Algorithmic and Combinatorial Questions

on some

Geometric Problems

on

Graphs

A THESIS
SUBMITTED FOR THE DEGREE OF

Doctor of Philosophy

IN THE FACULTY OF ENGINEERING

BY

Jasine Babu

Department of Computer Science and Automation
Indian Institute of Science
Bangalore - 560012

February 2014

to

MuSa
for his companionship,
both emotional and intellectual.

Acknowledgements

Sunil has been a nice supervisor and friend and it was really enjoyable working with him. The quality time he spent with us students and his flexibility are things that we always took for granted; but often made others envy about. I remember telling him that I would like working on problems having an algorithmic flavor, though it was not his usual nature of work. Now when I look back, I see that most of the works we did together are indeed algorithmic in nature, while using structural observations.

Sunil's graph theory course in our first year was interesting both because of its contents and also its active participants: Deepak, Reshmi, Nihar, Nitin Singh, Musthafa, Arunselvan, Bruhati and more. I was fortunate to attend classes of some of the best teachers at the mathematics department of IISc; in particular, probability courses offered by Manjunath. Those classes have helped me a lot in improving my understanding of the the foundations of mathematics.

Abhijin, Manu and Deepak have been my co-authors in multiple works. Working together and proving theorems had been a pleasure and an intellectually stimulating experience. I enjoyed interactions and discussions with Rogers, Neeldhara, Prachi, Meghana, Prabhanjan, Mathew, Anita, Arunselvan, Pradeesha - friends in our theory group in past and present. It was quite interesting working with our junior interns Roohani and Krishna as well.

I would also like to thank Prof. Anil Maheshwari and Prof. Michiel Smid for the time they offered me during my short term visit to Carleton University. Ahmad was a wonderful collaborator, though our communications were only over e-mails.

The CSA department has provided us a nice environment to work in. I am grateful to the chairman for his genuine interest in student welfare. He has put a lot of effort in arranging for travel grants, on more than one occasion when I was in need. I also thank MSR India for their conference travel support.

Above everything, it had been my friends who made my PhD life really enjoyable and memorable. Over-the-tea discussions with them, our outings and trips to western ghats were all re-energizing and cheering me up. I am not mentioning their names here; they had always been there and I hope to continue sharing my moments with them.

Looking once again towards the path traced so far, I consider myself fortunate to be in the right circumstances with the right people around, at every crucial point of my life. My heartfelt thanks to all of them who lead me on and lent me their hands to hold on.

Though I was a teacher for only a short span, the sparkling in the eyes of my students is still the biggest motivation for me forward.

Abstract

This thesis mainly focuses on algorithmic and combinatorial questions related to some geometric problems on graphs. In the last part of this thesis, a graph coloring problem is also discussed.

Boxicity and Cubicity: These are graph parameters dealing with geometric representations of graphs in higher dimensions. Both these parameters are known to be NP-Hard to compute in general and are even hard to approximate within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$, under standard complexity theoretic assumptions.

We studied algorithmic questions for these problems, for certain graph classes, to yield efficient algorithms or approximations. Our results include a polynomial time constant factor approximation algorithm for computing the cubicity of trees and a polynomial time constant (≤ 2.5) factor approximation algorithm for computing the boxicity of circular arc graphs. As far as we know, there were no constant factor approximation algorithms known previously, for computing boxicity or cubicity of any well known graph class for which the respective parameter value is unbounded.

We also obtained parameterized approximation algorithms for boxicity with various edit distance parameters. An $o(n)$ factor approximation algorithm for computing the boxicity and cubicity of general graphs also evolved as an interesting corollary of one of these parameterized algorithms. This seems to be the first sub-linear factor approximation algorithm known for computing the boxicity and cubicity of general graphs.

Planar grid-drawings of outerplanar graphs: A graph is outerplanar, if it has a planar embedding with all its vertices lying on the outer face. We give an efficient algorithm to 2-vertex-connect any connected outerplanar graph G by adding more edges to it, in order to obtain a supergraph of G such that the resultant graph is still outerplanar and its pathwidth is within a constant times the pathwidth of G. This algorithm leads to a constant factor approximation algorithm for computing minimum height planar straight line grid-drawings of outerplanar graphs, extending the existing algorithm known for 2-vertex connected outerplanar graphs.

Maximum matchings in triangle distance Delaunay graphs: Delaunay graphs of point sets are well studied in Computational Geometry. Instead of the Euclidean metric, if the Delaunay graph is defined with respect to the convex distance function defined by an equilateral triangle, it is called a Triangle Distance Delaunay graph. TD-Delaunay graphs are known to be equivalent to geometric spanners called half-Θ_6 graphs.

