7.1.

Let us now prove the other direction of the claim. Suppose that $l(e)$ and $l(e')$ are distinct and nonadjacent. We shall assume without loss of generality that $l(e) < l(e')$. As $l(e)l(e') \notin E(G)$ and $l(e) < l(e') < r(e')$, we have from Definition 7.1 that $l(e)$ has no neighbour in $\{l(e'),r(e')\}$. By Lemma 7.1 and Theorem 7.1, this implies that $\{e,e'\}$ is a uniquely restricted matching in G. Now let us suppose that $r(e)$ and $r(e')$ are distinct and nonadjacent. Again, we shall assume without loss of generality that $r(e) < r(e')$. As $l(e) < r(e) < r(e')$ and $r(e)r(e') \notin E(G)$, we have from Definition 7.1 that $r(e')$ has no neighbour in $\{l(e),r(e)\}$. Lemma 7.1 and Theorem 7.1 can now be used to infer that $\{e,e'\}$ is a uniquely restricted matching in G. ■

Lemma 7.4. *Let G be a proper interval graph with a proper vertex ordering $<$. Let e_1,e_2,e_3 be distinct edges of G such that $l(e_1) \le l(e_2) \le l(e_3)$ and $r(e_1) \le r(e_2) \le r(e_3)$. If $\{e_1,e_3\}$ is not a uniquely restricted matching in G, then neither $\{e_1,e_2\}$ nor $\{e_2,e_3\}$ is a uniquely restricted matching in G.*

Proof. We shall show that $\{e_2,e_3\}$ is not a uniquely restricted matching in G. The proof for the case of $\{e_1,e_2\}$ is similar and is left to the reader. If $\{e_2,e_3\}$ is not even a matching in G, then we are immediately done. So we shall assume otherwise—i.e., $l(e_2),r(e_2),l(e_3),r(e_3)$ are all distinct vertices of G. This means that $l(e_1) \le l(e_2) < l(e_3)$ and $r(e_1) \le r(e_2) < r(e_3)$. By Lemma 7.3, we now have $l(e_1)l(e_3), r(e_1)r(e_3) \in E(G)$. This implies by Definition 7.1 that $l(e_2)l(e_3), r(e_2)r(e_3) \in E(G)$. Lemma 7.3 now implies that $\{e_2,e_3\}$ is not a uniquely restricted matching. ■

From Lemma 7.2, it follows that the edges of any uniquely restricted matching M in a proper interval graph G with a proper vertex ordering $<$ can be labelled as $e_1,e_2,\ldots,e_{|M|}$ such that $l(e_1) < r(e_1) < l(e_2) < r(e_2) < \cdots < l(e_{|M|}) < r(e_{|M|})$. We say that the uniquely restricted matching M *starts with* the edge e_1.

We now give a stronger version of Theorem 7.2 for the case of proper interval graphs.

Theorem 7.3. *Let G be a proper interval graph and $<$ a proper vertex ordering of it. Let $M = \{e_1,e_2,\ldots,e_t\}$ be a matching in G where $l(e_1) < l(e_2) < \cdots < l(e_t)$. The matching M is uniquely restricted in G if and only if $\{e_i,e_{i+1}\}$ is a uniquely restricted matching in G, for each $i \in \{1,2,\ldots,t-1\}$.*

Proof. As every subset of a uniquely restricted matching is also a uniquely restricted matching, to prove the theorem, it is sufficient to show that whenever M is not a uniquely restricted matching, $\{e_i, e_{i+1}\}$ is not a uniquely restricted matching in G for some $i \in \{1, 2, \ldots, t-1\}$. Suppose that M is not a uniquely restricted matching. By Theorem 7.2, there exists some subset $\{e_p, e_q\}$ of M, where $1 \leq p < q \leq t$, such that $\{e_p, e_q\}$ is not a uniquely restricted matching. We choose p and q such that for any p', q' with $1 \leq p' < q' \leq t$ and $q' - p' < q - p$, $\{e_{p'}, e_{q'}\}$ is a uniquely restricted matching in G. Suppose that $q > p+1$. Observing that $l(e_p) < l(e_{q-1}) < l(e_q)$, we can deduce that if $r(e_q) < r(e_{q-1})$ or $r(e_{q-1}) < r(e_p)$, then by Lemma 7.2, either $\{e_{q-1}, e_q\}$ or $\{e_p, e_{q-1}\}$ respectively is not a uniquely restricted matching, in each case contradicting our choice of p and q. Therefore, we get $r(e_p) < r(e_{q-1}) < r(e_q)$. Now we can apply Lemma 7.4 to conclude that $\{e_{q-1}, e_q\}$ is not a uniquely restricted matching, again contradicting our choice of p and q. Thus we infer that $q = p+1$. For $i = p$, we now have that $\{e_i, e_{i+1}\}$ is not a uniquely restricted matching, hence the proof. ■

Corollary 7.1. *Let G be a proper interval graph with a proper vertex ordering $<$ and let M be a uniquely restricted matching in G starting with an edge $e' \in E(G)$. Let $e \in E(G)$ be such that $r(e) < l(e')$ and $\{e, e'\}$ is a uniquely restricted matching in G. Then $\{e\} \cup M$ is a uniquely restricted matching in G starting with e.*

Proof. The proof follows directly from Theorem 7.3. ■

From here onwards, we assume that G is a connected proper interval graph with a proper vertex ordering $<$. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$ where $v_1 < v_2 < \cdots < v_n$.

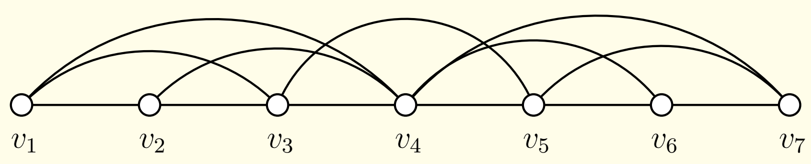
Observation 7.2. *For $1 \leq i < n$, $v_i v_{i+1} \in E(G)$.*

Proof. Suppose that for some $i \in \{1, 2, \ldots, n-1\}$, $v_i v_{i+1} \notin E(G)$. If for some pair of vertices v_p, v_q, where $1 \leq p \leq i < i+1 \leq q \leq n$, we have $v_p v_q \in E(G)$, then from Definition 7.1, $v_p v_q \in E(G) \Rightarrow v_p v_{i+1} \in E(G) \Rightarrow v_i v_{i+1} \in E(G)$, which is a contradiction. Therefore, there does not exist any edge $v_p v_q \in E(G)$ such that $p \leq i$ and $q \geq i+1$. But this would mean that there is no edge in G between a vertex in $\{v_1, v_2, \ldots, v_i\}$ and a vertex in $\{v_{i+1}, v_{i+2}, \ldots, v_n\}$, implying that G is disconnected. This contradicts our assumption that G is a connected graph. ■

For $u \in V(G)$, we define $\lambda(u) = \min_<\{v \in N(u) \cup \{u\}\}$ and $\rho(u) = \max_<\{v \in N(u) \cup \{u\}\}$. The following observation is an easy consequence of the property of proper vertex orderings given in Definition 7.1.

Observation 7.3. *For any vertex $u \in V(G)$, $N(u) = \{x \in V(G) : \lambda(u) \leq x \leq \rho(u)\}$.*

We now associate a pair of edges with each edge $e \in E(G)$. The *left successor* of e, denoted by $\sigma_l(e)$, is defined to be the edge $v_{i+1}v_{i+2}$, where $v_i = \rho(l(e))$. It is clear from Observation 7.2 that $\sigma_l(e)$ exists in G if and only if $\rho(l(e)) < v_{n-1}$. The *right successor* of e, denoted by $\sigma_r(e)$, is defined to be the edge $\lambda(v_{i+1})v_{i+1}$ where $v_i = \rho(r(e))$. Note that for any vertex $u \in V(G)$ for which $\lambda(u) \neq u$, we have $\lambda(u)u \in E(G)$. Moreover, by Observation 7.2, it follows that for every vertex $u \in V(G) \setminus \{v_1\}$, $\lambda(u) \neq u$. Therefore, it can be concluded that $\sigma_r(e)$ exists in G if and only if $\rho(r(e)) < v_n$. Please see Figure 7.2 for an example that shows the left and right successors of the edges of a proper interval graph, given a proper vertex ordering of it.



e	$\sigma_l(e)$	$\sigma_r(e)$
v_1v_2	v_5v_6	v_3v_5
v_1v_3	v_5v_6	v_4v_6
v_2v_4	v_5v_6	-
v_3v_5	v_6v_7	-
v_4v_5	-	-

Figure 7.2: Example of an proper interval graph shown with vertices arranged according to the proper vertex ordering $v_1, v_2, \ldots, v_7$. The table shows some edges and their left and right successors. Where a particular successor does not exist for an edge, the corresponding entry is marked as "-".

Observation 7.4. *Let $e \in E(G)$.*

(a) If $\sigma_l(e)$ exists, then $r(e) < l(\sigma_l(e))$ and $\{e, \sigma_l(e)\}$ is a uniquely restricted matching.

(b) If $\sigma_r(e)$ exists, then $r(e) < l(\sigma_r(e))$ and $\{e, \sigma_r(e)\}$ is a uniquely restricted matching.

Proof. We shall first prove (a). As $l(e)r(e) \in E(G)$, we know that $\rho(l(e)) \geq r(e)$. By definition of $\sigma_l(e)$, we have $r(e) \leq \rho(l(e)) < l(\sigma_l(e))$. Therefore, we have $l(e)l(\sigma_l(e)) \notin E(G)$. This implies by Lemma 7.3 that $\{e, \sigma_l(e)\}$ is a uniquely restricted matching in G.

Now let us prove (b). It is clear from the definition of $\sigma_r(e)$ that $\rho(r(e)) < r(\sigma_r(e))$. Therefore, we have $r(e) < r(\sigma_r(e))$ and $r(e)r(\sigma_r(e)) \notin E(G)$. From Lemma 7.3, we can now conclude that $\{e, \sigma_r(e)\}$ is a uniquely restricted matching in G. By Lemma 7.2, we also get that $r(e) < l(\sigma_r(e))$. ■

Lemma 7.5. *Let M be a uniquely restricted matching starting with an edge e in G such that $|M| \geq 3$. Then there exists a uniquely restricted matching M' that starts with either $\sigma_l(e)$ or $\sigma_r(e)$ in G such that $|M'| = |M| - 1$.*

Proof. Let $M = \{e = e_1, e_2, e_3, \ldots, e_t\}$, where $t \geq 3$ and $l(e_1) < l(e_2) < \cdots < l(e_t)$. By Lemma 7.2, we know that $l(e_1) < r(e_1) < l(e_2) < r(e_2) < \cdots < l(e_t) < r(e_t)$. As $\{e, e_2\}$ is a uniquely restricted matching, we know from Theorem 7.1 that at least one of $l(e)l(e_2), r(e)r(e_2)$ is not an edge.

Let us suppose first that $l(e)l(e_2) \notin E(G)$. Then we know by Observation 7.3 that $\rho(l(e)) < l(e_2) < r(e_2) \leq v_n$. This implies that $\rho(l(e)) < v_{n-1}$ and therefore, $\sigma_l(e)$ exists. It also implies that $l(\sigma_l(e)) \leq l(e_2)$ and that $r(\sigma_l(e)) \leq r(e_2)$. We can now apply Lemma 7.4 to the edges $\sigma_l(e)$, e_2 and e_3 to conclude that $\{\sigma_l(e), e_3\}$ is a uniquely restricted matching. Since $\{e_3, e_4, \ldots, e_t\}$ is a uniquely restricted matching and $r(\sigma_l(e)) \leq r(e_2) < l(e_3)$, we can use Corollary 7.1 to conclude that $M' = \{\sigma_l(e), e_3, e_4, \ldots, e_t\}$ is a uniquely restricted matching starting with $\sigma_l(e)$ in G. As $|M'| = |M| - 1$, we are done.

Now let us consider the case when $l(e)l(e_2) \in E(G)$ but $r(e)r(e_2) \notin E(G)$. We then have by Observation 7.3 that $\rho(r(e)) < r(e_2)$. This implies that $\rho(r(e)) < v_n$, which means that $\sigma_r(e)$ exists. It also implies that $r(\sigma_r(e)) \leq r(e_2)$. As $l(e) < r(e) < l(e_2)$, and because we have assumed that $l(e)l(e_2) \in E(G)$, we can deduce from Definition 7.1 that $r(e)l(e_2) \in E(G)$. Therefore, by Observation 7.3, $\rho(r(e)) \geq l(e_2)$. This gives us $l(e_2) < r(\sigma_r(e)) \leq r(e_2)$. As $l(e_2)r(e_2) \in E(G)$, we now have by Definition 7.1 that $l(e_2)r(\sigma_r(e)) \in E(G)$, which implies by Observation 7.3 that $\lambda(r(\sigma_r(e))) \leq l(e_2)$. It can be seen from the definition of $\sigma_r(e)$ that $l(\sigma_r(e)) = \lambda(r(\sigma_r(e)))$. Thus, we now have $l(\sigma_r(e)) \leq l(e_2)$. We can now apply Lemma 7.4 to the edges $\sigma_r(e)$, e_2 and e_3 to conclude that $\{\sigma_r(e), e_3\}$ is a uniquely restricted matching. As before, Corollary 7.1 now gives us that $M' = \{\sigma_r(e), e_3, e_4, \ldots, e_t\}$ is a uniquely restricted matching starting with $\sigma_r(e)$ in G. The proof is concluded by noting that we have $|M'| = |M| - 1$ in this case as well. ■

Now for each edge $e \in E(G)$ we define a set $U(e)$ of edges as follows.

$$U(e) = \begin{cases} \{e\} & \text{if neither } \sigma_l(e) \text{ nor } \sigma_r(e) \text{ exist} \\ \{e\} \cup U(\sigma_l(e)) & \text{if } \sigma_r(e) \text{ does not exist, or if both exist} \\ & \text{and } |U(\sigma_l(e))| \geq |U(\sigma_r(e))| \\ \{e\} \cup U(\sigma_r(e)) & \text{if } \sigma_l(e) \text{ does not exist, or if both exist} \\ & \text{and } |U(\sigma_l(e))| < |U(\sigma_r(e))| \end{cases}$$

From Observation 7.4, we know that $r(e) < l(\sigma_l(e))$ and $r(e) < l(\sigma_r(e))$. This means that $U(e)$ is well-defined. The next two lemmas will show that $U(e)$ is always a uniquely restricted matching starting with e of maximum possible cardinality (among the uniquely restricted matchings starting with e in G).

Lemma 7.6. *For any edge $e \in E(G)$, $U(e)$ is a uniquely restricted matching starting with e.*

Proof. We shall prove this by induction on $|U(e)|$. It is clear from the definition of $U(e)$ that $|U(e)| \geq 1$. If $|U(e)| = 1$, then it must be the case that $U(e) = \{e\}$. The statement of the lemma is easily seen to be true in this case. We shall now assume that $|U(e)| > 1$ and that the statement of the lemma has been shown to be true for all e' such that $|U(e')| < |U(e)|$. In this case, from the definition of $U(e)$, one of the following occurs: either (a) $\sigma_l(e)$ exists and $U(e) = \{e\} \cup U(\sigma_l(e))$, or (b) $\sigma_r(e)$ exists and $U(e) = \{e\} \cup U(\sigma_r(e))$. If (a) occurs, then we have $|U(\sigma_l(e))| = |U(e)| - 1$, and therefore by the induction hypothesis, $U(\sigma_l(e))$ is a uniquely restricted matching starting with $\sigma_l(e)$. Now, it follows from Observation 7.4 and Corollary 7.1 that $\{e\} \cup U(\sigma_l(e)) = U(e)$ is a uniquely restricted matching in G starting with e. On the other hand, if (b) occurs, then $|U(\sigma_r(e))| = |U(e)| - 1$, and therefore by the induction hypothesis, $U(\sigma_r(e))$ is a uniquely restricted matching starting with $\sigma_r(e)$. Then it follows from Observation 7.4 and Corollary 7.1 that $\{e\} \cup U(\sigma_r(e)) = U(e)$ is a uniquely restricted matching in G starting with e. This completes the proof. ■

Lemma 7.7. *Let M be a uniquely restricted matching starting with an edge $e \in E(G)$. Then, $|M| \leq |U(e)|$.*

Proof. We prove this by induction on $|U(e)| = k$.

Assume that $k = 1$. In this case, $U(e) = \{e\}$. Suppose that there exists a uniquely restricted matching M starting with e such that $|M| > 1$. Then there exists at least one edge e' in M such that $e' \neq e$. Now since $U(e) = \{e\}$, we can see from the definition of $U(e)$ that neither $\sigma_l(e)$ nor $\sigma_r(e)$ exist. As we noted earlier, this means that

$\rho(l(e)) \geq v_{n-1}$ and $\rho(r(e)) \geq v_n$. As M starts with e and $e' \in M$, we know by Lemma 7.2 that $l(e) < r(e) < l(e') < r(e')$. Since $\rho(l(e)) \geq v_{n-1}$, it must be the case that $\rho(l(e)) \geq l(e')$, which by Observation 7.3 implies that $l(e)l(e') \in E(G)$. Similarly, since $\rho(r(e)) \geq v_n$, we have $\rho(r(e)) \geq r(e')$, from which it follows by Observation 7.3 that $r(e)r(e') \in E(G)$. From Lemma 7.3, we now have that $\{e, e'\}$ is not a uniquely restricted matching, which is a contradiction to the fact that $\{e, e'\} \subseteq M$. Therefore, the statement of the lemma is true when $k = 1$.

Now, assume that $k > 1$ and that the statement of the lemma is true for every edge $e' \in E(G)$ with $|U(e')| < k$. Suppose that there exists a uniquely restricted matching M starting with e in G such that $|M| > k$. From Lemma 7.6, we know that $U(e)$ is a uniquely restricted matching in G. Note that as $k > 1$ and $|M| > k$, we have $|M| \geq 3$. By Lemma 7.5, we can now infer that there exists a uniquely restricted matching M' in G with $|M'| = |M| - 1$ such that M' starts with either $\sigma_l(e)$ or $\sigma_r(e)$. From the definition of $U(e)$, it can be seen that $|U(e)| \geq \max\{|U(\sigma_l(e))|, |U(\sigma_r(e))|\} + 1$. Since $|U(e)| = k$, we can therefore infer that $|U(\sigma_l(e))| \leq k - 1$ and $|U(\sigma_r(e))| \leq k - 1$. We can now apply the induction hypothesis on the starting edge of M' (which is either $\sigma_l(e)$ or $\sigma_r(e)$) to conclude that $|M'| \leq k - 1$. But as $|M'| = |M| - 1$, we now get $|M| \leq k$, which contradicts our assumption that $|M| > k$. ■

Lemma 7.8. *$U(v_1v_2)$ is a uniquely restricted matching of maximum cardinality in G.*

Proof. From Lemma 7.6, it is clear that $U(v_1v_2)$ is a uniquely restricted matching starting with v_1v_2 in G. Let $\{e_1, e_2, \ldots, e_k\}$ be any uniquely restricted matching in G where $l(e_1) < l(e_2) < \cdots < l(e_k)$. Clearly, $v_1 \leq l(e_1)$ and $v_2 \leq r(e_1)$. From Lemma 7.2, we have $l(e_1) < r(e_1) < l(e_2) < r(e_2) < \cdots < l(e_k) < r(e_k)$. Therefore, we have $v_1 \leq l(e_1) < l(e_2)$ and $v_2 \leq r(e_1) < r(e_2)$. We can now apply Lemma 7.4 to the edges v_1v_2, e_1 and e_2 to conclude that $\{v_1v_2, e_2\}$ is a uniquely restricted matching in G. By Corollary 7.1, we now have that $\{v_1v_2, e_2, e_3, \ldots, e_k\}$ is a uniquely restricted matching in G starting with v_1v_2. As the cardinality of this matching is k, we have by Lemma 7.7 that $|U(v_1v_2)| \geq k$. ■

Theorem 7.4. *There is a linear-time algorithm that computes a maximum cardinality uniquely restricted matching in a given proper interval graph.*

Proof. Let the input graph G have n vertices and m edges. We can assume that G is connected, as if it is not, we can just run the algorithm separately in each component of G, and then return the union of the maximum cardinality uniquely restricted matchings found in each component. From the input adjacency list, we can use the well known $O(n+m)$ time algorithms to generate a proper vertex ordering of G (for example, [Cor04] or [HH04]). Assuming that the position of each vertex in the ordering is known as a unique integer in $[1,n]$ associated with each vertex, we can easily generate $l(e)$ and $r(e)$ for every edge $e \in E(G)$ in $O(n+m)$ time by making one pass through the adjacency list. During the same pass, we can compute $\lambda(u)$ and $\rho(u)$ for every vertex $u \in V(G)$. Once we are done with this, we can easily find $\sigma_l(e)$ and $\sigma_r(e)$ for any edge $e \in E(G)$ in $O(1)$ time. For every edge $e \in E(G)$, $U(e)$ can be stored as a list, which is empty to start with. The following subroutine computes $U(e)$ for a given edge e.

Algorithm 5 ComputeU(e)

```
1: if $\sigma_l(e)$ exists and $U(\sigma_l(e)) = \emptyset$ then ComputeU($\sigma_l(e)$)
2: if $\sigma_r(e)$ exists and $U(\sigma_r(e)) = \emptyset$ then ComputeU($\sigma_r(e)$)
3: if both $\sigma_l(e)$ and $\sigma_r(e)$ exist then
4:     if $|U(\sigma_l(e))| \geq |U(\sigma_r(e))|$ then set $U(e) = \{e\} \cup U(\sigma_l(e))$
5:     else set $U(e) = \{e\} \cup U(\sigma_r(e))$
6: else if $\sigma_l(e)$ exists then set $U(e) = \{e\} \cup U(\sigma_l(e))$
7: else if $\sigma_r(e)$ exists then set $U(e) = \{e\} \cup U(\sigma_r(e))$
8: else set $U(e) = \{e\}$
```

The main algorithm just calls the procedure ComputeU(v_1v_2), where v_1 and v_2 are the vertices at the first and second positions, respectively in the ordering (recall that by Observation 7.2, $v_1v_2 \in E(G)$). The algorithm then finishes by returning the set of edges $U(v_1v_2)$. The correctness of the algorithm is guaranteed by Lemma 7.8. As every other part of the algorithm except the call to the procedure ComputeU(v_1v_2) takes $O(n+m)$ time, we shall restrict our attention to the time taken to complete this call. Notice that for any edge $e \in E(G)$, the time spent inside the procedure ComputeU(e) outside of the recursive calls to ComputeU is $O(1)$. Also observe that for any edge $e \in E(G)$, the call to ComputeU(e) happens at most once, which means that there are at most m calls to the procedure ComputeU. Therefore, the time taken to complete the procedure ComputeU(v_1v_2) is $O(m)$. Thus, the algorithm runs in time $O(n+m)$. ■

7.4 Bipartite Permutation Graphs

A bijection π of $\{1,2,\ldots,n\}$ to itself is called a permutation of order n. We write $\pi = (\pi(1),\pi(2),\ldots,\pi(n))$ to define a permutation of order n. The simple undirected graph G_π associated with a permutation π is a graph with $V(G_\pi) = \{1,2,\ldots,n\}$ and $E(G_\pi) = \{ij : (i-j)(\pi(i)-\pi(j)) < 0\}$. A graph G on n vertices is said to be a *permutation graph* if it is isomorphic to G_π for some permutation π of order n. In other words, a graph G on n vertices is a permutation graph if there exists a bijection $f : V(G) \to \{1,2,\ldots,n\}$ and a permutation π of order n such that for $u,v \in V(G)$, $uv \in E(G) \Leftrightarrow (f(u)-f(v))(\pi(f(u))-\pi(f(v))) < 0$.

Definition 7.2. For a graph $G = (V,E)$, an ordering $<$ of $V(G)$ is said to be a *doubly-transitive vertex ordering* if for $u,v,w \in V(G)$ such that $u < v < w$,

(a) $uv, vw \in E(G) \Rightarrow uw \in E(G)$, and

(b) $uw \in E(G) \Rightarrow uv \in E(G)$ or $vw \in E(G)$.

Let G be a permutation graph on n vertices and let f be the bijection from $V(G)$ to $\{1,2,\ldots,n\}$ as described above. Let $V(G) = \{v_1,v_2,\ldots,v_n\}$, where $v_i = f^{-1}(i)$ for $1 \le i \le n$. It is easy to see that $v_1,v_2,\ldots,v_n$ is a doubly-transitive vertex ordering of G. (It is well known and easy to see that the digraph with vertex set $V(G)$ and edge set $\{(u,v) : uv \in E(G)$ and $f(u) < f(v)\}$ corresponds to a transitive orientation of G. Similarly, the digraph with vertex set $V(G)$ and edge set $\{(u,v) : uv \notin E(G)$ and $f(u) < f(v)\}$ corresponds to a transitive orientation of $\overline{G}$. Hence our choice of the name "doubly-transitive" for this kind of vertex ordering.) Using the fact that permutation graphs are exactly the graphs that are both comparability and co-comparability [DM41], it is easy to see (and folklore) that a graph G is a permutation graph if and only if it has a doubly-transitive vertex ordering.

A permutation graph that is also bipartite is called a *bipartite permutation graph*. The class of bipartite permutation graphs is known to be the same as the classes of proper interval bigraphs, bipartite co-comparability graphs, bipartite asteroidal triple-free graphs and bipartite trapezoid graphs [HH04]. As permutation graphs do not contain odd induced cycles of length more than three, it is straightforward to verify that bipartite permutation graphs are exactly triangle-free permutation graphs. For such a graph, the following observation can easily be seen to be true.

Observation 7.5. *Let $<$ be a doubly-transitive vertex ordering of a bipartite permutation graph G.*

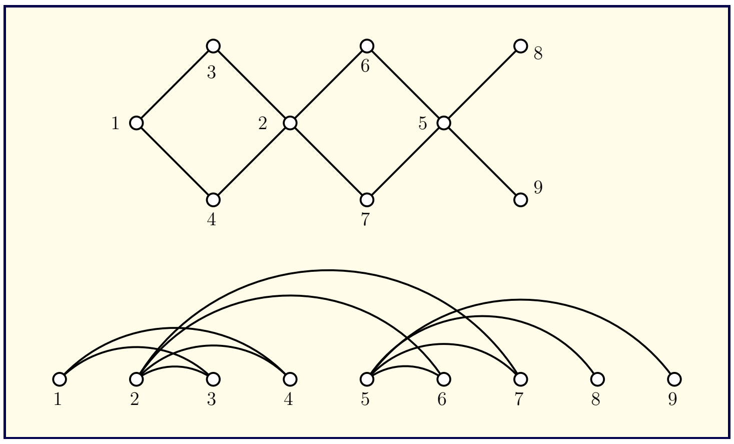


Figure 7.3: The figure at the top shows a bipartite permutation graph where $f(u)$ is written near each vertex u and $\pi = (3,4,1,6,7,2,8,9,5)$. The figure at the bottom shows a doubly-transitive vertex ordering of this graph.

(a) If $u < v$ and $uv \in E(G)$, then there exists no $w > v$ such that $vw \in E(G)$.

(b) If $u < v < w$ and $uw \in E(G)$, then $uv \in E(G)$ or $vw \in E(G)$ but not both.

Please refer [SBS87] for some other vertex ordering characterizations for bipartite permutation graphs that turn out to be useful in developing linear-time algorithms for problems that are NP-hard for larger classes of graphs.

Let G be a bipartite permutation graph with no isolated vertices, having a doubly-transitive vertex ordering $<$. For an edge $e = uv \in E(G)$, we define $l_<(e) = \min_<\{u,v\}$ and $r_<(e) = \max_<\{u,v\}$. When the ordering $<$ is clear from the context, we shorten $l_<(e)$ and $r_<(e)$ to $l(e)$ and $r(e)$ respectively.

A vertex $u \in V(G)$ is said to be a *left-vertex* if there is some edge $e \in E(G)$ such that $u = l(e)$. Similarly, a vertex $u \in V(G)$ is said to be a *right-vertex* if there exists some edge $e \in E(G)$ such that $u = r(e)$. As G has no isolated vertices, it is clear that every vertex in G is either a left-vertex or a right-vertex. We claim that no vertex can be both a left-vertex and a right-vertex. This is because if a vertex $u \in V(G)$ is such that $u = l(e) = r(e')$, for some $e, e' \in E(G)$, then we have $l(e') < u < r(e)$ and $l(e')u, ur(e) \in E(G)$, which contradicts Observation 7.5(a). Thus each vertex of G is either a left-vertex or a right-vertex but not both.

If u is a left-vertex, then it can have no neighbour v such that $v < u$, because if it does, then u becomes the right-vertex of the edge vu, which is a contradiction to the fact that no vertex can be both a left-vertex and a right-vertex. For the same reason, if u is a right-vertex, it can have no neighbour v such that $u < v$.

Since for every edge $e \in E(G)$, $l(e)$ is a left-vertex and $r(e)$ is a right-vertex, it can be concluded that every edge of G is between a left-vertex and a right-vertex. This tells us that the set of left-vertices and the set of right-vertices are both independent sets of G. Since G has no isolated vertices, these sets form a bipartition of the bipartite graph G.

We say that a vertex $u \in V(G)$ is *underneath* an edge $e \in E(G)$ if $l(e) \leq u \leq r(e)$. Suppose that a left-vertex $u \in V(G)$ is underneath an edge $e \in E(G)$. Clearly, $u \neq r(e)$, as it is a left-vertex and no vertex can be both a left-vertex and a right-vertex. If $u \neq l(e)$, it follows from Observation 7.5(b) that u is adjacent to $r(e)$ (note that u cannot be adjacent to $l(e)$ as both are left-vertices). If $u = l(e)$, then clearly it is adjacent to $r(e)$. Therefore, we can conclude that if a left-vertex is underneath an edge $e \in E(G)$, then it is adjacent to $r(e)$. Using very similar arguments, we can also see that if a right-vertex is underneath an edge $e \in E(G)$, then it is adjacent to $l(e)$.

Lemma 7.9. *Let G be a bipartite permutation graph with a doubly-transitive vertex ordering $<$ and let $e, e' \in E(G)$ such that $l(e) < l(e')$.*

(a) If $r(e) < l(e')$, then $\{e, e'\}$ is a uniquely restricted matching in G.

(b) If $l(e') < r(e) < r(e')$, then $\{e, e'\}$ is a uniquely restricted matching in G if and only if $l(e)r(e') \notin E(G)$.

(c) If $r(e') < r(e)$, then $\{e, e'\}$ is not a uniquely restricted matching in G.

Proof. Suppose that $r(e) < l(e')$. Then as we have $l(e) < r(e) < l(e') < r(e')$ and that $r(e)$ is a right-vertex, neither $l(e')$ nor $r(e')$ is a neighbour of $r(e)$. Then it can be seen from Lemma 7.1 that there is no alternating cycle with respect to $\{e, e'\}$ in G, which further implies by Theorem 7.1 that $\{e, e'\}$ is a uniquely restricted matching in G. This proves (a).

Now let us consider the case when $l(e') < r(e) < r(e')$. Then, as $r(e)$ is a right-vertex underneath the edge e', we know that $l(e')r(e) \in E(G)$. If $l(e)r(e') \in E(G)$, then we have the alternating cycle $l(e)r(e')l(e')r(e)l(e)$ with respect to $\{e, e'\}$ in G. If $l(e)r(e') \notin E(G)$, then we have that there is no neighbour of $r(e')$ in $\{l(e), r(e)\}$ (note that $r(e)$ and $r(e')$ cannot be adjacent as they are both right-vertices), implying by Lemma 7.1 that there is no alternating cycle with respect to $\{e, e'\}$ in G. This shows that there is an alternating cycle with respect to $\{e, e'\}$ in G if an only if $l(e)r(e') \in E(G)$. This implies, by Theorem 7.1, that $\{e, e'\}$ is a uniquely restricted matching in G if and only if $l(e)r(e') \notin E(G)$. This proves (b).

Finally, let us consider the case when $r(e') < r(e)$. Then we have $l(e) < l(e') < r(e') < r(e)$. Now, $l(e')$ is a left-vertex underneath e and $r(e')$ is a right-vertex underneath e, implying that we have the edges $l(e)r(e'), l(e')r(e) \in E(G)$. Now we have the cycle $l(e)r(e')l(e')r(e)l(e)$ in G, which is an alternating cycle with respect to $\{e, e'\}$ in G. By Theorem 7.1, $\{e, e'\}$ is not a uniquely restricted matching in G. This completes the proof of (c). ■

Lemma 7.10. *Let G be a bipartite permutation graph with a doubly-transitive vertex ordering $<$. Let $e_1, e_2, e_3 \in E(G)$ such that $l(e_1) \leq l(e_2) \leq l(e_3)$ and $r(e_1) \leq r(e_2) \leq r(e_3)$. If $\{e_1, e_3\}$ is not a uniquely restricted matching in G then neither $\{e_1, e_2\}$ nor $\{e_2, e_3\}$ is a uniquely restricted matching in G.*

Proof. We shall show only that $\{e_2, e_3\}$ is not a uniquely restricted matching, but the same kind of reasoning can be used to show that $\{e_1, e_2\}$ is also not a uniquely restricted matching. If $\{e_2, e_3\}$ is not even a matching, then we are immediately done. So let us suppose that $\{e_2, e_3\}$ is a matching. Thus, we have $l(e_2) < l(e_3)$ and $r(e_2) < r(e_3)$, which implies that $l(e_1) < l(e_3)$ and $r(e_1) < r(e_3)$. As $\{e_1, e_3\}$ is not a uniquely restricted matching, Lemma 7.9(a) tells us that we cannot have $r(e_1) < l(e_3)$. Therefore, it must be the case that $l(e_1) < l(e_3) < r(e_1) < r(e_3)$ (recall that $l(e_3) \neq r(e_1)$ as one is a left-vertex and the other a right-vertex), which by Lemma 7.9(b) means that $l(e_1)r(e_3) \in E(G)$. Note that the previous inequality means that $l(e_1) \leq l(e_2) < l(e_3) < r(e_1) \leq r(e_2) < r(e_3)$. Thus, the left-vertex $l(e_2)$ is underneath the edge $l(e_1)r(e_3)$, which tells us that $l(e_2)r(e_3) \in E(G)$. We can now apply Lemma 7.9(b) to the edges e_2 and e_3 to conclude that $\{e_2, e_3\}$ is not a uniquely restricted matching in G. ■

We now show that the statement of Theorem 7.2 also holds for bipartite permutation graphs.

Theorem 7.5. *Let $G = (V, E)$ be a bipartite permutation graph and let M be a matching in it. Then, the following statements are equivalent:*

(i) M is uniquely restricted

(ii) There is no alternating cycle of length 4 with respect to M in G

(iii) For any two edges $e, e' \in M$, $\{e, e'\}$ is a uniquely restricted matching in G

Proof. As *(ii)⇔(iii)* has already been noted in Observation 7.1, we shall only prove *(i)⇔(ii)*.

By Theorem 7.1, if M is a uniquely restricted matching, then there is no alternating cycle of any length with respect to M in G. So we only need to show that if M is not uniquely restricted, then there is an alternating cycle of length 4 with respect to M in G.

Let $<$ be a doubly-transitive vertex ordering of G. Suppose that the matching M is not uniquely restricted. Then, by Theorem 7.1, there exists some alternating cycle with respect to M in G. Let $u_1u_2u_3 \dots u_ku_1$ be an alternating cycle with respect to M of smallest possible length in G. Clearly, k is even and $4 \leq k \leq |V(G)|$. If $k = 4$, then this cycle is an alternating cycle of length 4 with respect to M and we are done. So let us assume that $k > 4$. We shall also assume without loss of generality that $u_1 = \min_<\{u_1, u_2, \dots, u_k\}$ and that $u_2 < u_k$ (otherwise we can relabel the vertices of the cycle to satisfy both these conditions). Note that this means that u_1 is a left-vertex and that both u_2 and u_k are right-vertices. Since set of left-vertices and set of right-vertices are both independent sets, we can see that u_i is a left-vertex if and only if i is odd. Now, let us examine the position of u_3 in the ordering $<$. As u_3 is a left-vertex and u_2 a neighbour of it, we must have $u_3 < u_2$. As $u_2 < u_k$, this means that u_3 is underneath the edge u_1u_k, which implies that $u_3u_k \in E(G)$. Then, at least one of the cycles $u_1u_2u_3u_ku_1$ or $u_3u_4u_5 \dots u_ku_3$ is an alternating cycle with respect to M in G having length smaller than k. This contradicts our assumption that $u_1u_2 \dots u_k$ is an alternating cycle with respect to M of smallest possible length in G. ■

Let M be a matching in a bipartite permutation graph with a doubly-transitive vertex ordering $<$. The edges of M can be labelled as $e_1, e_2, \dots, e_{|M|}$ such that $l(e_1) < l(e_2) < \cdots < l(e_{|M|})$. The matching M is then said to *start with* the edge e_1.

Theorem 7.6. *Let G be a bipartite permutation graph with a doubly-transitive vertex ordering $<$. Let $M = \{e_1, e_2, \dots, e_t\}$ be a matching in G where $l(e_1) < l(e_2) < \cdots < l(e_t)$. The matching M is uniquely restricted if and only if $\{e_i, e_{i+1}\}$ is a uniquely restricted matching in G, for each $i \in \{1, 2, \dots, t-1\}$.*

Proof. The proof this theorem closely follows the proof of Theorem 7.3. As every subset of a uniquely restricted matching is also a uniquely restricted matching, to prove the theorem, we only need to show that whenever M is not a uniquely restricted matching, there exists some $i \in \{1, 2, \dots, t-1\}$ such that $\{e_i, e_{i+1}\}$ is not a uniquely restricted matching. Suppose that M is not a uniquely restricted matching. Then, by Theorem 7.5, we know that there exists $e_p, e_q \in M$ with $1 \leq p < q \leq t$ such that $\{e_p, e_q\}$ is not a uniquely restricted matching. We choose p and q such that for any p', q' with $1 \leq p' < q' \leq t$ and $q' - p' < q - p$, $\{e_{p'}, e_{q'}\}$ is a uniquely restricted matching in G. Suppose that $q > p+1$. By our choice of p and q, we know that both $\{e_p, e_{q-1}\}$ and $\{e_{q-1}, e_q\}$ are uniquely restricted matchings. As we have $l(e_p) < l(e_{q-1}) < l(e_q)$, we know by Lemma 7.9(a) that $l(e_q) < r(e_p)$ and by Lemma 7.9(c), we have that $r(e_p) < r(e_{q-1})$ and $r(e_{q-1}) < r(e_q)$. We now have $l(e_p) < l(e_{q-1}) < l(e_q) < r(e_p) < r(e_{q-1}) < r(e_q)$. By Lemma 7.9(b), we can now say that $l(e_p)r(e_q) \in E(G)$ and that $l(e_{q-1})r(e_q) \notin E(G)$. But this is impossible as $l(e_{q-1})$ is now a left-vertex underneath the edge $l(e_p)r(e_q)$. Therefore, we can conclude that $q = p+1$. For $i = p$, we now have that $\{e_i, e_{i+1}\}$ is not a uniquely restricted matching, thereby completing the proof. ■

Corollary 7.2. *Let G be a bipartite permutation graph with a doubly-transitive vertex ordering $<$ and let M be a uniquely restricted matching in G starting with the edge $e' \in E(G)$. Let $e \in E(G)$ such that $l(e) < l(e')$ and $\{e, e'\}$ is a uniquely restricted matching in G. Then $\{e\} \cup M$ is a uniquely restricted matching in G starting with e.*

Proof. We shall first show that $\{e\} \cup M$ is a matching. As $\{e, e'\}$ is a uniquely restricted matching, it follows from Lemma 7.9(c) that $r(e) < r(e')$. For any edge $e'' \in M \setminus \{e'\}$, we have $l(e') < l(e'')$ as M starts with e' and therefore, from Lemma 7.9(c) and the fact that $\{e', e''\}$ is a uniquely restricted matching, we have $r(e') < r(e'')$. This tells us that for every edge $e'' \in M$, $r(e) < r(e'')$. Note that we also have $l(e) < l(e'')$ for every edge $e'' \in M$. Then, $l(e)$ and $r(e)$ are distinct from $l(e'')$ and $r(e'')$ for any edge $e'' \in M$ (recall that no vertex can be both a left-vertex and a right-vertex). This leads us to the conclusion that $\{e\} \cup M$ is a matching. The proof of the corollary now follows directly from Theorem 7.6. ■

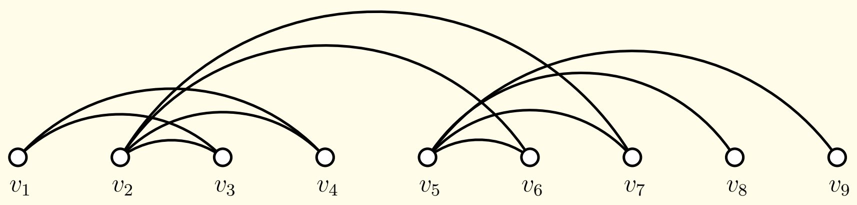
From here onwards, we assume that G is a bipartite permutation graph with no isolated vertices and having a doubly-transitive vertex ordering $<$. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$ where $v_1 < v_2 < \cdots < v_n$. For a vertex $u \in V(G)$, as in Section 7.3, we define $\lambda(u) = \min_<\{v \in N(u) \cup \{u\}\}$ and $\rho(u) = \max_<\{v \in N(u) \cup \{u\}\}$. Note that in a doubly-transitive vertex ordering, unlike in a proper vertex ordering, a vertex u may not be adjacent to all the vertices that come between $\lambda(u)$ and $\rho(u)$ in the ordering. For each left-vertex u, we further define $\gamma(u) = \min_<\{v \in N(u) \colon u < v\}$.

For every edge $e \in E(G)$, we now define a pair of edges $x(e)$ and $y(e)$ as follows. Let $\rho(l(e)) = v_i$. Then, we define $x(e) = \lambda(u)u$, where

$$u = \begin{cases} v_{i+1} & \text{if } v_{i+1} \text{ is a right-vertex} \\ \gamma(v_{i+1}) & \text{otherwise} \end{cases}$$

It is easy to see that $x(e)$ does not exist if and only if $\rho(l(e)) = v_n$ (recall that G has no isolated vertices). We define $y(e) = u\gamma(u)$ where $u = \min_<\{v \in V(G) \colon v > r(e)$ and v

is a left-vertex}. Clearly, $y(e)$ does not exist if and only if either $r(e) = v_n$ or if every $v > r(e)$ is a right-vertex (again, recall that G has no isolated vertices).



e	$x(e)$	$y(e)$
v_1v_3	v_2v_6	v_5v_6
v_2v_3	v_5v_8	v_5v_6
v_1v_4	v_2v_6	v_5v_6
v_2v_6	v_5v_8	-
v_5v_6	-	-

Figure 7.4: The bipartite permutation graph and its doubly-transitive ordering from Figure 7.3 is shown at the top and a table showing the values of $x(e)$ and $y(e)$ for some edges e (for this particular doubly-transitive ordering) are shown below it.