It is known that classical Delaunay graphs of point sets always contain a near perfect matching, for non-degenerate point sets. We show that Triangle Distance Delaunay graphs of a set of n points in general position will always contain a matching of size $\left\lceil \frac{n-1}{3} \right\rceil$ and this bound is tight. We also show that Θ_6 graphs, a class of supergraphs of half-Θ_6 graphs, can have at most $5n - 11$ edges, for point sets in general position.

Heterochromatic Paths in Edge Colored Graphs: Conditions on the coloring to guarantee the existence of long heterochromatic paths in edge colored graphs is a well explored problem in literature. The objective here is to obtain a good lower bound for $\lambda(G)$ - the length of a maximum heterochromatic path in an edge-colored graph G, in terms of $\vartheta(G)$ - the minimum color degree of G under the given coloring. There are graph families for which $\lambda(G) = \vartheta(G) - 1$ under certain colorings, and it is conjectured that $\vartheta(G) - 1$ is a tight lower bound for $\lambda(G)$.

We show that if G has girth is at least $4\log_2(\vartheta(G))+2$, then $\lambda(G) \geq \vartheta(G) - 2$. It is also proved that a weaker requirement that G just does not contain four-cycles is enough to guarantee that $\lambda(G)$ is at least $\vartheta(G) - o(\vartheta(G))$. Other special cases considered include lower bounds for $\lambda(G)$ in edge colored bipartite graphs, triangle-free graphs and graphs without heterochromatic triangles.

Contributions to literature

1. **A constant factor approximation algorithm for boxicity of circular arc graphs**
   with Abhijin Adiga, and L. Sunil Chandran
   In: Algorithms and Data Structures Symposium (WADS 2011). Springer -Verlag Lecture Notes in Computer Science, vol. 6844/2011, pp. 13-24.
   `http://dx.doi.org/10.1007/978-3-642-22300-6_2`

2. **Polynomial Time and Parameterized Approximation Algorithms for Boxicity**
   with Abhijin Adiga, and L. Sunil Chandran
   In: 7th International Symposium on Parameterized and Exact Computation (IPEC 2012). Springer - Verlag Lecture Notes in Computer Science, vol. 7535/2012, pp. 135-146.
   `http://dx.doi.org/10.1007/978-3-642-33293-7_14`

3. **Fixed-Orientation Equilateral Triangle Matching of Point Sets**
   with Ahmad Biniaz, Anil Maheshwari, and Michiel H. M. Smid.
   To Appear in: Theoretical Computer Science.
   `http://dx.doi.org/10.1016/j.tcs.2013.11.031`

4. **2-connecting Outerplanar Graphs without Blowing Up the Pathwidth**
   with Manu Basavaraju, L. Sunil Chandran, and Deepak Rajendraprasad.
   To Appear in: Theoretical Computer Science.
   `http://dx.doi.org/10.1016/j.tcs.2014.04.032`

5. **Heterochromatic paths in edge colored graphs without small cycles and heterochromatic-triangle-free graphs**
   with L. Sunil Chandran, and Deepak Rajendraprasad.
   Submitted to European Journal of Combinatorics.

6. **Approximating the Cubicity of Trees**
   with Manu Basavaraju, L. Sunil Chandran, Deepak Rajendraprasad, and Naveen Sivadasan.
   `http://arxiv.org/abs/1402.6310`

Contents

Chapter 1

Introduction

In this chapter, we give a brief introduction to the topics discussed in this thesis and give an overview about the organization of this thesis. Detailed descriptions of the respective topics are included in later chapters.

1.1 Boxicity and cubicity

Suppose each vertex of a graph G can be associated with an axis parallel box in the d-dimensional Euclidean space so that two boxes intersect if and only if the corresponding vertices are adjacent in G. Such a representation is called a d-dimensional box representation of G. Boxicity of a graph G, denoted by $\text{box}(G)$, is the minimum dimension d for which G has a d-dimensional box representation. If the axis-parallel boxes are further restricted to be d-dimensional unit hypercubes, the corresponding parameter is called cubicity, denoted by $\text{cub}(G)$, and the corresponding intersection representation is called a d-dimensional cube representation of G. (See Figure 1.1). Since a cube representation is also a box representation, $\text{box}(G) \leq \text{cub}(G)$. By convention, cubicity and boxicity of a complete graph are zero.