Observation 7.6. *Let $e \in E(G)$.*

(a) If $x(e)$ exists, then $l(e) < l(x(e))$ and $\{e, x(e)\}$ is a uniquely restricted matching.

(b) If $y(e)$ exists, then $l(e) < l(y(e))$ and $\{e, y(e)\}$ is a uniquely restricted matching.

Proof. We shall first prove (a). By definition of $x(e)$, we know that $\rho(l(e)) < r(x(e))$. As $r(e) \leq \rho(l(e))$, this implies that $l(e) < r(e) < r(x(e))$ and that $l(e)r(x(e)) \notin E(G)$. As $l(x(e))r(x(e)) \in E(G)$, this means that $l(x(e)) \neq l(e)$. If $l(x(e)) < l(e)$, then as we have $l(x(e)) < l(e) < r(x(e))$, the left-vertex $l(e)$ underneath the edge $l(x(e))r(x(e))$ has to be adjacent to $r(x(e))$, contradicting our previous observation. Therefore, we can conclude that $l(e) < l(x(e))$. If $r(e) < l(x(e))$, then by Lemma 7.9(a), we have that $\{e, x(e)\}$ is a uniquely restricted matching in G, and thus we are done. So let us assume that $l(x(e)) < r(e)$ (note that they cannot be equal as one is left-vertex and the other a right-vertex). We now have the inequality $l(e) < l(x(e)) < r(e) < r(x(e))$. Now since $l(e)r(x(e)) \notin E(G)$, Lemma 7.9(b) can be used to conclude that $\{e, x(e)\}$ is a uniquely restricted matching in G. This completes the proof of (a).

Next, we shall prove (b). From the definition of $y(e)$, it is clear that $r(e) < l(y(e))$ and therefore $l(e) < l(y(e))$. Furthermore, from Lemma 7.9(a), we get that $\{e, y(e)\}$ is a uniquely restricted matching in G, thus proving (b). ■

Lemma 7.11. *Let M be a uniquely restricted matching starting with an edge e in G such that $|M| \geq 3$. Then there exists a uniquely restricted matching M' that starts with either $x(e)$ or $y(e)$ in G such that $|M'| = |M| - 1$.*

Proof. Let $M = \{e = e_1, e_2, e_3, \ldots, e_t\}$, where $t \geq 3$ and $l(e_1) < l(e_2) < \cdots < l(e_t)$. From Lemma 7.9(c), we know that $r(e_1) < r(e_2) < \cdots < r(e_t)$.

Suppose that $l(e_2) < r(e)$. Since $\{e, e_2\}$ is a uniquely restricted matching, we have from Lemma 7.9(c) that $r(e) < r(e_2)$ and then further from Lemma 7.9(b) that $l(e)r(e_2) \notin E(G)$. This means that $r(e_2)$ cannot be underneath the edge $l(e)\rho(l(e))$, which implies that $\rho(l(e)) < r(e_2)$. Let $\rho(l(e)) = v_i$. Then, $v_{i+1} \leq r(e_2)$. Now suppose that $r(x(e)) > r(e_2)$. Then clearly, $r(x(e)) \neq v_{i+1}$. This can only mean that v_{i+1} is a left-vertex, which implies that $v_{i+1} < r(e_2)$, and that $r(x(e)) = \gamma(v_{i+1})$, which implies that $\gamma(v_{i+1}) > r(e_2)$ (as we have assumed that $r(x(e)) > r(e_2)$). Since we now have $v_{i+1} < r(e_2) < \gamma(v_{i+1})$, the vertex $r(e_2)$ is a right-vertex underneath the edge $v_{i+1}\gamma(v_{i+1})$, implying that $v_{i+1}r(e_2) \in E(G)$. But this contradicts our choice of $\gamma(v_{i+1})$. Therefore, we can conclude that $r(x(e)) \leq r(e_2)$. It is easy to see that by the definition of $x(e)$, we always have $r(e) \leq \rho(l(e)) < r(x(e))$. As $l(e_2) < r(e)$, we now have $l(e_2) < r(x(e)) \leq r(e_2)$. Then, $r(x(e))$ is a right-vertex underneath the edge e_2, which means that $l(e_2)r(x(e)) \in E(G)$. Since by definition of $x(e)$, we have $l(x(e)) = \lambda(r(x(e)))$, this implies that $l(x(e)) \leq l(e_2)$. Since we also have $r(x(e)) \leq r(e'_2)$, we can now apply Lemma 7.4 to the edges $x(e)$, e_2 and e_3 to conclude that $\{x(e), e_3\}$ is a uniquely restricted matching. Since $\{e_3, e_4, \ldots, e_t\}$ is a uniquely restricted matching and $l(x(e)) \leq l(e_2) < l(e_3)$, we can use Corollary 7.2 to conclude that $M' = \{x(e), e_3, e_4, \ldots, e_t\}$ is a uniquely restricted matching starting with $x(e)$ in G. As $|M'| = |M| - 1$, we are done.

Now let us suppose that $r(e) < l(e_2)$. As $l(e_2)$ is a left-vertex that comes after $r(e)$ in the ordering $<$, we know that $y(e)$ exists and that $l(y(e)) \leq l(e_2)$. It can be seen from the definition of $y(e)$ that $r(y(e)) = \gamma(l(y(e)))$. If $r(y(e)) > r(e'_2)$, then we have $l(y(e)) \leq l(e_2) < r(e_2) < r(y(e))$, and therefore $r(e_2)$ is a right-vertex underneath the edge $y(e)$, which implies that $l(y(e))r(e_2) \in E(G)$. But this contradicts the earlier observation that $r(y(e)) = \gamma(l(y(e)))$. We can therefore conclude that $r(y(e)) \leq r(e_2)$. We can now apply Lemma 7.4 to the edges $y(e)$, e_2 and e_3 to conclude that $\{y(e), e_3\}$ is a uniquely restricted matching. Since $\{e_3, e_4, \ldots, e_t\}$ is a uniquely restricted matching and $l(y(e)) \leq l(e_2) < l(e_3)$, we can use Corollary 7.2 to conclude that $M' = \{y(e), e_3, e_4, \ldots, e_t\}$ is a uniquely restricted matching starting with $y(e)$ in G. The proof is concluded by noting that $|M'| = |M| - 1$. ■

We shall now define a set $U(e)$ of edges for every edge $e \in E(G)$ as follows.

$$U(e) = \begin{cases} \{e\} & \text{if neither } x(e) \text{ nor } y(e) \text{ exists} \\ \{e\} \cup U(x(e)) & \text{if } y(e) \text{ does not exist, or if both exist} \\ & \text{and } |U(x(e))| \geq |U(y(e))| \\ \{e\} \cup U(y(e)) & \text{if } x(e) \text{ does not exist, or if both exist} \\ & \text{and } |U(x(e))| < |U(y(e))| \end{cases}$$

From Observation 7.6, we know that $l(e) < l(x(e))$ and $l(e) < l(y(e))$, which implies that $U(e)$ is well-defined. The next two lemmas will show that $U(e)$ is always a uniquely restricted matching starting with e and that it has the maximum possible cardinality among all the uniquely restricted matchings starting with e in G.

Lemma 7.12. *For any edge $e \in E(G)$, $U(e)$ is a uniquely restricted matching starting with e in G.*

Proof. We shall prove this by induction on $|U(e)|$. If $|U(e)| = 1$, then it must be the case that $U(e) = \{e\}$. In this case, the statement of the lemma is clearly true. Now let us assume that $|U(e)| > 1$ and that the statement of the lemma is true for all $e' \in E(G)$ such that $|U(e')| < |U(e)|$. As $|U(e)| > 1$, we know that at least one of $x(e), y(e)$ exists. From the definition of $U(e)$, it can be seen there are only two possibilities: $x(e)$ exists and $U(e) = \{e\} \cup U(x(e))$, or $y(e)$ exists and $U(e) = \{e\} \cup U(y(e))$. From the induction hypothesis, $U(x(e))$ is a uniquely restricted matching starting with $x(e)$ and $U(y(e))$ is a uniquely restricted matching starting with $y(e)$. It now follows from Observation 7.6 and Corollary 7.2 that $U(e)$ is a uniquely restricted matching starting with e. ■

Lemma 7.13. *Let M be a uniquely restricted matching starting with e in G. Then $|M| \leq |U(e)|$.*

Proof. We will use induction on $|U(e)| = k$ to prove this. Suppose first that $k = 1$. Then $U(e) = \{e\}$, which can be the case only if neither $x(e)$ nor $y(e)$ exist. As $x(e)$ does not exist, we have $\rho(l(e)) = v_n$. In this case, for an edge $e' \neq e$ in M, we must have $r(e')$ underneath the edge $l(e)\rho(l(e))$, implying that $l(e)r(e') \in E(G)$. As $\{e, e'\}$ is a uniquely restricted matching, from Lemma 7.9(b) and Lemma 7.9(c), we have $r(e) < l(e')$. This tells us that $r(e) \neq v_n$ and that there exists a left-vertex $l(e') > r(e)$. But this contradicts the fact that $y(e)$ does not exist. We can therefore conclude that e' does not exist, or in other words, $M = \{e\}$, thereby proving the statement of the lemma. We shall now assume that $k > 1$ and that for any edge $e' \in E(G)$ such that $|U(e')| < k$, the statement of the lemma is true.

Let us assume for the sake of contradiction that $|M| > k$. From Lemma 7.12, we know that $U(e)$ is a uniquely restricted matching starting with e in G. Note that as $k > 1$, we have $|M| \geq 3$. By Lemma 7.11, we can now infer that there exists a uniquely restricted matching M' in G with $|M'| = |M| - 1$ such that M' starts with either $x(e)$ or $y(e)$. From the definition of $U(e)$, it can be seen that $|U(e)| \geq \max\{|U(x(e))|, |U(y(e))|\} + 1$. Since $|U(e)| = k$, we can therefore infer that $|U(x(e))| \leq k - 1$ and $|U(y(e))| \leq k - 1$. We can now apply the induction hypothesis on the starting edge of M' (which is either $x(e)$ or $y(e)$) to conclude that $|M'| \leq k - 1$. But as $|M'| = |M| - 1$, we now get $|M| \leq k$, which contradicts our assumption that $|M| > k$. ■

Lemma 7.14. *$U(v_1\gamma(v_1))$ is a uniquely restricted matching of maximum cardinality in G.*

Proof. By Lemma 7.12, it is clear that $U(v_1\gamma(v_1))$ is a uniquely restricted matching starting with $v_1\gamma(v_1)$ in G. Let $\{e_1, e_2, \ldots, e_k\}$ be any uniquely restricted matching in G where $l(e_1) < \cdots < l(e_k)$. Clearly, $v_1 \leq l(e_1) < l(e_2)$. As $\{e_1, e_2\}$ is a uniquely restricted matching, we know from Lemma 7.9(c) that $r(e_1) < r(e_2)$. If $r(e_1) < \gamma(v_1)$, then $r(e_1)$ is a right-vertex underneath the edge $v_1\gamma(v_1)$, implying that $v_1 r(e_1) \in E(G)$. But this would be a contradiction to the choice of $\gamma(v_1)$. It must therefore be the case that $\gamma(v_1) \leq r(e_1)$. We now have $v_1 \leq l(e_1) < l(e_2)$ and $\gamma(v_1) \leq r(e_1) < r(e_2)$. We can now apply Lemma 7.10 to the edges $v_1\gamma(v_1)$, e_1 and e_2 to conclude that $\{v_1\gamma(v_1), e_2\}$ is a uniquely restricted matching in G. By Corollary 7.2, we now have that $\{v_1\gamma(v_1), e_2, e_3, \ldots, e_k\}$ is a uniquely restricted matching in G starting with $v_1\gamma(v_1)$. As the cardinality of this matching is k, we have by Lemma 7.13 that $|U(v_1\gamma(v_1))| \geq k$. ■

Theorem 7.7. *There is a linear-time algorithm that given a bipartite permutation graph as input, computes a maximum cardinality uniquely restricted matching in it.*

Proof. We can construct a linear-time algorithm along the lines of the proof of Theorem 7.4. We first remove isolated vertices from G and then use one of the known linear-time algorithms to generate a doubly-transitive vertex ordering $<$ of $V(G)$ (for example, [HH04]). In a single pass through the adjacency list that takes time $O(n+m)$, every vertex in G can be marked as a left-vertex or right-vertex and the values $\lambda(u)$, $\rho(u)$ for each vertex $u \in V(G)$ and the value $\gamma(u)$ for each left-vertex u can be computed. The algorithm further computes for all $u \in V(G)$, a value $\nu(u) = \min_<\{v \in V(G) \colon v > u$ and v is a left-vertex$\}$ using a single pass in the backward direction through the vertex ordering $<$, taking $O(n)$ time. It is not hard to see that once this is done, the values $x(e)$ and $y(e)$ for an edge $e \in E(G)$ can be computed in $O(1)$ time. Then, a dynamic programming algorithm very similar to the one from the proof of Theorem 7.4 can be used to compute $U(e)$ for every edge $e \in E(G)$, in $O(n+m)$ time. Finally, the algorithm returns $U(v_1\gamma(v_1))$. The correctness of the algorithm follows from Lemma 7.12 Lemma 7.13 and Lemma 7.14 and the algorithm clearly runs in $O(n+m)$ time. ■

Summary

In this chapter, we give linear-time algorithms for finding maximum cardinality uniquely restricted matchings in proper interval graphs and bipartite permutation graphs.

CHAPTER 8

Balanced Connected Subgraph

Contents

Related Publications:

1. Sujoy Bhore, Sourav Chakraborty, Satyabrata Jana, Joseph SB Mitchell, Supantha Pandit, and Sasanka Roy. "The Balanced Connected Subgraph Problem." In *Discrete Applied Mathematics*, 2021.

2. Sujoy Bhore, Satyabrata Jana, Supantha Pandit, and Sasanka Roy. "Balanced connected subgraph problem in geometric intersection graphs." In *International Conference on Combinatorial Optimization and Applications*, pp. 56-68. Springer, 2019.

3. Sujoy Bhore, Sourav Chakraborty, Satyabrata Jana, Joseph SB Mitchell, Supantha Pandit, and Sasanka Roy. "The balanced connected subgraph problem." In *Conference on Algorithms and Discrete Applied Mathematics*, pp. 201-215. Springer, 2019.

8.1 Introduction

Given a graph $G = (V,E)$, and a subset of vertices $U \subset V$, the induced subgraph $G[U]$ is the graph whose vertex set is U and whose edge set consists of all of the edges in E that have both endpoints in U. A plethora of problems in graph theory and combinatorial optimization involve determining if a given graph G has an induced subgraph with certain properties. Some of the related optimization problems include finding cliques, independent sets, connected dominating sets, connected vertex cover, induced paths, cycles, matchings, etc.

In this chapter, we study the problem in which we are given a simple connected graph $G = (V,E)$ where each vertex in V is colored with either "*red*" or "blue" (note, the color assignment might not be a proper 2-coloring of the vertices, i.e., we allow vertices of the same color to be adjacent in G). Sometime, we call it as red-blue graph.

We seek a maximum-cardinality subset $V' \subseteq V$ of the vertices such that V' is color-balanced, i.e. having same number of red and blue vertices in V', and such that the induced subgraph H by V' in G is connected. We formally define the problem as follows.

Balanced Connected Subgraph (BCS) Problem: Given graph $G = (V, E)$, with vertex set $V = V_R \cup V_B$ partitioned into red vertices (V_R) and blue vertices (V_B),

➩ find a maximum-cardinality $V' \subseteq V$ such that $G[V']$ is connected and V' conatins equal number of red and blue vertices.

We can assume that G is connected as if it is not, we consider the problem separately in each component to find a maximum cardinality balanced connected subgraph in each of them and then take the one which has the largest cardinality.

8.1.1 Our contributions

We have considered the BCS problem on various graph classes including some geometric intersection graphs. We show that,

- Hardness on for planar graphs, bipartite graphs, chordal graphs, unit disk graphs, outerstring graphs, complete grid graphs, and unit square graphs.
- The existence of a balanced connected subgraph containing a specific vertex is NP-Complete.
- Finding the maximum balanced path in a graph is NP-Hard.
- Polynomial-time algorithms for trees, split graphs, bipartite graphs with a proper 2-coloring, graphs with diameter 2 , interval graphs, circular-arc graphs and permutation graphs.
- The BCS problem is fixed-parameter tractable for general graphs ($2^{O(k)}n^2 \log n$) while parameterized by number of vertices in a balanced connected subgraph.

8.2 Hardness Results

8.2.1 Bipartite graphs

In this section we prove that the BCS problem is NP-Hard for bipartite graphs with a general red/blue color assignment, not necessarily a proper 2-coloring. We give a reduction from the Exact-Cover-by-3-Sets (EC3Set) problem [GJ79a]. In this EC3Set problem, we are given a set U with $3k$ elements and a collection S of m subsets of U such that each $s_i \in S$ contains exactly 3 elements. The objective is to find an exact cover for U (if one exists), i.e., a sub-collection $S' \subseteq S$ such that every element of U occurs in exactly one member of S'. During the reduction, we generate an instance $G = (R \cup B, E)$ of the BCS problem from an instance $X(S,U)$ of the EC3Set problem as follows:

Reduction. For each set $s_i \in S$, we take a blue vertex $s_i \in B$. For each element $u_j \in U$, we take a red vertex $u_j \in R$. Now consider a set $s_i \in S$ containing three elements, u_α, u_β, and u_γ, and add the three edges (s_i, u_α), (s_i, u_β), and (s_i, u_γ) to the edge set E. Additionally, we consider a path of $5k$ blue vertices starting and ending with vertices b_1 and b_{5k}, respectively. Similarly, we consider a path of $3k$ red vertices starting and ending with vertices r_1 and r_{3k}, respectively. We connect these two paths by joining the vertices r_{3k} and b_1 by an edge. Finally, we add edges connecting each vertex s_i with b_{5k}. This completes the construction. See Figure 8.1 for the complete construction. Clearly, the numbers of vertices and edges in G are polynomial in terms of the numbers of elements and sets in X; hence, the construction can be done in polynomial time. We now prove the following lemma.

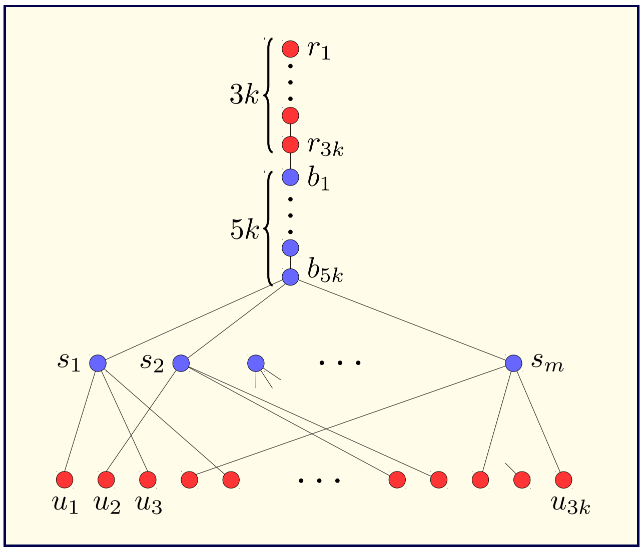


Figure 8.1: Construction of the instance G of the BCS problem.

Lemma 8.1. *The instance X of the* EC3Set *problem has a solution if and only if the instance G of the* BCS *problem has a balanced connected subgraph T with $12k$ vertices ($6k$ red and $6k$ blue).*

Proof. Assume that the EC3Set problem has a solution. Let S^* be an optimal solution in it. We choose the corresponding vertices of S^* in T. Since this solution covers all

u_j's. So we select all u_j's in T. Finally we select all the $5k$ blue and $3k$ red vertices in T, resulting in a total of $6k$ red and $6k$ blue vertices.

On the other hand, assume that there is a balanced tree T in G with $6k$ vertices of each color. The solution must pick the $5k$ blue vertices $b_1,\ldots,b_{5k}$. Otherwise, it exclude the $3k$ red vertices $r_1,\ldots,r_{3k}$, and reducing the size of the solution. Since the graph G has at most $6k$ red vertices, at most k vertices can be picked from the set $s_1,\ldots,s_m$ and need to cover all the $3k$ red vertices corresponding to u_j for $1 \le j \le 3k$. Hence, this k sets give an exact cover. ■

It is easy to see that the graph we constructed from the (EC3Set) problem in Figure 8.1 is indeed a bipartite graph. Hence we conclude the following theorem.