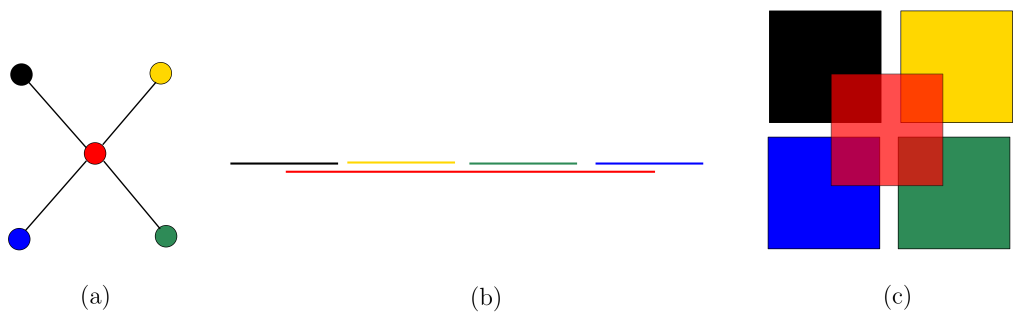


Figure 1.1: (a) A graph G on 5 vertices (b) a one-dimensional box representation of G (c) a two-dimensional cube representation of G.

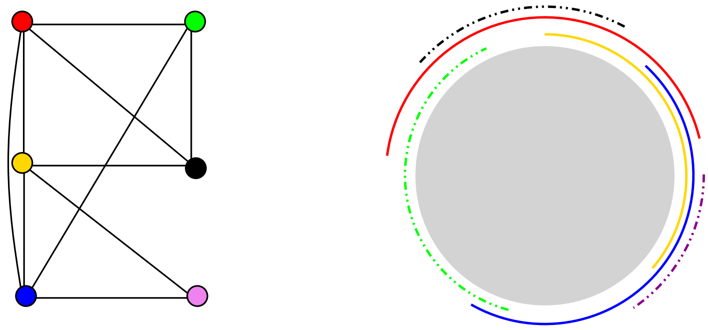


Figure 1.2: A circular arc graph and its circular arc representation

In the special case when $d = 1$, a box (resp. cube) is just an interval (resp. unit interval) on the real line. In this case, the graph is an intersection graph of intervals (resp. unit intervals) and we call it an interval (resp. unit interval) graph.

These parameters were introduced by F. S. Roberts [78] in 1968 for studying some problems in Ecology. Knowing a low dimensional box representation allows space efficient representation for dense graphs. Some well known NP-hard problems like the max-clique problem becomes polynomial time solvable [80], if a low dimensional box representation of the graph is known. Boxicity is also studied in relation with other dimensional parameters of graphs like partial order dimension and threshold dimension [4, 3, 98].

Boxicity and cubicity of a graph on n vertices are at most $\left\lfloor \frac{n}{2} \right\rfloor$ and $\left\lceil \frac{2n}{3} \right\rceil$ respectively [78]. Bounds of boxicity in terms of parameters like maximum degree [46, 4], minimum vertex cover size [27] and tree-width [29] are known. It was shown by Scheinerman [81] in 1984 that the boxicity of outer planar graphs is at most two. In 1986, Thomassen [91] proved that the boxicity of planar graphs is at most 3.

In polynomial time we can decide whether a graph G has boxicity (resp. cubicity) one, because interval (resp. unit interval) graphs are recognizable in polynomial time. However, given a graph G and an integer k, deciding whether $\mathrm{box}(G) \leq k$ (resp. $\mathrm{cub}(G) \leq k$) is NP-Hard, even when $k = 2$ or $k = 3$ [38, 98, 64, 18]. Further, boxicity and cubicity are hard to approximate in polynomial time: these are inapproximable within an $O(n^{1-\epsilon})$-factor for any $\epsilon > 0$, unless $NP = ZPP$ [25]. This hardness result holds for restricted graph classes like bipartite, co-bipartite and split graphs as well. Even for special classes of graphs, there were not many approximation algorithms known to exist for these problems.

In Chapter 2, we discuss the boxicity of circular arc graphs - intersection graphs of arcs on a circle. Figure 1.2 shows a circular arc graph and its circular arc intersection representation. We show that if a circular arc graph is co-bipartite, then its boxicity is computable in polynomial time. Using this result, we derive a polynomial time constant factor approximation algorithm for computing the boxicity of circular arc graphs. Given any circular arc graph G, this algorithm computes a box representation of G of dimension at most $2\,\mathrm{box}(G) + 1$. Using this, a cube representation of G of dimension at most $2\,\mathrm{cub}(G) + \log n$ is also derived in polynomial time.