Theorem 8.1. *The* BCS *problem is* NP-Hard *for bipartite graphs.*

8.2.2 Planar graphs

In this section we prove that BCS problem is NP-Hard for planar graphs. We give a reduction from the Steiner Tree problem in planar graphs (STPG) [GJ79a]. In this problem, we are given a planar graph $G = (V,E)$, a subset $X \subseteq V$, and a positive integer $k \in \mathbb{N}$. The objective is to find a tree $T = (V',E')$ with at most k edges such that $X \subseteq V'$. Without loss of generality we assume that $k \ge |X| - 1$, otherwise the STPG problem has no solution.

Reduction. We generate an instance $H = (R \cup B, E(H))$ for the BCS problem

from an instance $G = (V,E)$ of the STPG problem. We color all the vertices, V, in G as blue. We create a set of $|X|$ red vertices as follows: for each vertex $u_i \in X$, we create a red vertex u'_i in H, and we connect u'_i to u_i via an edge. Additionally, we take a set Z of $(k+1-|X|)$ red vertices in H and the edges (z_j, u'_1) into $E(H)$, for each $z_j \in Z$. Hence we have, $B = V$, and $R = Z \cup \{u'_i;\ 1 \le i \le |X|\}$. Note that $|R| < |B|$ and $|R| = (k+1)$. This completes the construction. For an illustration see Figure 8.2. Clearly the number of vertices and edges in H are polynomial in terms of vertices in G. Hence the construction can be done in polynomial time. We now prove the following lemma.

Lemma 8.2. *The* STPG *problem has a solution if and only if the instance H of the* BCS *problem has a balanced connected subgraph with $(k+1)$ vertices each of the two colors.*

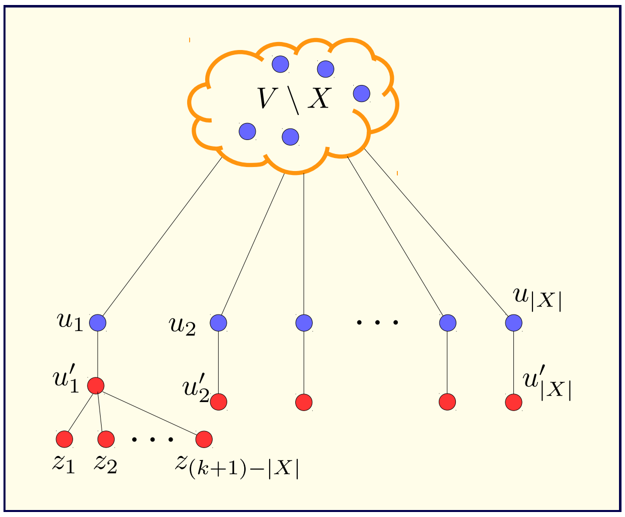


Figure 8.2: Schematic construction for planar graphs.

Proof. Assume that STPG has a solution. Let $T = (V', E')$ be the resulting Steiner tree, which contains at most k edges and $X \subseteq V'$. If $|V'| = (k+1)$ then the subgraph of H induced by $(V' \cup R)$ is connected and balanced with $(k+1)$ vertices of each color. If $|V'| < (k+1)$ then we take a set Y of $((k+1) - |V'|)$ many vertices from V such that the subgraph of G induced by $(V' \cup Y)$ is connected. Clearly $|V' \cup Y| = (k+1)$. Now the subgraph of H induced by $(V' \cup Y \cup R)$ is connected and balanced with $(k+1)$ vertices of each red and blue color.

On the other hand, assume that there is a balanced connected subgraph H' of H with $(k+1)$ vertices of each color. Note that, except vertex u'_1, in H all the red vertices are of degree 1 and connected to blue vertices. Let G' be the subgraph of G induced by all blue vertices in H'. Since H is connected and there is no edge between any two red vertices, G' is connected. Since G' contains $(k+1)$ vertices, any spanning tree T of H' contains k edges. So T is a solution of the STPG problem. ■

Theorem 8.2. *The* BCS *problem is* NP-Hard *for planar graphs.*

8.2.3 Chordal graphs

In this section we prove that the BCS problem is NP-Hard when the input graph is a chordal graph. The hardness construction is similar to the construction in Section 8.2.1; we modify the construction so that the graph is chordal. In particular, we add edges between s_i and s_j for each $i \neq j, 1 \leq i, j \leq m$. For this modified graph, it is easy to see that a lemma identical to Lemma 8.1 holds. Hence, we conclude that the BCS problem is NP-Hard for chordal graphs.

8.2.4 Grid graphs

We prove that the BCS problem is NP-Hard for grid graphs. To prove the NP-Hardness, we give a reduction from the Rectilinear Steiner Tree (RST) problem that is known to be NP-Hard [GJ77]. A *Rectilinear Steiner Tree* for a finite point-set P in the plane is a tree that connects the points in P using only horizontal and vertical segments. In the RST problem, we are given a set P of integer-coordinated points in the plane and an integer L. The objective is to find a rectilinear Steiner tree T (if one exists) of length at most L that connects the points in P, here the length of the tree is defined as the sum of all edge-lengths.

In [BCJ$^+$19], the NP-Hardness of the BCS problem for planar graphs is shown by giving a reduction from the Steiner tree problem on planar graphs. The theme of that reduction is basically carefully adding additional vertices to the given instance of the Steiner tree problem maintaining the planarity (actually, in that reduction, we added a star graph to a vertex, however, adding a path does not make any difference in the proof). Here also we use the same theme i.e., adding a path to a vertex in the given instance of the RST problem, however, here we are careful about maintaining the grid structure.

Reduction During the reduction, we first generate a bicolored grid graph (an instance of the BCS problem in grid graph) from an instance $X(P,L)$ of the RST problem. The construction is done in three steps. In Step 1, we first generate a rectangular grid graph D. Next, in Step 2, we add a path δ to D. In Step 3, we make the graph $D \cup \delta$ a rectangular grid graph by adding additional vertices.

Step 1: Generating a rectangular grid graph D

Suppose we have an instance $X(P,L)$ of the RST problem. For any point $p \in P$, let $x(p)$ and $y(p)$ denote the x- and y-coordinates of p, respectively. Let p_t, p_b, p_l, and p_r be the topmost (largest y-coordinate), bottommost (smallest y-coordinate), leftmost (smallest x-coordinate), and rightmost (largest x-coordinate) points in P, respectively (tie is broken by choosing any point arbitrarily). We now take a unit integer rectangular grid graph D on the plane such that the coordinates of the lower-left grid vertex is $(x(p_l), y(p_b))$ and upper-right grid vertex is $(x(p_r), y(p_t))$. Then we associate each point p in P with a grid vertex d_p in D having the same x- and y-coordinates of p. After that we assign colors to the points in D. The vertices of D that correspond to points in P are colored red and the remaining grid vertices in D are colored blue.

Step 2: Adding a path δ to D

Observe that if there exists a Steiner tree T of length (say L) $|P|-1$, then T does not include any blue vertex in D. Further, if there exists a Steiner tree T of length (say L) $2|P|-1$ then T contains equal number of red and blue vertices in D. Based on this observation, we consider two cases to add some additional vertices (not necessarily forming a grid structure) to D.

Case 1. [$\boldsymbol{L \geq 2|P|-1}$]: In this case the number of blue vertices in a Steiner tree T (if exists) is more than or equals to the number of red vertices in D. We consider a path δ of $(L-2|P|+1)$ red vertices starting and ending with vertices r_1 and $r_{L-2|P|+1}$, respectively. The position of r_i is $(x(p_l)-i, y(p_l))$, for $1 \leq i \leq L-2|P|+1$. We connect this path with D using an edge between the vertices r_1 and p_l. See Figure 8.2.4 for an illustration of this construction. Let the resulting graph be $H_1 = D \cup \delta$.

Case 2. [$\boldsymbol{L < 2|P|-1}$]: In this case the number of red vertices in a Steiner tree T (if exists) is more than the number of blue vertices in D. We consider a path δ of $(2|P|-L)$ blue vertices $b_1, b_2, b_{2|P|-L}$ and a red vertex r'. The position of b_i is $(x(p_l)-i, y(p_l))$, for $1 \leq i \leq 2|P|-L$ and the position of r' is $(x(p_{2|P|-L})-1, y(p_l))$. b_i is connected to b_i+1, for $1 \leq i \leq 2|P|-L-1$ and $b_{2|P|-L}$ is connected to r'. We connect this path δ with D using an edge between the vertices b_1 and p_l.

See Figure 8.2.4 for an illustration of this construction. Let the resulting graph be $H_2 = D \cup \delta \cup \{r'\}$

In Step 3, we make the graph $D \cup \delta$ a rectangular grid graph by adding additional vertices.

Step 3: making the graph $D \cup \delta$ (i.e., either H_1 or H_2) a rectangular grid graph

We add some vertices and edges in H_1(or H_2) to form a grid graph as follows. Consider the graph H_1. Notice that H_1 has two components D and δ. D is a grid graph. We add δ to it to the left of D. We now add blue vertices at each integer coordinates (r,s), where $(r,s) \notin \delta, (x(p_l)-L+2|P|-1) \leq r < x(p_l)$ and $y(p_b) \leq s \leq y(p_t)$. Let the resulting graph be G_1 and clearly, G_1 becomes a grid graph. Similarly, we can make H_2 a grid graph G_2 by adding only blue vertices.

This completes the construction. Clearly, the construction (either G_1 or G_2) can be done in polynomial time. Now we prove the following lemma. For a graph G, $V(G)$ denotes all the vertices in G. Let $Y = G_1 \setminus D$ (resp, $G_2 \setminus D$). Also let P_{G_1} (resp, P_{G_2}) is the set of red vertices corresponding to P in G_1 (resp, G_2).

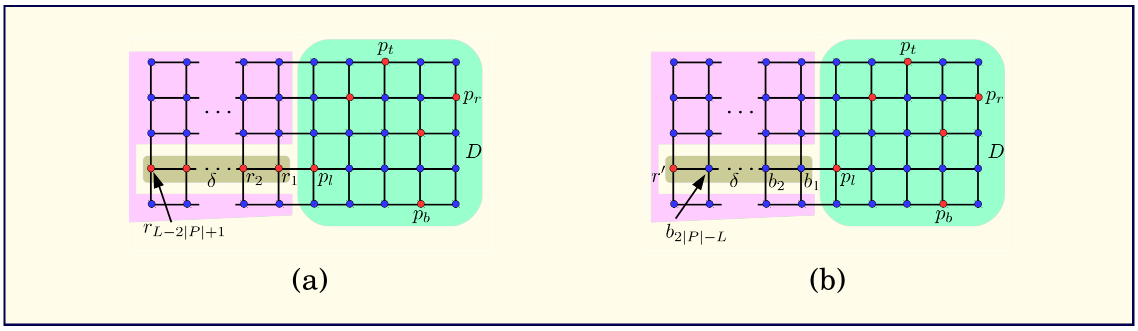


Figure 8.3: (a) Construction of the instance G_1. (b) Construction of the instance G_2.

Lemma 8.3. *Let H be a connected subgraph of G_1 (resp, G_2) containing P_{G_1} (resp, P_{G_2}). We can find a subgraph H' in G_1 (resp, G_2) such that $G_1[V(H')\setminus Y]$ (resp, $G_2[V(H')\setminus Y]$) has a connected component containing P_{G_1} (resp, P_{G_2}) and $|V(H')| \leq |V(H)|$.*

Proof. We prove the lemma for G_1, a similar proof holds for G_2. If $G_1[V(H)\setminus Y]$ has a connected component containing P_{G_1} then we are done. Else there is a pair of vertices, say u and v in P_{G_1} such that each path connecting u with v passes through some vertices in Y. Let $\mathscr{P}$ be a path between u and v. And let for a pair of vertices a and b, $< a, b >$ and (a, b) denotes a path and the edge between a and b. Now there exists four vertices say v_1, v_2, v_3, v_4 in $\mathscr{P}$ such that $\mathscr{P}$ is a concatenation of five subpaths: $< u, v_1 >, (v_1, v_2), < v_2, v_3 >, (v_3, v_4), < v_4, v >$, where $v_1, v_4 \in D$, $v_2, v_3 \in Y$. Clearly, $x(v_1) = x(v_4)$, $x(v_2) = x(v_3)$. Now there is exactly one shortest path from v_1 to v_4 in D. Let this shortest path consists of the set Z of vertices. Let $W = \mathscr{P} \cap Y$. Let $H_1 = G_1[(V(H)\setminus W) \cup Z]$. If $G_1[V(H_1)\setminus Y]$ has a connected component containing P_{G_1} then we are done with $H' = H_1$ as $|V(H_1)| \leq |V(H)|$. Else there is a pair of vertices, such that each path connecting them passes through some vertices in Y. Then we repeat the procedure until there is no such pair. ■

Lemma 8.4. *The instance X of the* RST *problem has a rectilinear Steiner tree T of length at most L if and only if*

- ***For Case 1:*** *the instance G_1 has a balanced connected subgraph H with at least $2(L - |P| + 1)$ vertices.*

- ***For Case 2:*** *the instance G_2 has a balanced connected subgraph H with at least $2(|P| + 1)$ vertices.*

Proof. We prove this lemma for Case 1. The proof of Case 2 is similar.

For Case 1: Assume that X has a Steiner tree T of length at most L, where $L \geq 2|P|-1$. Let U be the set of those vertices in G_1 corresponds to the vertices in T. Clearly, U contains $|P|$ red vertices and $L-|P|+1$ blue vertices. Since $L \geq 2|P|-1$, to make U balanced it needs $L-|P|+1-|P|$ additional red vertices. So we can add the $L-2|P|+1$ red vertices on the path δ to U. Therefore, $U \cup \delta$ becomes connected and balanced (contains $L-|P|+1$ vertices in each color).

On the other hand, assume that there is a balanced connected subgraph H in G with at least $(L-|P|+1)$ vertices of each color. The number of red vertices in G_1 is exactly $(L-|P|+1)$. So H must pick $(L-|P|+1)$ blue vertices that connect the vertices in G_1 corresponding to P. Now H is a connected subgraph containing all vertices corresponding to P. Applying Lemma 8.3, we can a find a subgraph H' in G such that $G[V(H') \setminus Y]$ has a connected component containing P and $|V(H')| \leq |V(H)|$. As $|V(H)| = 2(L-|P|+1)$ so $|V(H')| \leq 2(L-|P|+1)$. Also $|Y| \geq (L-2|P|+1)$. Therefore the number of vertices in $G[V(H') \setminus Y]$ is at most $L+1$. As $G[V(H') \setminus Y]$ has a connected component containing all vertices corresponding to P, so X has a Steiner tree of length at most L. ∎

8.2.5 Unit disk graphs

In this section we focus on the BCS problem on unit-disk graphs. In [BCJ+19], it is shown that this problem is NP-Hard on planar graphs. It is noted that every planar graph can be represented as a disk graph (due to Koebe's kissing disk embedding theorem, see [**?**]). Therefore, the NP-Hardness of the BCS problem on disk graphs directly follows. Here, we show that the BCS problem remains NP-Hard even on the unit-disk graphs.

In Section 8.2.4, we show that the BCS problem is NP-Hard for grid graphs. During the construction in Section 8.2.4, we generate an instance, either the graph G_1 or G_2 for the BCS problem in grid graphs. We now show that either G_1 or G_2 is a unit-disk graph. Let us consider the graph G_1. For each vertex v in G_1, we take a unit disk whose radius is $\frac{1}{2}$ and center is on the vertex v. Therefore from the Lemma 8.4, we conclude that,

Theorem 8.3. *The* BCS *problem is* NP-Hard *for the unit-disk graphs.*

8.2.6 Unit square graphs

We show that the BCS problem is NP-Hard for the unit-square graphs. We have the result that the BCS problem is NP-Hard for the grid graphs. In the reduction in

Section 8.2.4, either the graph G_1 or G_2 is generated from an instance of the RST problem. Now the only thing we need to show that both the graphs G_1 and G_2 are intersection graphs of unit squares.

Let us consider the graph G_1. For each grid vertex v in G_1, take an axis-parallel square and rotate it 45° with the x-axis. The side length of this square is $\frac{1}{\sqrt{2}}$ and whose center is on v. Finally, we rotate the complete construction by an angle of $-45°$ and scale it by a factor of $\sqrt{2}$. This makes the squares axis-parallel and unit side length. It is not hard to verify that G_1 is an intersection graph of this set of axis-parallel unit squares. Hence we conclude:

Theorem 8.4. *The* BCS *problem is* NP-Hard *for axis-parallel unit-square graphs.*

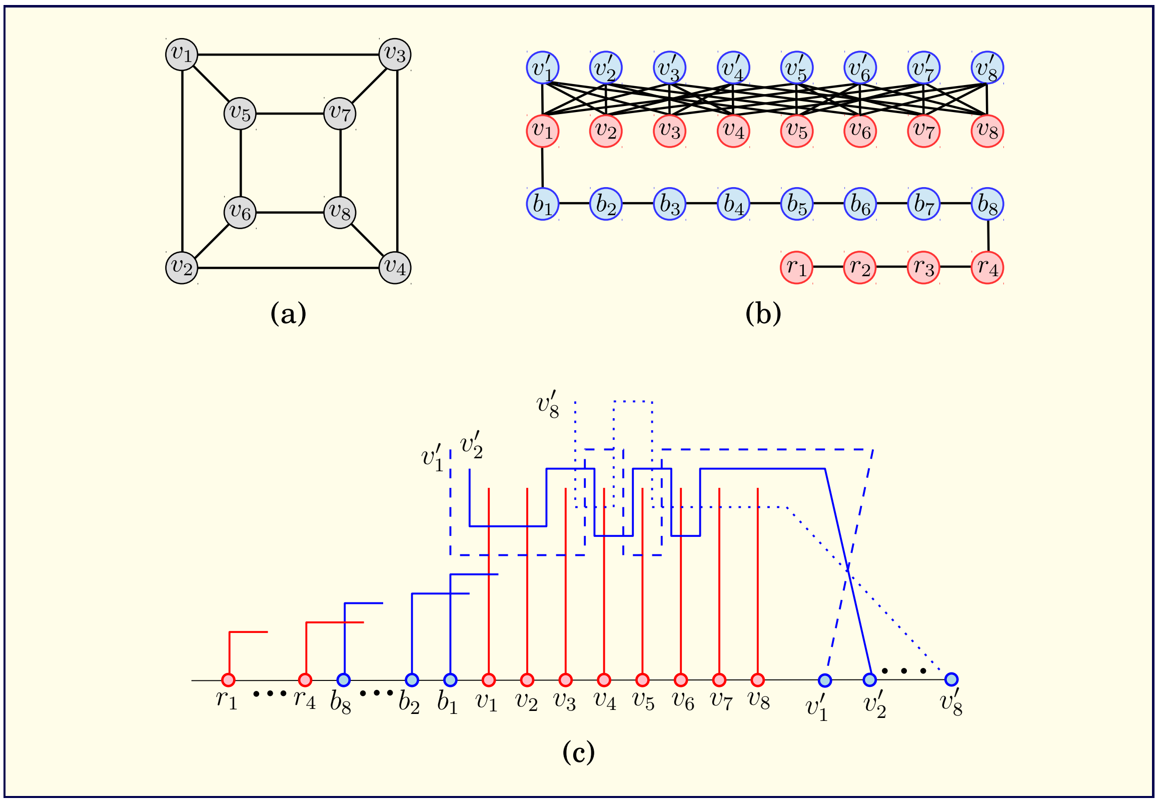
8.2.7 Outerstring graphs

A graph is said to be an outerstring graph if it has an intersection model consisting of curves that lie in a common half-plane and have one endpoint on the boundary of that half-plane. We study the BCS problem on string graphs. We know that finding a balanced subgraph on planar graphs is NP-Hard. Now, every planar graph can be represented as a string graph(by drawing a string for each vertex that loops around the vertex and around the midpoint of each adjacent edge). So NP-Hardness of BCS problem holds for string graphs. Below we show that this problem remains NP-Hard even for outerstring graphs. We give a reduction from the dominating set problem which is known to be NP-Complete on general graphs [KYK80], the problem is defined as follows. Given a graph $G = (V, E)$ and a positive integer k, the *dominating set problem* asks whether there exists a set $U \subseteq V$ such that $|U| \leq k$ and $N[U] = V$, where $N[U]$ denotes neighbours of U in G including U itself.

During the reduction, we first generate a bicolored geometric intersection graph $H = (R \cup B, E')$ from an instance $X(G, k)$ of the dominating set problem on general graph, where R and B denotes the set of red and blue vertices, respectively in H. Next, we show that H is an outerstring graph.

Reduction. Let $G = (V, E)$ be graph with vertex set $V = \{v_1, v_2, \ldots, v_n\}$. R consists of a copy of $V(G)$, denoted by $\{v_i, 1 \leq i \leq n\}$ and k new vertices $\{r_i, 1 \leq i \leq k\}$. Similarly, B contains a second copy of $V(G)$, denoted by $\{v'_i, 1 \leq i \leq n\}$ and n new vertices $\{b_i, 1 \leq i \leq n\}$. The edges in E' are defined in following way. For each edge $(v_i, v_j) \in E$, we add two edges $(v_i, v'_j), (v'_i, v_j)$ in E'. Next, we add two paths, one is of length k starting at r_1 and ending at r_k, other is of length n starting at b_1 and ending at

b_n. We join the edges $(b_n, r_k), (b_1, v_1)$ into E'. Then we add edges between all pair of vertices in $\{v'_1, v'_2, \dots v'_n\}$. Our construction ends with adding n edges (v_i, v'_i) into E' for $1 \le i \le n$. This completes the construction. See Figure 8.4 for an illustration of this construction that can be made in polynomial time.



v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
(a)
v'_1 v'_2 v'_3 v'_4 v'_5 v'_6 v'_7 v'_8
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8
r_1 r_2 r_3 r_4
(b)
v'_8
v'_1 v'_2
r_1 r_4 b_8 b_2 b_1 v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v'_1 v'_2 v'_8
(c)

Figure 8.4: (a) A graph G. (b) Construction of H from G with $k = 4$. For the clarity of the figure we omit the edges between each pair of vertices v'_i and v'_j, for $i \ne j$. (c) Intersection model of H.

Lemma 8.5. *The instance X has a dominating set of size k if and only if H has a balanced connected subgraph T with $2(n+k)$ vertices.*

Proof. Assume that G has a dominating set U of size k. Now we take the subgraph T of H induced by $\{v'_i : v_i \in U\}$ along with the vertices $\{r_i : 1 \le i \le k\} \cup \{b_j : 1 \le j \le n\} \cup \{v_i : 1 \le j \le n\}$ in H. Now clearly H is connected and balanced with $2(n+k)$ vertices.

On the other hand, assume that there is a balanced connected tree T in H with $(n+k)$ vertices of each color. The number of red vertices in H is exactly $(n+k)$. So the solution must pick the blue vertices $\{b_i : 1 \le i \le n\}$ that connect v_1 with r_k. As T has exactly $(n+k)$ blue vertices then it T should pick exactly k vertices from the set $\{v'_i : 1 \le i \le n\}$.

Since $v_1,\dots,v_n$ is an independent set contained in the balanced connected tree T, so the set of vertices in V corresponding to these k vertices gives us a dominating set of size k in G. ■

We now verify that H is an outerstring graph. For this, we provide an intersection model of it consisting of curves that lie in a common half-plane. For an illustration see Figure 8.4(c). We draw a horizontal line $y = 0$. For each vertex $v_i \in H$, draw the line segment $L_i = \overline{(i,0)(i,1)}$. For each vertex $v'_j \in H$, we draw a curve C_j, having one endpoint on the line $y = 0$, in such a way that for each edge $(v'_j, v_i) \in H$, C_j bend above L_i (the line segment corresponding to v_i) and intersects the lines L_i from top. Also all the curves C_j's intersect each other. Now we add the curves corresponding to $\{r_i : 1 \le i \le k\} \cup \{b_j : 1 \le j \le n\}$ having one endpoint on the line $y = 0$ with satisfying the adjacency. Finally, using Lemma 8.5, we conclude the following theorem.

Theorem 8.5. *The* BCS *problem is* NP-Hard *for the outerstring graphs.*

8.3 Hardness: BCS Problem with a Specific Vertex

In this section we prove that the existence of a balanced subgraph containing a specific vertex is NP-Complete. We call this problem the BCS-existence problem. The reduction is similar to the reduction used in showing the NP-Hardness of the BCS problem; we also use here a reduction from the EC3Set problem (see Section 8.2.1 for the definition).

Reduction. Assume that we are given a EC3Set problem instance $X = (U, S)$, where set U contains $3k$ elements and a collection S of m subsets of U such that each $s_i \in S$ contains exactly 3 elements. We generate an instance $G(R,B,E)$ of the BCS-existence problem from X as follows. The red vertices R are the elements $u_j \in U$; i.e., $R = U$. The blue vertices B are the 3-element sets $s_i \in S$; i.e., $B = S$. For each blue vertex $s_i = \{u_\alpha, u_\beta, u_\gamma\} \in S = B$, we add the 3 edges (s_i, u_α), (s_i, u_β), and (s_i, u_γ) to the set E of edges of G.

We instantiate an additional set of $2k$ blue vertices, $\{b_1, \dots, b_{2k}\}$, and add edges to E to link them into a path $(b_1, b_2, \dots, b_{2k})$. Finally, we add an edge from b_{2k} to each of the blue vertices s_i. Refer to Figure 8.5.

Clearly, the number of vertices and edges in G are polynomial in terms of number of elements and sets in the EC3Set problem instance X, and hence the construction can be done in polynomial time. We now prove the following lemma.

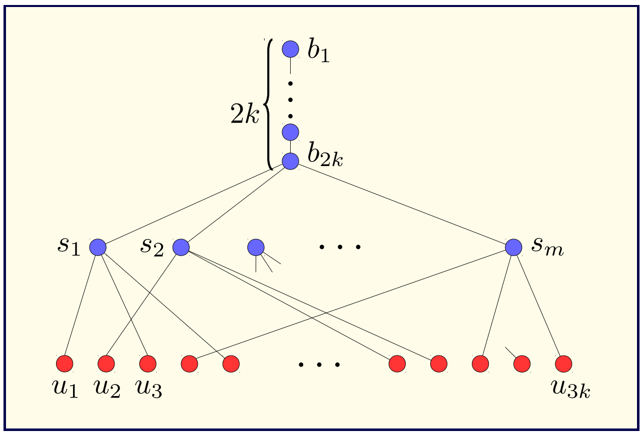


Figure 8.5: Construction of the instance G of the BCS problem containing b_1.

Lemma 8.6. *The instance X of the* EC3Set *problem has a solution iff the instance G of the corresponding* BCS *existence problem has a balanced subgraph T containing the vertex b_1.*

Proof. Assume that the EC3Set problem has a solution, and let S^* be the collection of $k = |S^*|$ sets of S in the solution.

Then, we obtain a balanced subgraph T that contains b_1 as follows: T is the induced subgraph of the $3k$ red vertices U, together with the k blue vertices S^* and the $2k$ blue vertices $b_1, \dots, b_{2k}$. Note that T is balanced and connected and contains b_1.

Conversely, assume there is a balanced connected subgraph T containing b_1. Let t be the number of (blue) vertices of S within T. First, note that $t \leq k$. (Since T is balanced and contains at most $3k$ red vertices, it must contain at most $3k$ blue vertices, $2k$ of which must be $\{b_1, \dots, b_{2k}\}$, in order that T is connected.)

Next, we claim that, in fact, $t \geq k$. To see this, note that each of the t blue vertices of T that corresponds to a set in S is connected by edges to 3 red vertices; thus, T has at most $3t$ red vertices. Now, T has $2k + t$ blue vertices (since it has t vertices other than the path $(b_1, \dots, b_{2k})$), and T is balanced; thus, T has exactly $2k + t$ red vertices, and we conclude that $2k + t \leq 3t$, implying $k \leq t$, as claimed.

Therefore, we need to select exactly k blue vertices corresponding to the sets S, and these vertices connect to all $3k$ of the red vertices. The k sets corresponding to these k blue vertices is a solution for the EC3Set problem. ∎

Clearly, the BCS existence problem is in NP. Hence, we conclude the following theorem.

Theorem 8.6. *It is* NP-Complete *to decide if there exists a balanced connected subgraph that contains a specific vertex.*

8.4 Hardness: Balanced Connected Path Problem

In this section we consider the Balanced Connected Path (BCP) Problem and prove that it is NP-Hard. In this problem instead of finding a balanced connected subgraph, our goal is to find a balanced path with a maximum cardinality of vertices. To prove the BCP problem is NP-Hard we give a polynomial time reduction from the Hamiltonian Path (Ham-Path) problem which is known to be NP-Complete [GJ79a]. In this problem, we are given an undirected graph Q, and the goal is to find a Hamiltonian path in Q i.e., a path which visits every vertex in Q exactly once. In the reduction we generate an instance G of the BCP problem from an instance Q of the Ham-Path problem as follows:

Reduction. We make a new graph Q' from Q. Let us assume that the graph Q contains m vertices. If m is even then $Q' = Q$. If m is odd, then we add a dummy vertex u in Q, connect to every other vertices in Q by edges with u and attach a path of length 2 to u. The resulting graph is our desired Q'. It is easy to observe that, Q has a Hamiltonian path if and only if Q' has a Hamiltonian path.

Now we have a Ham-Path instance Q' with even number of vertices, say n. We arbitrarily choose any $n/2$ vertices in Q' and color them red and color the remaining $n/2$ vertices blue. Let G be the colored graph. This completes the construction. Clearly, this can be done in polynomial time.

Lemma 8.7. *Q' has a Hamiltonian path T if and only if G has a balanced path P with exactly n vertices.*

Proof. Assume that Q' has a Hamiltonian path T. This implies that, T visits every vertex in Q'. Since by the construction there are exactly half of the vertices in G is red and remaining are blue, the same path T is balanced with $n/2$ vertices of each color. On the other hand, assume that there is a balanced path P in G with exactly $n/2$ vertices of each color. Since, G has a total of n vertices, the path P visits every vertex in G. Hence, P is a Hamiltonian path. ■

Therefore, we have the following observation.

Observation 8.1. *The* BCP *problem is* NP-Hard *for general graphs.*

8.5 Algorithmic Results

In this section, we consider several graph families and devise polynomial time algorithms for the BCS problem. Notice that, if the graph is a path or cycle, the optimal solution is just a path. Hence, one can do brute-force search to obtain the maximum balanced path. In case of a complete graph K_n, we output a sub-graph H of K_n induced by V, where $|V| = 2|B|$, $B \subset V$, and B is the set of all blue vertices in K_n (assuming that, the number of blue vertices is at most the number of red vertices in K_n). Clearly, H is the maximum-cardinality balanced connected subgraph in K_n. We consider trees, split graphs, bipartite graphs (properly colored), graphs of diameter 2, and present polynomial algorithms for each of them.

8.5.1 Trees

In this section we give a polynomial-time algorithm for the BCS problem in the case that the input graph is a tree $T = (V,E)$, with vertices $V = V_R \cup V_B$ colored red (V_R) or blue (V_B). Our goal is to find a maximum-cardinality balanced subtree of T.

Fix an embedding of the tree $T = (V,E)$ in the plane [AGH+99]. For a non-leaf vertex $v \in V$ in the tree T, let m be the number of children of v. (If v is a leaf, $m = 0$.) In the embedding of T, assume that the children of each vertex v are drawn in a horizontal row below v, so that we can speak of the left-to-right (i.e., counter-clockwise about v) order of the children, denoted $u_1, u_2, \ldots, u_m$. For a non-leaf vertex v and $1 \leq i \leq m$, let $T_{v,i}$ denote the subtree of T, rooted at v, consisting of the subtrees rooted at the children $u_1, \ldots, u_i$, together with v and the edges linking v to these i children.

For any vertex $v \in V$ and integer $\Delta \in [-|V_B|, |V_R|]$, let $f_v(\Delta)$ be the maximum number of vertices in a subtree of T rooted at v for which the *red excess* is Δ, where the red excess of a subtree is defined to be the number of red vertices in the subtree minus the number of blue vertices in the subtree. (Thus, a color-balanced subtree has red excess $\Delta = 0$, and a subtree with more blue vertices than red vertices has red excess $\Delta < 0$.) If there is no subtree rooted at v with red excess Δ, then we define $f_v(\Delta) = -\infty$. We solve the BCS problem by tabulating $f_v(\Delta)$ for all choices of $v \in V$ and Δ; there are $O(n^2)$ values to tabulate. The size of a largest color-balanced subtree rooted at v is $f_v(0)$, and the size of an overall optimal solution to the BCS problem is then given by $\max_v f_v(0)$.

Note that if v is a leaf of T, then $f_v(1) = 1$ if v is red, $f_v(-1) = 1$ if v is blue, and $f_v(\Delta) = -\infty$ otherwise.

If $v \in V$ is not a leaf of T, let $F_v(i, \Delta)$ be the maximum cardinality of a subtree of $T_{v,i}$ that is rooted at v, and has red excess of Δ. Then, for each (non-leaf) v, $f_v(\Delta) = F_v(m_v, \Delta)$, where m_v is the number of children of vertex v.

We tabulate the values $F_v(i, \Delta)$ from the leaves of T upwards, and for increasing values of i.

The main recursion that allows us to tabulate $F_v(i+1, \Delta)$ is obtained by considering all possible red excess values δ for the subtree rooted at child u_{i+1} that might be included in the subtree of $T_{v,i+1}$ rooted at v: We have the option to attach a tree (rooted at v) of size $F_v(i, \Delta - \delta)$ to a tree (rooted at u_{i+1}) of size $f_{u_{i+1}}(\delta)$, via the edge (v, u_{i+1}), resulting in a subtree of $T_{v,i+1}$, rooted at v, with red excess $\Delta - \delta + \delta = \Delta$. We compare this option with that of not using u_{i+1} at all (yielding a tree of size $F_v(i, \Delta)$). Thus, we have

$$F_v(i+1, \Delta) = \max\left\{F_v(i, \Delta), \max_{\delta}\{F_v(i, \Delta - \delta) + f_{u_{i+1}}(\delta)\}\right\}.$$

At the base of this recursion, we compute, for each non-leaf v,

$$F_v(1, \Delta) = \begin{cases} f_{u_1}(\Delta - 1) + 1 & \text{if } v \text{ is red,} \\ f_{u_1}(\Delta + 1) + 1 & \text{if } v \text{ is blue.} \end{cases}$$

The overall evaluation proceeds by first tabulating the values of $f_v(\Delta)$ for all leaves v and all values of Δ. Then, for non-leaf vertices v, we tabulate values of $F_v(i, \Delta)$, for each choice of Δ, for values of $i = 1, 2, \ldots, m_v$.

Now, there are $O(n)$ choices of the pair v, i (more precisely, there are at most $n-1$ such choices, since these choices correspond to edges of T, linking v to one of its children), and $O(n)$ choices of Δ (since $-|V_B| \le \Delta \le |V_R|$). The optimization in the recursion is over $O(n)$ choices of δ, so the overall tabulation takes time $O(n^3)$, using $O(n^2)$ space.

The actual tree realizing $F_v(i, \Delta)$ or $f_v(\Delta)$ is found by keeping track, during the recursive evaluation of $F_v(i+1, \Delta)$, of whether or not the maximization was achieved using the subtree rooted at u_{i+1} (of size $f_{u_{i+1}}(\delta)$, for an optimal choice of δ), which means using the edge (v, u_{i+1}). This data tells us exactly which children of each vertex v are connected to v in an optimizing tree. We can now conclude the following theorem.

Theorem 8.7. *Let T be a tree whose n vertices are colored either red or blue. Then, in $O(n^3)$ time and $O(n^2)$ space, one can compute a maximum-cardinality balanced subtree of T.*

Remark: The algorithm resulting in Theorem 8.7 generalizes to the case in which vertices of T may take on colors other than just red and blue, and we desire a maximum-cardinality subtree having any specified target ratio of cardinalities among the vertices of the subtree; the running time is polynomial, for a fixed number c of colors, with time bounded by $n^{O(c)}$.

8.5.2 Split graphs

A graph $G = (V,E)$ is defined to be a split graph if there is a partition of V into two sets S and K such that S is an independent set and K is a complete graph. There is no restriction on edges between vertices of S and K. Here we give a polynomial time algorithm for the BCS problem where the input graph $G = (V,E)$ is a split graph. Let V be partitioned into S and K where S and K induce an independent set and a clique respectively in G. Also, let S_B and S_R be the sets of blue and red vertices in S, respectively. Similarly, let K_B and K_R be the sets of blue and red vertices in K, respectively. We argue that there exists a balanced connected subgraph in G, having $\min\{|S_B \cup K_B|, |S_R \cup K_R|\}$ vertices of each color.

Note that if $|S_B \cup K_B| = |S_R \cup K_R|$ then G itself is balanced. Now, w.l.o.g., we can assume that $|S_B \cup K_B| < |S_R \cup K_R|$. We will find a balanced connected subgraph H of G, where the number of vertices in H is exactly $2|S_B \cup K_B|$. To do so, we first modify the graph $G = (V,E)$ to a graph $G' = (V,E')$. Then, from G', we will find the desired balanced subgraph with $|S_B \cup K_B|$ many vertices of each color. Moreover, this process is done in two steps.

Step 1. Construct $G' = (V,E')$ from $G = (V,E)$. For each $u \in S_B$, if u is adjacent to at least a vertex u' in K_R, then remove all incident edges with u except the edge (u,u'). Similarly, for each $v \in S_R$, if v is adjacent to at least a vertex v' in K_B, then remove all incident edges with v except the edge (v,v').

Step 2. Let $k = |S_R \cup K_R| - |S_B \cup K_B|$. Now we we have following cases.

- **Case** 1. $|S_R| \geq k$. We remove k vertices from S_R in G'. Clearly, after this modification, G' is connected, and we get a balanced subgraph having $|S_B \cup K_B|$ vertices of each color.

- **Case** 2. $|S_R| < k$. Then we know, $|K_R| > |K_B \cup S_B|$. Let $S'_B \subseteq S_B$ be the set of vertices in G' such that each vertex of S'_B has exactly one neighbor in K_R. Then, we take a set $X \subset K_R$ with cardinality $|K_B \cup S_B|$ such that X contains all adjacent vertices of S'_B. Now we take the subgraph H of G' induced by $(S_B \cup K_B \cup X)$. Now H is optimal and balanced.

Running time. Step 1 takes $O(|E|)$ time to construct G' from $G = (V,E)$. Now in step 2, both Case 1 and Case 2 take $O(|V|)$ time to delete $|S_R \cup K_R| - |S_B \cup K_B|$ vertices from G'. Hence, the total time taken is $O(|V| + |E|)$.

We can now conclude the following theorem.

Theorem 8.8. *Given a split graph $G = (V,E)$, with r red and b blue ($|V| = r + b$) vertices, then, in $O(|V| + |E|)$ time we can find a balanced connected subgraph of G having* $\min\{b,r\}$ *vertices of each color.*

8.5.3 Bipartite graphs with proper coloring

In this section, we describe a polynomial-time algorithm for the BCS problem where the input graph is a bipartite graph whose vertices are colored red/blue according to proper 2-coloring of vertices in a graph. We show that there is a balanced connected subgraph of G having $\min\{b,r\}$ vertices of each color where G contains r red vertices and b blue vertices. Note that we earlier showed that the BCS problem is NP-Hard in bipartite graphs whose vertices are colored red/blue arbitrarily; here, we insist on the coloring being a proper coloring (the construction in the hardness proof had adjacent pairs of vertices of the same color). We begin with the following lemma.

Lemma 8.8. *Consider a tree T (which is necessarily bipartite) and a proper 2-coloring of its vertices, with r red vertices and b blue vertices. If $r < b$, then T has at least one blue leaf.*

Proof. We prove it by contradiction. Let there is no blue leaf. Now assign any blue vertex, say b_r, as a root. Note that it always exists. Now b_r is at level 0 and b_r has degree at least 2. Otherwise, b_r is a leaf with blue color. We put all the adjacent vertices of b_r in level 1. This level consists of only red vertices. In level 2 we put all the adjacent vertices of level 1. So level 2 consists of only blue vertices. This way we traverse all the vertices in T and let that we stop at k^{th}-level. k cannot be even as all the vertices in even level are blue. So k must be odd. Now for each $0 \leqslant i \leqslant \frac{k-1}{2}$, in the vertices of (level $2i$ $\cup$ level $(2i+1)$), number of blue vertices is at most the number of red vertices. Which leads to the contradiction that $r < b$. Hence there exists at least one leaf with blue color. ■

Now we describe the algorithm. We first find a spanning tree T in G. If $r = b$ then T itself is a maximum balanced subtree (subgraph also) of G. Without loss of generality assume that $r < b$. So by Lemma 8.8, T has at least 1 blue leaf. Now we remove that blue leaf from T. Using similar reason, we repetitively remove $(b-r)$ blue vertices from T. Finally, T becomes balanced subgraph of G, with r vertices of each color.

Running time. Finding a spanning tree in $G=(V,E)$ requires $O(|E|)$ time. To find all the leaves in the tree T requires $O(|V|)$ time (breadth first search). Hence the total time is needed is $O(|V|+|E|)$.

We can now conclude the following theorem.

Theorem 8.9. *Given a bipartite graph $G=(V,E)$ with a proper 2 coloring (r red or b blue vertices), then in $O(|V|+|E|)$ time we can find a balanced connected subgraph in G having* $\min\{b,r\}$ *vertices of each color.*

8.5.4 Graphs of diameter 2

In this section, we give a polynomial time algorithm for the BCS-problem where the input graph has diameter 2. Let $G(V,E)$ be such a graph which contains b blue vertex set B and r red vertex set R. We find a balanced connected subgraph H of G having $\min\{b,r\}$ vertices of each color. Assume that $b<r$. This can be done in two phases. In phase 1, we generate an induced connected subgraph G' of G such that (i) G' contains all the vertices in B, and (ii) the number of vertices in G' is at most $(2b-1)$. In phase 2, we find H from G'.

Phase 1. To generate G', we use the following observation regarding graphs of diameter 2.

Observation 8.2. *Let $G=(V,E)$ be a graph of diameter 2. Then for any pair of non adjacent vertices u and v from G, there always exists a vertex w such that both $(u,w)\in E$ and $(v,w)\in E$.*

We first include B in G'. Now we have the following two cases.

Case 1. The induced subgraph $G[B]$ of B is connected. In this case, G' is $G[B]$.

Case 2. The induced subgraph $G[B]$ of B is not connected. Assume that $G[B]$ has $k(>1)$ components. Let $B_1,B_2,\ldots,B_k$ be k disjoints sets of vertices such that each induced subgraph $G[B_i]$ of B_i in G is connected. Now using Lemma 8.2, any two vertices $v_i\in B_i$ and $v_j\in B_j$ are adjacent to a vertex say $u_\ell\in R$. We repetitively apply Lemma 8.2 to merge all the k subgraphs into a larger graph. We need at most $(k-1)$ red vertices to merge k subgraph. We take this larger graph as the graph G'.

Phase 2. In this phase, we find the balanced connected subgraph H with b vertices of each color. Note that the graph G' generated in phase 1 contains b blue and at most $(b-1)$ red vertices. Assume that G' contains b' red vertices. We add $(b-b')$ red vertices from $G \setminus G'$ to G'. This is possible since G in connected.

Running time. In phase 1, first finding all the blue vertices and it's induced subgraph takes $O(|V|+|E|)$ time. Now to merge all the k components into a single component which is G' needs $O(|E|)$ time. In phase 2, adding $(b-b')$ red vertices to G' takes $O(|E|)$ time as well. Hence, total time requirement is $O(|V|+|E|)$.

We can now come up to the following theorem.

Theorem 8.10. *Given a graph $G=(V,E)$ of diameter 2, where the vertices in G are colored either red or blue. If G has b blue and r red vertices then, in $O(|V|+|E|)$ time we can find a balanced connected subgraph in G having* $\min\{b,r\}$ *vertices of each color.*

8.5.5 Interval graphs

A graph $G=(V,E)$ is said to be an interval graph if each vertex $u \in V$ is associated with an interval $I_u=[l_u,r_u]$ (where l_u and r_u denote the left and right endpoints of I_u, respectively), and for any pair of vertices $u,v \in V$, $(u,v) \in E$ if and only if $I_u \cap I_v \neq \emptyset$. Here we study the BCS problem on connected interval graphs. Given an n vertex interval graph $G=(V,E)$, we order the vertices of G based on the left endpoints $\{l_v : \forall v \in V\}$ of their corresponding intervals. For a pair of vertices $u,v \in V$ with $l_u \leq l_v$, we define a set $S_{u,v}=\{w : w \in V, l_u \leq l_w < r_w \leq r_v\} \cup \{u,v\}$. We also consider the case when $u=v$, and define $S_{u,u}$ in a similar fashion i.e., $S_{u,u}=\{w : w \in V, l_u \leq l_w < r_w \leq r_u\} \cup \{u,v\}$. Let H be a subgraph of G induced by $S_{u,v}$ (or $S_{u,u}$) in G. Let r and b be the number of red and blue vertices in $S_{u,v}$, respectively. Without loss of generality, we may assume that $b \leq r$. The goal is to find a balanced connected subgraph (if exists) of cardinality $2b$ in H, containing u and v. Let B be the set of all blue vertices in $S_{u,v}$ and $T=B \cup \{u,v\}$.

Now we compute a connected subgraph that contains T and may use some extra red intervals. For that, we need to solve the Steiner tree problem on interval graphs. In 1985, White et al. [WFP85] gave a polynomial time solvable algorithm for Steiner tree problem on strongly chordal graphs which is superclass of interval graphs. Here we proposed a different algorithm to solve the same problem on the interval graphs.

Steiner tree on interval graphs. Given a simple connected interval graph $G=(V,E)$ and a set $T \subseteq V$ of terminals, the minimum Steiner tree problem seeks a

Algorithm 6 $Select\_Steiners(G = (V,E),T)$

```
1: $S \leftarrow V \setminus T$;
2: $D \leftarrow \emptyset$;
3: $C \leftarrow$ set of components induced by $T \cup D$;
4: Let $I_{C_1},\dots,I_{C_m}$ be the left to right ordered set based on the right endpoints of the
   components induced by intervals associated with the vertices in $T \cup D$. Where
   $C = \{C_1, C_2, \dots C_m\}$ be a set of $m$ components (for some $m \geq 1$);
5: if $|C| = 1$ then
6:     return $D$;
7: else
8:     $D \leftarrow D \cup I_j$, where $I_j \in S$ and the rightmost interval in $N(I_{C_1})$;
9:     go to step 3;
10: end if
```

smallest tree that spans over T. The vertices in the tree other than T are denoted as D and called as Steiner vertices. First we describe a greedy algorithm called $Select\_Steiners(G = (V,E),T)$ that computes a minimum cardinality Steiner vertices D in G.

Let the vertices in T induces m ($m \in [n]$) components $\{C_1,\dots,C_m\}$ sorted from left to right based on the right endpoints of the components (note that, each component can be interpreted as an interval, union of all the intervals of that component, on the real line). Let I_{C_i} be the rightmost interval of the i-th component C_i. We consider the first component C_1 and the neighborhood set $N(I_{C_1})$ of I_{C_1}. Let I_j be the rightmost interval in $N(I_{C_1})$. By rightmost, we mean that the interval having the rightmost endpoint. We add I_j in D. Now, we recompute the components based on $T \cup D$. Note that, $C_1 \cup I_j$ is now forming a single component. We repeat this procedure until $T \cup D$ becomes a single component. Finally we return D as a solution. The pseudocode of the above mentioned procedure is given in Algorithm 6.

Correctness. It is easy to verify that the graph induced by $T \cup D$ is connected. Now we prove that the optimality of the algorithm $Select\_Steiners(G = (V,E),T)$ by induction. The base case is that we have to connect the first two components C_1 and C_2. We choose the rightmost interval from the neighborhood of I_{C_1}. Note that, we have to connect C_1 with C_2 and therefore it is inevitable that we have to pick an interval from $N(I_{C_1})$. We choose the rightmost interval (say, (I_ℓ)). Now, if this choice already connects C_2, we are done. Otherwise, $C_1 = C_1 \cup I_\ell$. Now, let us assume that we have obtained an optimal solution until step i. At step $i+1$, we have to connect the first two components. By applying the same argument as the base case, we choose the rightmost interval from the first component and proceed. As step 8 takes $O(n)$

Algorithm 7 $BCS\_Interval(H)$

```
1: $T \leftarrow B \cup \{u,v\}$
2: $D = Select\_Steiners(H,T)$
3: $b' \leftarrow$ number of blue vertices in $T$
4: $r' \leftarrow$ number of red vertices in $D \cup \{u,v\}$
5: if $r' > b'$ then
6:     return $\emptyset$ ;
7: else if $r' = b'$ then
8:     return $G[D \cup T]$;
9: else
10:    return $G[D \cup T \cup X]$, where $X \subset V(H)$ is the set of $(b'-r')$ red vertices with
       $X \cap (D \cup T) = \emptyset$;
11: end if
```

time, this algorithm runs in $O(n^2)$ time.

Now, we return to the BCS problem. Recall that, for any pair of vertices $u,v \in V$, our objective is to compute a balanced connected component containing u and v of cardinality $2b$ (if exists), where b and r is the number of blue and red intervals in $S_{u,v}$, respectively, and $b \leq r$. Let H be the subgraph of G induced by $S_{u,v}$. Let R and B denote the set of red and blue intervals in $S_{u,v}$, respectively. We apply Algorithm 6 on the input pair (H,T), where $T = B \cup \{u,v\}$. Let $D = Select\_Steiners(H,T)$. Also let b' and r' be the number of blue vertices in T and number of red vertices in $D \cup \{u,v\}$, respectively. If $r' > b'$, then we conclude that there does not exist a BCS in H with cardinality $2b$. In other case, if $r' \leq b'$, we can add required number of red vertices to get a BCS in H with cardinality $2b$. We describe this process in Algorithm 7. We repeat this procedure for every pair of intervals and report the solution set with the maximum number of intervals.

Correctness. We prove that our algorithm yields an optimum solution. Let G' be such a solution. Let u and v be the intervals with leftmost endpoint and rightmost endpoint of G', respectively. Now we show that $|V(G')| = \min\{2r, 2b\}$, where r and b be the number of red and blue color vertices is $S_{u,v}$, respectively. Let us assume $|V(G')| \neq \min\{2r, 2b\}$. Then there exists at least one blue interval z and one red interval z' that belong to $S_{u,v} \setminus V(G')$. However, we know that $S_{u,v}$ induces an intersection graph of intervals corresponding to the vertices $\{w : w \in V, l_u \leq l_w < r_w \leq r_v\} \cup \{u,v\}$, and G' contain both u and v. So, $N[z] \cap G' \neq \emptyset$, $N[z'] \cap G' \neq \emptyset$. Therefore $V(G') \cup \{z,z'\}$ induces a balanced connected subgraph in G. It contradicts that G' is optimal and hence the proof.

Time Complexity. Here we use the algorithm $Select\_Steiners(G=(V,E),T)$ as a subroutine. Using the range tree data structure, the set $S_{u,v}$ can be obtained in $O(\log n+k)$ time, where where k is the number of points in the query interval. As the output can be linear, hence the total running time is $O(n^5)$. Therefore, we have the following theorem.

Theorem 8.11. *Let G be an interval graph with n vertices that are colored either red or blue. Then the* BCS *problem on G can be solved in $O(n^5)$ time.*

8.5.6 Circular-arc graphs

In this section we study the BCS problem on the circular-arc graphs. A circular-arc graph is an intersection graph of a set of arcs on the circle. It has one vertex for each arc in the set, and an edge between a pair of vertices in the graph if their corresponding arcs intersect.

Let us assume that H be a resulting maximum balanced connected subgraph of G, and let S denote the set of vertices in H. Since H is a connected subgraph of G, H covers the circle either partially or entirely. We propose an algorithm that computes a maximum size balanced connected subgraph H of G in polynomial time. Let V_B and V_R denotes the set of blue and red vertices in G. Without loss of generality we assume that $V_B \le V_R$. For any arc $u \in V$, let $l(u)$ and $r(u)$ denote the two endpoints of u in the clockwise order of the endpoints of the arc. To design the algorithm, we shall concentrate on the following two cases – **Case A** and **Case B**. In Case A, we check all possible instances while the the output set does not cover the circle fully. Then, in Case B, we handle all those instances while the output covers the entire circle. Later, we prove that the optimum solution lies in one of these instances. The objective is to reach the optimum solution by exploiting these instances exhaustively.

Case A. [*H* covers the circle partially] In this case, there must be a clique of K ($|K| \ge 1$) arcs that is not present in the optimal solution. We consider any pair of arcs $u,v \in V$ such that $r(u) \prec l(v)$ in the clockwise order of the endpoints of the arcs ($x \prec y$ means x occurs before than y in the ordering), and consider the vertex set $S_{u,v} = \{w : w \in V, l(v) \le l(w) < r(w) \le r(u)\} \cup \{u,v\}$. Then, we use the Algorithm 7 to compute maximum BCS on $G[S_{u,v}]$. This process is repeated for each such pair of arcs, and report the BCS with maximum number of arcs.

Case B. [*H* covers the circle entirely] In this case, S must contains $2|V_B|$ number of arcs and in fact that is the maximum number of arcs S can opt. In order to compute such a set S, first we add the vertices in V_B to S, then considering the vertices in V_B

as a set T of terminal arcs, we need to find a minimum number of red arcs $D \subseteq V_R$ that connects T. We further assume two sub-cases depending upon $T \cup D$.

B.1. [$T \cup D$ covers the circle partially] This case is similar to Case A without some extra red arcs that would be added afterwards to ensure that S contains $2|V_B|$ arcs. Similar to Case A, we again try all possible subsets obtained by pair of vertices u, v with $r(u) < l(v)$ and $S_{u,v}$ contains all blue vertices and we find optimal Steiner tree by using Algorithm $Select\_Steiners(G = (V, E), T)$. Then, we add $(|V_R| - |D|)$ (where D is the set of Steiner arcs) red arcs from V_R to S.

B.2. [$T \cup D$ covers the circle entirely] First, we obtain a set $\mathscr{C}$ of m (for some $m \in [n]$) components from T. We may see each component $C \in \mathscr{C}$ as an arc and the neighborhood set $N(C)$ as the union of the neighborhoods of the arcs contained in C. Observe that, for any component $C \in \mathscr{C}$, D contains either one arc from $N(C)$ that covers C, or two arcs from $N(C)$ where none of them individually covers C. Let us consider one component $C \in \mathscr{C}$. Let $l(C)$ and $r(C)$ be the left and right endpoints of C, respectively. If $|N(C) \cap D| = 1$, we consider each arc from $N(C)$ separately that covers C. For each such arc $I(C) \in N(C)$, we do the following three step operations –1) include $I(C)$ in D, 2) remove $N(C)$ from the graph, 3) include two blue arcs $(l(I(C)), r(C))$ and $(r(C), r(I(C)))$ in the vertex set of the graph. Now, $T = T \cup \{[l(I(C)), r(C)), (r(C), r(I(C))]\}$. We give this processed graph that is an interval graph, as an input of the Steiner tree (Algorithm $Select\_Steiners(G = (V, E), T)$) and look for a tree with at most $(|D| - 1)$ Steiner red arcs. Else, when $|N(C) \cap D| = 2$, we take the arcs C_ℓ and C_r from $N(C)$ with leftmost and rightmost endpoints, respectively, in D. We do the same three step operations –1) include C_ℓ and C_r in D, 2) remove $N(C)$ from the graph, 3) include two blue arcs $(l(C_\ell), l(C)))$, $(r(C), r(C_r)))$. Now, $T = T \cup \{[l(C_\ell), l(C))), (r(C), r(C_r))]\}$. We give this processed graph that is an interval graph, as an input of the Steiner tree (Algorithm $Select\_Steiners(G = (V, E), T)$) and look for a tree with at most $(|D| - 2)$ Steiner red arcs. This completes the procedure.

Correctness. We prove that our algorithm produces an optimum solution. The proof of correctness follows from the way we have designed the algorithm. The algorithm is divided into two cases. For case A, the primary objective is to construct the instances from a circular-arc graph to some interval graph. Thereafter, we can solve it optimally. Now, Case B is further divided into two sub-cases. Here we know that all blue vertices present in optimum solution. Therefore, our goal is to employ the Steiner tree algorithm with terminal set $T = V_B$. Note, for B.1, we again try all possible subsets obtained by pair of vertices u, v where $r(u) < l(v)$ and $S_{u,v}$ contains all blue vertices. Note, $G[S_{u,v}]$ is an interval graph and V_B is the set of terminals (since we assumed, w.l.o.g, $V_B \leq V_R$). Therefore we directly apply the Steiner tree procedure and obtain the optimum subset for each such pair. Indeed, this process reports the maximum balanced connected subgraph. In the case B.2, we modify the input graph in three step operations. Moreover, we update the expected output size to make it coherent to the modified input instance. This process is done for one arbitrary component of T (of size ≥ 1), which gives an interval graph. Clearly, the choice of this component makes no impact on the size of the balanced connected subgraph. Thereafter, we follow the Steiner tree algorithm on this graph. Moreover, the algorithm exploits all possible cases and reduce the graph into interval graph without affecting the size of the balanced connected subgraph. Thereby, putting everything together, we conclude the proof.

Time complexity. For case A, we try all pairs of arcs that holds certain condition and consider the subset (such subset can be computed in $O(\log n)$ time given the clockwise order of the vertex set and a range tree where the arcs are stored). For each such subset we use the algorithm for interval graph to compute the maximum balanced connected subgraph. This whole process takes $O(n^6 \log n)$ time. The complexity of Case A dominates complexity of Case B and we get the total running time.

We can now state the following theorem.

Theorem 8.12. *Given an n vertex circular-arc graph G whose vertices are colored either red or blue, the* BCS *problem on G can be solved in $O(n^6 \log n)$ time.*

8.5.7 Permutation graphs

In this section, we study the BCS problem on permutation graphs. A graph $G = (V, E)$ with $|V| = n$ is called a permutation graph if there exists a bijection $f : V \to \{1, 2, \ldots, n\}$ and a permutation π of order n such that for every pair of vertices $u, v \in V, (u, v) \in E \Leftrightarrow (f(u) - f(v))(\pi(f(u)) - \pi(f(v))) < 0$. This can be represented as an intersection graph of line segments whose endpoints lie on parallel lines $y = 0$ and $y = 1$. We order the vertices of G based on the endpoints of their corresponding lines on $y = 1$. Let $v_1 < v_2 < \ldots < v_n$ be the ordering, where $v_i < v_j \Leftrightarrow p_i < p_j$ (Assuming $(p_i, 1)$ and $(p_j, 1)$ are the endpoints of the line segments corresponding to vertices v_i and v_j, respectively). For each pair of vertices $v_i, v_j \in V$ (where $1 \leq i < j \leq n$), we define the subgraph $G_{i,j}$ induced by $\{v_i, v_{i+1}, \ldots, v_{j-1}, v_j\}$ in G. Let $R_{i,j}$ and $B_{i,j}$ denote the set of red and blue vertices, respectively in $G_{i,j}$.

Now we propose an algorithm to find a BCS, of $G_{i,j}$, with size $\min\{2|R_{i,j}|, 2|B_{i,j}|\}$ (if exists), containing both v_i and v_j. We do this process for all pair of numbers i and j for which $G_{i,j}$ is connected. Finaly we report the BCS with maximum cardinality. In $G_{i,j}$, to get a BCS (if exists) with size $\min\{2|R_{i,j}|, 2|B_{i,j}|\}$, we apply the algorithm for the Steiner tree problem on permutation graphs with terminal set $B_{i,j}$ (if $|B_{i,j}| \leq |R_{i,j}|$) or $R_{i,j}$ (if $|R_{i,j}| < |B_{i,j}|$). We use the following theorem to solve Steiner tree problem in $G_{i,j}$.

Theorem 8.13. *[AR92] A minimum cardinality Steiner tree in an n vertex permutation graph can be found in $O(m + n\log n)$ time. Here m denotes the number of edges in the graph.*

Now, let $T_{i,j}$ be the solution of Steiner tree problem in $G_{i,j}$ with terminal vertices $B_{i,j}$, where $|B_{i,j}| \leq |R_{i,j}|$. Recall that, we only consider the cases where $G_{i,j}$ is connected. If the number of red vertices in $T_{i,j}$ is strictly greater than $|B_{i,j}|$, then we can say that there does not exist a BCS, of $G_{i,j}$, with size $\min\{2|R_{i,j}|, 2|B_{i,j}|\}$. In other case, if the number of red vertices in $T_{i,j}$ is less than or equal to $|B_{i,j}|$, then we obtain a BCS of size $2|B_{i,j}|$ by simply adding the required number of red vertices. Note, this does not affect the connectivity. By using this method we find a BCS with size $\min\{2|R_{i,j}|, 2|B_{i,j}|\}$ in $G_{i,j}$ for all pair of integers i and j, where$1 \leq i < j \leq n$.

Correctness. We prove that our algorithm produces an optimum solution. Let H be an optimal solution of size t and suppose H consists of the vertices $v_{k_1}, v_{k_2}, \ldots, v_{k_t}$, where $k_1 < k_2 \ldots < k_t$. The graph G_{k_1,k_t} must be connected (otherwise, H becomes disconnected). Consider the Steiner tree problem on the instance $G_{i,j}$, where $i = k_1$ and $j = k_t$. Now, we show that $|V(H)| = \min\{2|R_{i,j}|, 2|B_{i,j}|\}$. Let $|V(H)| \neq \min\{2|R_{i,j}|, 2|B_{i,j}|\}$ then there exists at least one blue vertex u and one red vertex v, such that $u, v \in G_{i,j} \setminus H$. As G_{k_1,k_t} is an intersection graph of line segments corresponding to the vertices $v_{k_1}, v_{k_2}, \ldots, v_{k_t}$, where $k_1 < k_2 < \ldots < k_t$, and H contains both v_{k_1} and v_{k_t}. So, $N[u] \cap H \neq \emptyset$ and $N[v] \cap H \neq \emptyset$. Therefore $V(H) \cup \{u, v\}$ induces a balanced connected subgraph in G. It contradicts the fact that H is a maximum induced balanced connected subgraph in G. Therefore, we conclude the proof.

Time Complexity. We use the algorithm in [CS90] to find minimum cardinality Steiner tree as a subroutine that we call for every pair of integers i, j where $1 \leq i \leq j \leq n$ and $G_{i,j}$ is connected. In case of permutation graphs, we require linear time to obtain an induced subgraph $G_{i,j}$ for any pair of vertices $u, v \in V$. Now, computing a minimum cardinality Steiner tree on permutation graphs takes $O(m+n\log n)$ time, and thus our algorithm solves the BCS problem in $\mathcal{O}(mn^3+n^4\log n)$ time.

We can now conclude the following theorem.

Theorem 8.14. *Given an n vertex, m edge permutation graph G, where the vertices in G are colored either red or blue, the* BCS *problem on G can be solved in $\mathcal{O}(mn^3+n^4\log n)$ time.*

8.6 FPT Algorithm

In this section, we show that the BCS problem is fixed-parameter tractable for general graphs while parameterized by the solution size. Let $G=(V,E)$ be a simple connected graph, and let k be a given parameter. A family $\mathscr{F}$ of functions from V to $\{1,2,\dots,k\}$ is called a perfect hash family for V if the following condition holds. For any subset $U\subseteq V$ with $|U|=k$, there is a function $f\in\mathscr{F}$ which is injective on U, i.e., $f|_U$ is one-to-one. For any graph of n vertices and a positive integer k, it is possible to obtain a perfect hash family of size $2^{O(k)}\log n$ in $2^{O(k)}n\log n$ time; see [AYZ95]. Now, the k-BCS problem can be defined as follows.

***k*-Balanced Connected Subgraph (*k*-BCS) Problem:** Given graph $G=(V,E)$, with vertex set $V=V_R\cup V_B$ partitioned into red vertices (V_R) and blue vertices (V_B), and a positive integer k.

- ➪ Output “Yes" if there is a subset $V'\subseteq V$ such that $G[V']$ is connected and V' conatins $k/2$ red and $k/2$ blue vertices. Else “No".

We employ two methods to solve the k-BCS problem: (i) color coding technique, and (ii) batch procedure. Our approach is motivated by the approach of Fellows et al. [FFHV11], where they have used these techniques to provide a FPT-algorithm for the graph motif problem. Suppose H is a solution to the k-BCS problem and $\mathscr{F}$ is a perfect hash family for V. This ensures us that at least one function of $\mathscr{F}$ assigns vertices of H with k distinct labels. Therefore, if we iterate through all functions of $\mathscr{F}$ and find the subsets of V of size k that are distinctly labeled by our hashing, we are guaranteed to find H. Now, we try to solve the following problem:

Given a hash function $f:V\to\{1,2,\dots,k\}$ from perfect hash family $\mathscr{F}$, decide if there is a subset $U\subseteq V$ with $|U|=k$ such that $G[U]$ is a balanced connected subgraph of G and $f|_U$ is one-to-one.

First, we create a table, denoted by M. For a label $L \subseteq \{1,2,\dots,k\}$ and a color-pair (b,r) of non-negative integers where $b+r=|L|$, we put $M[v\ ;\ L,\ (b,r)]=1$ if and only if there exists a subset $U \subseteq V$ of vertices such that the conditions holds: (i) $v \in U$, (ii) $f|_U = L$, (iii) $G[U]$ is connected, and (iv) U consisting exactly b blue vertices and r red vertices.

Notice that, the total number of entries of this table is $O(2^k kn)$. If we can fill all the entries of the table M, then we can just look at the entries $M[v\ ;\ \{1,2,\dots,k\},\ (\frac{k}{2},\frac{k}{2})]$, $\forall v \in V$, and if any of them is one then we can claim that the k-BCS problem has a solution. Now we use the batch procedure to compute $M[v\ ;\ L,\ (b,r)]$ for each subset $L \subseteq \{1,2,\dots k\}$, and for each pair (b,r) of non-negative integers such that $(b+r)=|L|$. Now, we explain the batch procedure. Without loss of generality, we assume that $L=\{1,2,\dots,t\}$, $f(v)=t$, and the color of v is red.

Batch Procedure $(\boldsymbol{v},\boldsymbol{L},(\boldsymbol{b},\boldsymbol{r}))$:

- **Initialize:** Construct the set $\mathcal{S}$ of pairs $(L',(b',r'))$, where $b'+r'=|L'|$ such that $L' \subseteq \{1,2,\dots,t-1\}$, $b' \le b$, $r' \le r-1$ and $M[u\ ;\ L',\ (b',r')]=1$ for some neighbour u of v.
- **Update:** If there exists two pairs $\{(L_1,(b_1,r_1)),(L_2,(b_2,r_2))\} \in \mathcal{S}$ such that $L_1 \cap L_2 = \emptyset$ and $(b_1,r_1)+(b_2,r_2) \le (b,r-1)$, then add $(L_1 \cup L_2,(b_1+b_2,r_1+r_2))$ into $\mathcal{S}$. Repeat this step until unable to add any more.
- **Decide:** Set $M[v\ ;\ L,\ (b,r)]=1$ if $(\{1,2,\dots,t-1\},(b,r-1)) \in \mathcal{S}$, otherwise 0.

Correctness of the Batch Procedure.

Lemma 8.9. *The batch procedure correctly computes* $M[v\ ;\ L,\ (b,r)]$ *for all* v, $L \subseteq \{1,2,\dots,k\}$ *with* $b+r=|L|$.

Proof. Without loss of generality we assume that $L=\{1,2,\dots,t\}$, $f(v)=t$ and color of v is red. We have to show that $M[v\ ;\ L,\ (b,r)]=1 \Leftrightarrow$ there exists a connected subgraph, containing v, and having exactly b blue vertices and r red vertices. Firstly, we assume that $M[v\ ;\ L,\ (b,r)]=1$. So, $(\{1,2,\dots,t-1\},(b,r-1)) \in \mathcal{S}$. So, there must exist some neighbours $\{v'_1,v'_2,\dots,v'_l\}$ of v such that $M[v'_1;\ L_1,\ (b_1,r_1)] = M[v'_2;\ L_2,\ (b_2,r_2)] = \cdots = M[v'_l;\ L_l,\ (b_l,r_l)]=1$ with $\bigcup_{i=1}^{l} L_i = L\setminus\{t\}, \sum_{i=1}^{l} b_i = b, \sum_{i=1}^{l} r_i = r-1$ where $L_1,L_2,\dots,L_l$ are pairwise disjoint. Thus, there exists a connected subgraph containing $\{v,v'_1,\dots,v'_l\}$ having exactly b blue vertices and r red vertices. Other direction of the proof follows from the same idea. ■

Lemma 8.10. *Given a hash function $f : V \rightarrow \{1,2,\ldots,k\}$, the batch procedure determines all of the entries in the table M in $O(2^{4k}k^3n^2)$ time.*

Proof. The initialization depends on the number of the search process in the entries correspond to the neighbour of v. Now, this number is bounded by the size of M. The first step takes $O(2^k kn)$ time. Now the size of $\mathscr{S}$ can be at most $2^k k$. Each update takes $O(2^{2k}k^2)$ time. So step 2 takes $O(2^{3k}k^3)$ time. Now the value of $M[v\ ;\ L,\ (b,r)]$ can be decided in $O(2^k k)$ time. As the number of entries in M is $O(2^k kn)$, so the total running time is $O(2^{4k}k^3n^2)$. ■

The algorithm for the k-BCS problem is following:

1. Construct a perfect hash family $\mathscr{F}$ of $2^{O(k)}\log n$ functions in $2^{O(k)}n\log n$ time.

2. For each function, $f \in \mathscr{F}$ build the table M using batch procedure. For each function $f \in \mathscr{F}$ it takes $O(2^{4k}k^3n^2)$ time.

3. As each $f \in \mathscr{F}$ is perfect, by an exhaustive search through all function in $\mathscr{F}$ our algorithm correctly decide whether there exists a balanced connected subgraph of k vertices. We output yes, if and only if there is a vertex v and $f \in \mathscr{F}$ for which the corresponding table M contains the entry one in $M[v\ ;\ \{1,2,\ldots,k\},\ \left(\frac{k}{2},\frac{k}{2}\right)]$.

Hence the following theorem.

Theorem 8.15. *The k-BCS problem can be solved in time $2^{O(k)}n^2\log n$ time.*

Summary

In this chapter, we have studied the tractability landscape of the maximum balanced connected subgraph problem on bipartite graphs, chordal graphs, planar graphs, trees, split graphs, etc along with some well known geometric intersection graphs like interval, circular-arc, permutation, unit disk, outerstring graphs, etc. We have obtained several algorithmic and NP-Hardness results. We show that BCS problem remains NP-Hard on restricted geometric intersection graph classes such as unit disk graphs, outerstring graphs, complete grid graphs, and unit square graphs. Moreover, we presented a FPT algorithm for the BCS problem on general graphs.

CHAPTER 9

CONCLUSION

Contents

In this thesis, we have studied some well-structured subgraph finding problems on several graph classes mainly on geometric intersection graphs. In this chapter, we list some open problems that occurred in this thesis. Some of the considered problems are shown to be NP-Hard, and several approximation results are presented. We also obtain exact polynomial-time solvable algorithms for some of the studied problems.

9.1 Future directions

Here we list some open problems, occurred in this thesis, that can be considered as future works.

Chapter 4

In this chapter, we studied the construction of the Manhattan network for a given point set. Here, we are able to construct a planar Manhattan network G for a given convex point set S of size n, where both the running time of the construction and size of G is $O(n)$. It remains an open question if it is possible to construct a planar Manhattan network for general point sets using a subquadratic number of Steiner points.

Chapter 5

In this chapter, we studied the problem of computing a maximum-size bipartite subgraph on geometric intersection graphs. We mainly obtain several approximation algorithms for the problem on unit squares, unit-height rectangles, and variants of unit disks. Here we propose several unsolved problems.

- Does MBS admit a PTAS on unit-height rectangles, or is it APX-hard?
- finding complexity status of maximum bipartite subgraph problem and balance connected subgraph problem in the graph classes that we do not consider. For example, outerstring graphs, outerplanar graphs, graphs with bounded treewidth, etc.
- Is there a polynomial-time algorithm for MBS on a set of n unit disks intersecting a common horizontal line or more generally, unit disks contained in a slab of fixed height?
- Find geometric intersection graph classes for which MBS is NP-Hard but MIS is solvable in polynomial time.

Chapter 6

In this chapter, we study variations of the Set Cover, Independent Set, and Dominating Set problems on a planar subdivision induced by a finite set of axis-parallel line segments. Here we leave with some interesting open problems.

- Producing efficient constant-factor approximation algorithms for all the three problems: the STABBING-SUBDIVISION problem, INDEPENDENT-SUBDIVISION problem, and DOMINATING-SUBDIVISION problem?
- Finding FPT algorithms in all three problems individually parameterized by the size of an optimal solution of the corresponding problem.

Chapter 7

In this chapter, we obtain exact linear-time algorithms for computing a maximum cardinality uniquely restricted matching in proper interval graphs and bipartite permutation graphs respectively. It would be interesting to explore this problem on the graph classes that are still not considered,e.g., unit disk graphs, outerstring graphs, graphs with bounded treewidth.

Chapter 8

In this chapter, we studied the BCS problem on bipartite graphs, chordal graphs, planar graphs, trees, split graphs, etc along with some well-known geometric intersection graphs like interval, circular-arc, permutation, unit-disk, outer-string graphs, etc. We have obtained several algorithmic and NP-Hardness results. In future, the BCS problem can be explored in the approximation algorithm paradigm. Specifically, let G be a vertex-colored graph and let OPT be the total number of vertices in an optimal solution for the BCS problem. The objective is to design a polynomial-time approximation algorithm that yields a solution H such that contains at least $\alpha \times \mathsf{OPT}$ vertices and the difference of red and blue vertices in H is at most β.

BIBLIOGRAPHY

[AAFM$^+$15] Oswin Aichholzer, Nieves Atienza, Ruy Fabila-Monroy, Pablo Perez-Lantero, Jose M Dıaz-Báñez, David Flores-Peñaloza, Birgit Vogtenhuber, and Jorge Urrutia. Balanced Islands in Two Colored Point Sets in the Plane. *arXiv preprint arXiv:1510.01819*, 2015.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ABB$^+$16] Mahdi Amani, Ahmad Biniaz, Prosenjit Bose, Jean-Lou De Carufel, Anil Maheshwari, and Michiel Smid. A plane 1.88-spanner for points in convex position. *Journal of Computational Geometry*, 7(1):520–539, 2016.

[ABKK$^+$10] Louigi Addario-Berry, W Sean Kennedy, Andrew D King, Zhentao Li, and Bruce Reed. Finding a maximum-weight induced k-partite subgraph of an i-triangulated graph. *Discrete Applied Mathematics*, 158(7):765–770, 2010.

[ACC$^+$96] Srinivasa Arikati, Danny Z Chen, L Paul Chew, Gautam Das, Michiel Smid, and Christos D Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *European Symposium on Algorithms (ESA)*, pages 514–528. Springer, 1996.

[AdBF$^+$11] Mohammad Ali Abam, Mark de Berg, Mohammad Farshi, Joachim Gudmundsson, and Michiel H. M. Smid. Geometric Spanners for Weighted Point Sets. *Algorithmica*, 61(1):207–225, 2011.

[AGH$^+$99] Manuel Abellanas, Jesus Garcia-Lopez, Gregorio Hernández-Peñalver, Marc Noy, and Pedro A. Ramos. Bipartite Embeddings of Trees in the Plane. *Discrete Applied Mathematics*, 93(2-3):141–148, 1999.

[AI77] K. Aoshima and Masao Iri. Comments on F. Hadlock's Paper: "Finding a Maximum Cut of a Planar Graph in Polynomial Time". *SIAM Journal on Computing*, 6(1):86–87, 1977.

[AR92] K Arvind and C Pandu Regan. Connected domination and Steiner set on weighted permutation graphs. *Information processing letters*, 41(4):215–220, 1992.

[Aro98] Sanjeev Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998.

[AvKS98] Pankaj K. Agarwal, Marc J. van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry*, 11(3-4):209–218, 1998.

[AW13] Anna Adamaszek and Andreas Wiese. Approximation Schemes for Maximum Weight Independent Set of Rectangles. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 400–409, 2013.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995.

[Bak94] Brenda S. Baker. Approximation Algorithms for NP-complete Problems on Planar Graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.

[BB16] Mourad Baïou and Francisco Barahona. Maximum Weighted Induced Bipartite Subgraphs and Acyclic Subgraphs of Planar Cubic Graphs. *SIAM Journal on Discrete Mathematics*, 30(2):1290–1301, 2016.

[BBBN20] Sayan Bandyapadhyay, Aritra Banik, Sujoy Bhore, and Martin Nöllenburg. Geometric Planar Networks on Bichromatic Points. In *Annual International Conference on Algorithms and Discrete Applied Mathematics (CALDAM)*, pages 79–91. Springer, 2020.

[BBF95] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. Constant Ratio Approximations of the Weighted Feedback Vertex Set Problem for Undirected Graphs. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 142–151, 1995.

[BBF99] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-Approximation Algorithm for the Undirected Feedback Vertex Set Problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[BCC$^+$12] Prosenjit Bose, Jean Cardinal, Sébastien Collette, Ferran Hurtado, Matias Korman, Stefan Langerman, and Perouz Taslakian. Coloring and guarding arrangements. *arXiv preprint arXiv:1205.5162*, 2012.

[BCJ$^+$19] Sujoy Bhore, Sourav Chakraborty, Satyabrata Jana, Joseph SB Mitchell, Supantha Pandit, and Sasanka Roy. The Balanced Connected Subgraph Problem. In *Annual International Conference on Algorithms and Discrete Applied Mathematics (CALDAM)*, pages 201–215. Springer, 2019.

[BCJ$^+$21] Sujoy Bhore, Sourav Chakraborty, Satyabrata Jana, Joseph SB Mitchell, Supantha Pandit, and Sasanka Roy. The Balanced Connected Subgraph Problem. *Discrete Applied Mathematics*, 2021.

[BCK$^+$19] Prosenjit Bose, Paz Carmi, J. Mark Keil, Anil Maheshwari, Saeed Mehrabi, Debajyoti Mondal, and Michiel H. M. Smid. Computing Maximum Independent Set on Outerstring Graphs and Their Relatives. In *International Symposium on Algorithms and Data Structures (WADS)*, pages 211–224, 2019.

[BHK$^+$15] Sergey Bereg, Ferran Hurtado, Mikio Kano, Matias Korman, Dolores Lara, Carlos Seara, Rodrigo I Silveira, Jorge Urrutia, and Kevin Verbeek. Balanced partitions of 3-colored geometric sets in the plane. *Discrete Applied Mathematics*, 181:21–32, 2015.

[BHK$^+$20] Sujoy Bhore, Jan-Henrik Haunert, Fabian Klute, Guangping Li, and Martin Nöllenburg. Balanced Independent and Dominating Sets on Colored Interval Graphs. *CoRR*, abs/2003.05289, 2020.

[BJPR19] Sujoy Bhore, Satyabrata Jana, Supantha Pandit, and Sasanka Roy. Balanced Connected Subgraph Problem in Geometric Intersection Graphs. In *International Conference on Combinatorial Optimization and Applications (COCOA)*, pages 56–68, 2019.

[BK85] Andreas Brandstädt and Dieter Kratsch. On the restriction of some NP-complete graph problems to permutation graphs. In *Fundamentals of Computation Theory (FCT)*, pages 53–62, 1985.

[BK87] Andreas Brandstädt and Dieter Kratsch. On Domination Problems for Permutation and Other Graphs. *Theoretical Computer Science*, 54:181–198, 1987.

[BMMP17] Niranjan Balachandran, Rogers Mathew, Tapas Kumar Mishra, and Sudebkumar Prasant Pal. System of unbiased representatives for a collection of bicolorings. *arXiv preprint arXiv:1704.07716*, 2017.

[BMS14] Ahmad Biniaz, Anil Maheshwari, and Michiel HM Smid. Bottleneck Bichromatic Plane Matching of Points. In *Canadian Conference on Computational Geometry (CCCG)*, 2014.

[Bra92] Andreas Brandstädt. On Improved Time Bounds for Permutation Graph Problems. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 1–10, 1992.

[BRS09] Sebastian Böcker, Florian Rasche, and Tamara Steijger. Annotating fragmentation patterns. In Steven L. Salzberg and Tandy Warnow, editors, *Algorithms in Bioinformatics*, pages 13–24, 2009.

[BRS19] Julien Baste, Dieter Rautenbach, and Ignasi Sau. Approximating maximum uniquely restricted matchings in bipartite graphs. *Discrete Applied Mathematics*, 267:30–40, 2019.

[BS$^+$99] Andreas Brandstadt, Jeremy P Spinrad, et al. *Graph classes: a survey*, volume 3. Siam, 1999.

[BS17] Édouard Bonnet and Florian Sikora. The Graph Motif problem parameterized by the structure of the input graph. *Discrete Applied Mathematics*, 231:78–94, 2017.

[BSTV13] Binh-Minh Bui-Xuan, Ondrej Suchý, Jan Arne Telle, and Martin Vatshelle. Feedback vertex set on graphs of low clique-width. *European Journal of Combinatorics*, 34(3):666–679, 2013.

[BvBF$^+$11] N. Betzler, R. van Bevern, M. R. Fellows, C. Komusiewicz, and R. Niedermeier. Parameterized Algorithmics for Finding Connected Motifs in Biological Networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(5):1296–1308, 2011.

[BvKL$^+$09] Magdalene G. Borgelt, Marc J. van Kreveld, Maarten Löffler, Jun Luo, Damian Merrick, Rodrigo I. Silveira, and Mostafa Vahedi. Planar bichromatic minimum spanning trees. *Journal of Discrete Algorithms*, 7(4):469–478, 2009.

[BWWS06] Marc Benkert, Alexander Wolff, Florian Widmann, and Takeshi Shirabe. The minimum Manhattan network problem: Approximations and exact solutions. *Computational Geometry*, 35(3):188–208, 2006.

[Cao18] Yixin Cao. A Naive Algorithm for Feedback Vertex Set. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 1:1–1:9, 2018.

[CC09] Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 892–901, 2009.

[CCJ90] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

[CCJ91] Brent N Clark, Charles J Colbourn, and David S Johnson. Unit disk graphs. In *Annals of Discrete Mathematics*, volume 48, pages 165–177. Elsevier, 1991.

[CE16] Julia Chuzhoy and Alina Ene. On Approximating Maximum Independent Set of Rectangles. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 820–829, 2016.

[CEF12] Yijia Chen, Kord Eickmeyer, and Jörg Flum. The Exponential Time Hypothesis and the Parameterized Clique Problem. In *International Symposium on Parameterized and Exact Computation (IPEC)*, pages 13–24, 2012.

[CF89] Derek G. Corneil and Jean Fonlupt. The complexity of generalized clique covering. *Discrete Applied Mathematics*, 22(2):109–118, 1989.

[CFHW18] Steven Chaplick, Stefan Felsner, Udo Hoffmann, and Veit Wiechert. Grid intersection graphs and order dimension. *Order*, 35(2):363–391, 2018.

[CFK$^+$15] Marek Cygan, Fedor V Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.

[CFL$^+$08] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188–1198, 2008.

[CGK$^+$17] Merce Claverol, Delia Garijo, Matias Korman, Carlos Seara, and Rodrigo I Silveira. Stabbing segments with rectilinear objects. *Applied Mathematics and Computation*, 309:359–373, 2017.

[CGP98] Zhi-Zhong Chen, Enory Grigni, and Christos H. Papadimitriou. Planar Map Graphs. In *ACM Symposium on Theory of Computing (STOC)*, pages 514–523, 1998.

[CGP02] Zhi-Zhong Chen, Michelangelo Grigni, and Christos H. Papadimitriou. Map Graphs. *Journal of the ACM (JACM)*, 49(2):127–138, 2002.

[CGS11] Francis YL Chin, Zeyu Guo, and He Sun. Minimum Manhattan network is NP-complete. *Discrete & Computational Geometry*, 45(4):701–722, 2011.

[CH12] Timothy M. Chan and Sariel Har-Peled. Approximation Algorithms for Maximum Independent Set of Pseudo-Disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

[Che01] Zhi-Zhong Chen. Approximation Algorithms for Independent Sets in Map Graphs. *Journal of Algorithms*, 41(1):20–40, 2001.

[CHO$^+$14] Steven Chaplick, Pavol Hell, Yota Otachi, Toshiki Saitoh, and Ryuhei Uehara. Intersection dimension of bipartite graphs. In *International Conference on Theory and Applications of Models of Computation*, pages 323–340. Springer, 2014.

[CKP$^+$18] Merce Claverol, Elena Khramtcova, Evanthia Papadopoulou, Maria Saumell, and Carlos Seara. Stabbing circles for sets of segments in the plane. *Algorithmica*, 80(3):849–884, 2018.

[CKX11] Shiliang Cui, Iyad A. Kanj, and Ge Xia. On the stretch factor of Delaunay triangulations of points in convex position. *Computational Geometry*, 44(2):104–109, February 2011.

[CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[CM07] Denis Cornaz and Ali Ridha Mahjoub. The Maximum Induced Bipartite Subgraph Problem with Edge Weights. *SIAM Journal on Discrete Mathematics*, 21(3):662–675, 2007.

[CNR89] Hyeong-Ah Choi, Kazuo Nakajima, and Chong S. Rim. Graph Bipartization and via Minimization. *SIAM Journal on Discrete Mathematics*, 2(1):38–47, 1989.

[CNV08] Victor Chepoi, Karim Nouioua, and Yann Vaxes. A rounding algorithm for approximating minimum Manhattan networks. *Theoretical Computer Science*, 390(1):56–69, 2008.

[Cor04] D. G. Corneil. A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs. *Discrete Applied Mathematics*, 138(3):371–379, 2004.

[CP20] Juhi Chaudhary and BS Panda. On the Complexity of Minimum Maximal Uniquely Restricted Matching. In *International Conference on Combinatorial Optimization and Applications*, pages 364–376. Springer, 2020.

[CRCS$^+$94] Jurek Czyzowicz, Eduardo Rivera-Campo, Nicola Santoro, Jorge Urrutia, and Joseph Zaks. Guarding rectangular art galleries. *Discrete Applied Mathematics*, 50(2):149–157, 1994.

[CS90] Charles J Colbourn and Lorna K Stewart. Permutation graphs: connected domination and Steiner trees. *Discrete Mathematics*, 86(1-3):179–189, 1990.

[DB91] Mark De Berg. On rectilinear link distance. *Computational Geometry*, 1(1):13–34, 1991.

[DFJ$^+$20] Konrad K Dabrowski, Carl Feghali, Matthew Johnson, Giacomo Paesani, Daniël Paulusma, and Paweł Rzążewski. On cycle transversals and their connected variants in the absence of a small linear forest. *Algorithmica*, 82(10):2841–2866, 2020.

[DFK$^+$18] Aparna Das, Krzysztof Fleszar, Stephen G. Kobourov, Joachim Spoerhase, Sankar Veeramoni, and Alexander Wolff. Approximating the Generalized Minimum Manhattan Network Problem. *Algorithmica*, 80(4):1170–1190, 2018.

[DGKP19] Benoît Darties, Rodolphe Giroudeau, Jean-Claude König, and Valentin Pollet. The Balanced Connected Subgraph Problem: Complexity Results in Bounded-Degree and Bounded-Diameter Graphs. In *International*

Conference on Combinatorial Optimization and Applications (COCOA), pages 449–460. Springer, 2019.

[DK01] Adrian Dumitrescu and Rick Kaye. Matching colored points in the plane: some new results. *Computational Geometry*, 19(1):69–85, 2001.

[DM41] B. Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63(3):600–610, 1941.

[DP02] Adrian Dumitrescu and János Pach. Partitioning colored point sets into monochromatic parts. *International Journal of Computational Geometry & Applications*, 12(05):401–412, 2002.

[DR90] Amitava Datta and GDS Ramkumar. On some largest empty orthoconvex polygons in a point set. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 472 of *LNCS*, pages 270–285. Springer, 1990.

[DS08] Michael Dom and Somnath Sikdar. The parameterized complexity of the rectangle stabbing problem and its variants. In *International Workshop on Frontiers in Algorithmics*, pages 288–299. Springer, 2008.

[EK14] Mohammed El-Kebir and Gunnar W. Klau. Solving the Maximum-Weight Connected Subgraph Problem to Optimality. *CoRR*, abs/1409.5308, 2014.

[EMP$^+$82] Herbert Edelsbrunner, Hermann A. Maurer, Franco P. Preparata, Arnold L. Rosenberg, Emo Welzl, and Derick Wood. Stabbing line segments. *BIT Numerical Mathematics*, 22(3):274–281, 1982.

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[Fel14] Stefan Felsner. The order dimension of planar maps revisited. *SIAM Journal on Discrete Mathematics*, 28(3):1093–1101, 2014.

[FFHV11] Michael R. Fellows, Guillaume Fertin, Danny Hermelin, and Stéphane Vialette. Upper and lower bounds for finding connected motifs in vertex-colored graphs. *Journal of Computer and System Sciences*, 77(4):799–811, 2011.

[FGPR08] Fedor V. Fomin, Serge Gaspers, Artem V. Pyatkin, and Igor Razgon. On the Minimum Feedback Vertex Set Problem: Exact and Enumeration Algorithms. *Algorithmica*, 52(2):293–307, 2008.

[FJJ18] Mathew C. Francis, Dalu Jacob, and Satyabrata Jana. Uniquely Restricted Matchings in Interval Graphs. *SIAM Journal on Discrete Mathematics*, 32(1):148–172, 2018.

[FLP$^+$19] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Decomposition of Map Graphs with Applications. In *International Colloquium on Automata, Languages, and Programming, ICALP 2019*, pages 60:1–60:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[FPT81] Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12:133 – 137, 1981.

[Fra80] András Frank. On chain and antichain families of a partially ordered set. *Journal of Combinatorial Theory, Series B*, 29(2):176–184, 1980.

[Fre91] Greg N. Frederickson. Planar Graph Decomposition and All Pairs Shortest Paths. *Journal of the ACM (JACM)*, 38(1):162–204, 1991.

[GHK$^+$15] Alfredo García, Ferran Hurtado, Matias Korman, Inês Matos, Maria Saumell, Rodrigo I Silveira, Javier Tejel, and Csaba D Tóth. Geometric biplane graphs II: Graph augmentation. *Graphs and Combinatorics*, 31(2):427–452, 2015.

[GHL01] M. C. Golumbic, T. Hirst, and M. Lewenstein. Uniquely restricted matchings. *Algorithmica*, 31:139–154, 2001.

[Gib85] Alan Gibbons. *Algorithmic graph theory*. Cambridge university press, 1985.

[GIK02] Daya Ram Gaur, Toshihide Ibaraki, and Ramesh Krishnamurti. Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem. *Journal of Algorithms*, 43(1):138–152, 2002.

[GJ77] Michael R Garey and David S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[GJ79a] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GJ79b] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[GKKS07] Joachim Gudmundsson, Oliver Klein, Christian Knauer, and Michiel Smid. Small Manhattan networks and algorithmic applications for the Earth mover's distance. In *European Workshop on Computational Geometry (EuroCG)*, pages 174–177, 2007.

[GKPP19] Andrzej Grzesik, Tereza Klimošová, Marcin Pilipczuk, and Michal Pilipczuk. Polynomial-time algorithm for maximum weight independent set on p_6-free graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1257–1271. SIAM, 2019.

[GLN01] Joachim Gudmundsson, Christos Levcopoulos, and Giri Narasimhan. Approximating a Minimum Manhattan Network. *Nordic Journal of Computing*, 8(2):219–232, 2001.

[GSZ11] Zeyu Guo, He Sun, and Hong Zhu. Greedy Construction of 2-Approximate Minimum Manhattan Networks. *International Journal of Computational Geometry*, 21(3):331–350, 2011.

[Had75] F. Hadlock. Finding a Maximum Cut of a Planar Graph in Polynomial Time. *SIAM Journal on Computing*, 4(3):221–225, 1975.

[HH04] P. Hell and J. Huang. Certifying LexBFS Recognition Algorithms for Proper Interval Graphs and Proper Interval Bigraphs. *SIAM Journal on Discrete Mathematics*, 18(3):554–570, 2004.

[HKM11] Hirotoshi Honma, Yutaro Kitamura, and Shigeru Masuyama. An Algorithm for Minimum Feedback Vertex Set Problem on a Trapezoid Graph. *IEICE Transactions*, 94-A(6):1381–1385, 2011.

[HM85a] Dorit S. Hochbaum and Wolfgang Maass. Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

[HM85b] Dorit S. Hochbaum and Wolfgang Maass. Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

[HNS16] Hirotoshi Honma, Yoko Nakajima, and Atsushi Sasaki. An algorithm for the feedback vertex set problem on a normal Helly circular-arc graph. *Journal of Computer and Communications*, 4(08):23, 2016.

[IA83] Hiroshi Imai and Takao Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4(4):310–323, 1983.

[IMTP18] Giuseppe F. Italiano, Yannis Manoussakis, Nguyen Kim Thang, and Hong Phong Pham. Maximum Colorful Cliques in Vertex-Colored Graphs. In *International Computing and Combinatorics Conference (COCOON)*, pages 480–491. Springer, 2018.

[JMMR20] Satyabrata Jana, Anil Maheshwari, Saeed Mehrabi, and Sasanka Roy. Maximum Bipartite Subgraph of Geometric Intersection Graphs. In *International Conference on Algorithms and Computation (WALCOM)*, pages 158–169, 2020.

[Joh85] David S Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 6(1):145 – 159, 1985.

[JP19] Satyabrata Jana and Supantha Pandit. Covering and Packing of Rectilinear Subdivision. In *International Conference on Algorithms and Computation (WALCOM)*, pages 381–393, 2019.

[JP20] Satyabrata Jana and Supantha Pandit. Covering and packing of rectilinear subdivision. *Theoretical Computer Science*, 840:166–176, 2020.

[Kar72] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[KG89] J. Mark Keil and Carl A. Gutwin. The Delauney Triangulation Closely Approximates the Complete Euclidean Graph. In *Workshop on Algorithms and Data Structures (WADS)*, pages 47–56, 1989.

[KIA02] Ryo Kato, Keiko Imai, and Takao Asano. An improved algorithm for the minimum Manhattan network problem. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 344–356. Springer, 2002.

[KK03] Atsushi Kaneko and Mikio Kano. Discrete geometry on red and blue points in the plane—a survey—. In *Discrete and computational geometry*, pages 551–570. Springer, 2003.

[KKM$^+$19] Yasuaki Kobayashi, Kensuke Kojima, Norihide Matsubara, Taiga Sone, and Akihiro Yamamoto. Algorithms and Hardness Results for the Maximum Balanced Connected Subgraph Problem. In *International Confer-*

ence on Combinatorial Optimization and Applications (COCOA), pages 303–315. Springer, 2019.

[KKW17] Atsushi Kaneko, Mikio Kano, and Mamoru Watanabe. Balancing Colored Points on a Line by Exchanging Intervals. *Journal of Information Processing*, 25:551–553, 2017.

[KM16] Ekkehard Köhler and Lalla Mouatadid. A linear time algorithm to compute a maximum weighted independent set on cocomparability graphs. *Information Processing Letters*, 116(6):391–395, 2016.

[KMT08] Dieter Kratsch, Haiko Müller, and Ioan Todinca. Feedback vertex set on AT-free graphs. *Discrete Applied Mathematics*, 156(10):1936–1947, 2008.

[KN90] Jan Kratochvíl and Jaroslav Nešetřil. Independent set and clique problems in intersection-defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 31(1):85–93, 1990.

[KN13] R. Krithika and N. S. Narayanaswamy. Parameterized Algorithms for (r,ℓ)-Partization. *Journal of Graph Algorithms and Applications*, 17(2):129–146, 2013.

[KP14] Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556–560, 2014.

[KPR18] Matias Korman, Sheung-Hung Poon, and Marcel Roeloffzen. Line segment covering of cells in arrangements. *Information Processing Letters*, 129:25–30, 2018.

[KR92] Donald E Knuth and Arvind Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992.

[KS11] Christian Knauer and Andreas Spillner. A fixed-parameter algorithm for the minimum Manhattan network problem. *Journal of Computational Geometry*, 2(1), 2011.

[KYK80] Tohru Kikuno, Noriyoshi Yoshida, and Yoshiaki Kakuda. The NP-Completeness of the dominating set problem in cubic planer graphs. *IEICI Transactions (1976-1990)*, 63(6):443–444, 1980.

[LAP03] Fumei Lam, Marina Alexandersson, and Lior Pachter. Picking alignments from (Steiner) trees. *Journal of Computational Biology*, 10(3-4):509–520, 2003.

[LC97] Y. Daniel Liang and Maw-Shang Chang. Minimum Feedback Vertex Sets in Cocomparability Graphs and Convex Bipartite Graphs. *Acta Informatica*, 34(5):337–346, 1997.

[LFS06] V. Lacroix, C. G. Fernandes, and M. Sagot. Motif Search in Graphs: Application to Metabolic Networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):360–368, 2006.

[Lia94] Y. Daniel Liang. On the Feedback Vertex Set Problem in Permutation Graphs. *Information Processing Letters*, 52(3):123–129, 1994.

[Lic82] David Lichtenstein. Planar Formulae and Their Uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[LM08] Vadim V Lozin and Martin Milanič. A polynomial algorithm to find an independent set of maximum weight in a fork-free graph. *Journal of Discrete Algorithms*, 6(4):595–604, 2008.

[LMS15] Abhiruk Lahiri, Joydeep Mukherjee, and C. R. Subramanian. Maximum Independent Set on B_1-VPG Graphs. In *International Conference on Combinatorial Optimization and Applications (COCOA)*, pages 633–646. 2015.

[LR00] Bing Lu and Lu Ruan. Polynomial time approximation scheme for the rectilinear Steiner arborescence problem. *Journal of Combinatorial Optimization*, 4(3):357–363, 2000.

[LSS09] Daniel Lokshtanov, Saket Saurabh, and Somnath Sikdar. Simpler Parameterized Algorithm for OCT. In *International Workshop on Combinatorial Algorithms (IWOCA)*, pages 380–384, 2009.

[LSW12] Daniel Lokshtanov, Saket Saurabh, and Magnus Wahlström. Subexponential Parameterized Odd Cycle Transversal on Planar Graphs. In *Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 424–434, 2012.

[LT97] Chin Lung Lu and Chuan Yi Tang. A linear-time algorithm for the weighted feedback vertex problem on interval graphs. *Information Processing Letters*, 61(2):107–111, 1997.

[LY80] John M Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[Mar05] Dániel Marx. Efficient Approximation Schemes for Geometric Problems? In *European Symposium on Algorithms (ESA)*, pages 448–459, 2005.

[Mar06] Dániel Marx. Parameterized Complexity of Independence and Domination on Geometric Graphs. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 154–165, 2006.

[Mat98] Tomomi Matsui. Approximation Algorithms for Maximum Independent Set Problems and Fractional Coloring Problems on Unit Disk Graphs. In *Japanese Conference on Discrete and Computational Geometry (JCDCG)*, pages 194–200, 1998.

[Mis11] S. Mishra. On the Maximum Uniquely Restricted Matching for Bipartite Graphs. *Electronic Notes in Discrete Mathematics*, 37:345–350, 2011.

[Mou18] Lalla Mouatadid. *Efficient Algorithms on Cocomparability Graphs via Vertex Orderings*. PhD thesis, 2018.

[MP15] Apurva Mudgal and Supantha Pandit. Covering, Hitting, Piercing and Packing Rectangles Intersecting an Inclined Line. In *International Conference on Combinatorial Optimization and Applications (COCOA)*, pages 126–137, 2015.

[MR10] Nabil H Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[MRR14] Nabil H. Mustafa, Rajiv Raman, and Saurabh Ray. Settling the APX-Hardness Status for Geometric Set Cover. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 541–550, 2014.

[Nar89] Giri Narasimhan. The maximum k-colorable subgraph problem. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 1989.

[NPR17] Subhas C. Nandy, Supantha Pandit, and Sasanka Roy. Faster approximation for maximum independent set on unit disk graph. *Information Processing Letters*, 127:58–61, 2017.

[NS07] Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

[Pan17] Supantha Pandit. Dominating Set of Rectangles Intersecting a Straight Line. In *Canadian Conference on Computational Geometry (CCCG)*, pages 144–149, 2017.

[PRdSS18] Lucia Draque Penso, Dieter Rautenbach, and Ueverton dos Santos Souza. Graphs in which some and every maximum matching is uniquely restricted. *Journal of Graph Theory*, 89(1):55–63, 2018.

[PW10] Arnaud Pêcher and Annegret K Wagler. Clique and chromatic number of circular-perfect graphs. *Electronic Notes in Discrete Mathematics*, 36:199–206, 2010.

[Raz06] Igor Razgon. Exact Computation of Maximum Induced Forest. In *Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 160–171, 2006.

[RN95] Chong S Rim and Kazuo Nakajima. On rectangle intersection and overlap graphs. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 42(9):549–553, 1995.

[RSHS92] Sailesh K. Rao, P. Sadayappan, Frank K. Hwang, and Peter W. Shor. The Rectilinear Steiner Arborescence Problem. *Algorithmica*, 7(2&3):277–288, 1992.

[RSV04] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[SBS87] J. Spinrad, A. Brandstädt, and L. Stewart. Bipartite Permutation Graphs. *Discrete Applied Mathematics*, 18:279–292, 1987.

[Sch96] Sven Schuierer. An optimal data structure for shortest rectilinear path queries in a simple rectilinear polygon. *International Journal of Computational Geometry & Applications*, 6(02):205–225, 1996.

[Sch09] Marcus Schaefer. Complexity of some geometric and topological problems. In *International Symposium on Graph Drawing*, pages 334–344. Springer, 2009.

[Spi03] Jeremy P. Spinrad. *Efficient graph representations*, volume 19 of *Fields Institute monographs*. American Mathematical Society, 2003.

[SS05] Weiping Shi and Chen Su. The Rectilinear Steiner Arborescence Problem is NP-Complete. *SIAM Journal on Computing*, 35(3):729–740, 2005.

[SSŠ03] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in NP. *Journal of Computer and System Sciences*, 67(2):365–380, 2003.

[SU05] Sebastian Seibert and Walter Unger. A 1.5-approximation of the minimal Manhattan network problem. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 246–255. Springer, 2005.

[Tho98] Mikkel Thorup. Map Graphs In Polynomial Time. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 396–405, 1998.

[Tuc80] Alan Tucker. An efficient test for circular-arc graphs. *SIAM Journal on Computing*, 9(1):1–24, 1980.

[Vaz01] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[vL09] Erik Jan van Leeuwen. *Optimization and approximation on systems of geometric objects*. PhD thesis, University of Amsterdam, 2009.

[W$^+$96] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, NJ, 1996.

[WFP85] Kevin White, Martin Farber, and William Pulleyblank. Steiner trees, connected domination and strongly chordal graphs. *Networks*, 15(1):109–124, 1985.

[Xia13] Ge Xia. The Stretch Factor of the Delaunay Triangulation Is Less than 1.998. *SIAM Journal on Computing*, 42(4):1620–1659, January 2013.

[Yan78] Mihalis Yannakakis. Node- and Edge-Deletion NP-Complete Problems. In *ACM Symposium on Theory of Computing (STOC)*, pages 253–264, 1978.

[Yan81] Mihalis Yannakakis. Node-Deletion Problems on Bipartite Graphs. *SIAM Journal on Computing*, 10(2):310–327, 1981.

[YG87] Mihalis Yannakakis and Fanica Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2):133–137, 1987.

[Zac00] Martin Zachariasen. On the approximation of the rectilinear Steiner arborescence problem in the plane. *Unpublished manuscript, see http://citeseerx. ist. psu. edu/viewdoc/summary*, 2000.

List of Publications

Journal Articles

- Mathew C. Francis, Dalu Jacob, and **Satyabrata Jana**. "Uniquely restricted matchings in interval graphs." In *SIAM Journal on Discrete Mathematics*, 32(1):148-172, 2018.
- **Satyabrata Jana**, and Supantha Pandit. "Covering and packing of rectilinear subdivision." In *Theoretical Computer Science*, 840:166-176, 2020.
- Sujoy Bhore, Sourav Chakraborty, **Satyabrata Jana**, Joseph SB Mitchell, Supantha Pandit, and Sasanka Roy. "The Balanced Connected Subgraph Problem." In *Discrete Applied Mathematics*, 2021.
- **Satyabrata Jana**, Anil Maheshwari, and Sasanka Roy. "Linear Size Planar Manhattan Network for Convex Point Sets." In *Computational Geometry: Theory and Applications*, 2021.

Under Submission in Journals

- **Satyabrata Jana**, Anil Maheshwari, Saeed Mehrabi, and Sasanka Roy. "Maximum Bipartite Subgraph of Geometric Intersection Graphs." In *Theoretical Computer Science* - Under Major revision
- Sujoy Bhore, **Satyabrata Jana**, Supantha Pandit, and Sasanka Roy. "Balanced Connected Subgraph Problem in Geometric Intersection Graphs." In *Discrete Applied Mathematics* - Under Major revision

Conference Proceedings

- **Satyabrata Jana**, Supantha Pandit. "Covering and packing of rectilinear subdivision." In *International Workshop on Algorithms and Computation*, pp. 381-393. Springer, 2019
- Sujoy Bhore, Sourav Chakraborty, **Satyabrata Jana**, Joseph SB Mitchell, Supantha Pandit, and Sasanka Roy. "The balanced connected subgraph problem." In *Conference on Algorithms and Discrete Applied Mathematics*, pp. 201-215. Springer, 2019.
- Sujoy Bhore, **Satyabrata Jana**, Supantha Pandit, and Sasanka Roy. "Balanced connected subgraph problem in geometric intersection graphs." In *International Conference on Combinatorial Optimization and Applications*, pp. 56-68. Springer, 2019.
- **Satyabrata Jana**, Anil Maheshwari, Saeed Mehrabi, and Sasanka Roy. "Maximum bipartite subgraph of geometric intersection graphs." In *International Workshop on Algorithms and Computation*, pp. 158-169. Springer, 2020.