In Chapter 3, we present a randomized algorithm that runs in polynomial time and computes cube representations of trees, of dimension within a constant factor of the optimum. If we do not insist for a cube representation, then the cubicity of trees can be approximated within a constant factor in polynomial time, without using any randomization.

In Chapter 4, we derive an $O\left(\frac{n\sqrt{\log\log n}}{\sqrt{\log n}}\right)$ factor approximation algorithm for computing the boxicity of general graphs and an $O\left(\frac{n(\log\log n)^{\frac{3}{2}}}{\sqrt{\log n}}\right)$ factor approximation algorithm for computing the cubicity of general graphs. These algorithms are derived as corollaries of one of the parameterized approximation algorithms for boxicity described in the same chapter. To our knowledge, these are the first $o(n)$ factor approximation algorithms for boxicity and cubicity of general graphs.

We also give some parameterized approximation algorithms for cubicity in this chapter.

1.2 Planar grid-drawings of outerplanar graphs

Computing planar straight line drawings of planar graphs, with their vertices placed on a two dimensional grid, is a well known problem in graph drawing. In Figure 1.3, a planar graph and its planar straight line grid drawing are shown. In 1990, Schnyder [82] showed that any planar graph on n vertices has a planar straight line drawing on an $(n-1) \times (n-1)$ sized grid.

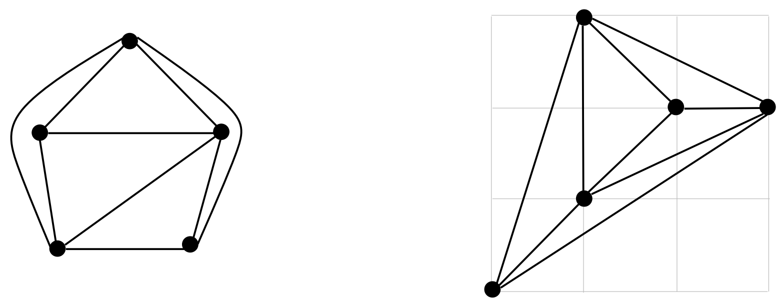


Figure 1.3: A planar graph on 5 vertices and its straight line planar drawing on a 4×4 grid

A well studied optimization problem in this context is to minimize the height (i.e. the smaller of the two dimensions) of the grid on which the drawing

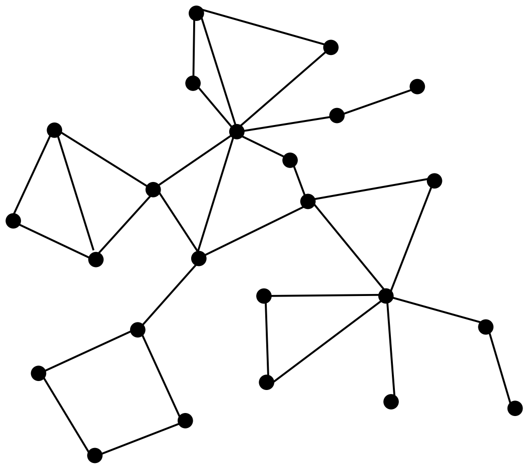


Figure 1.4: An outerplanar embedding of an outerplanar graph

is made. Pathwidth of a graph, a structural parameter widely used in graph drawing and layout problems, is a lower bound for the height of the grid on which the graph can be drawn. In general, the grid height required by a planar graph is not necessarily upper bounded by a function of its pathwidth. However, for some special cases, like that of trees, efficient algorithms that compute a planar straight line drawing of the tree on a grid of height at most a constant times its pathwidth is known; giving a constant factor approximation for the optimization problem.

A graph $G(V, E)$ is outerplanar, if it has a planar embedding with all its vertices lying on the outer face. See Figure 1.4 for an example. Outerplanar graphs form a superclass of trees. For 2-vertex-connected outerplanar graphs, Biedl [14] obtained an algorithm that computes a planar straight line drawing of the graph on a grid of height at most a constant times its pathwidth. It was left as an open problem to extend this algorithm to work for arbitrary outerplanar graphs. We address this problem in Chapter 5.

To solve this problem, it is enough to design an algorithm for adding edges to a given outerplanar graph G to obtain a 2-vertex-connected supergraph G' of G that is still outerplanar and having pathwidth at most a constant times the pathwidth of G. To obtain a planar straight line drawing of G, we just need to compute a planar straight line drawing of G' using Biedl's algorithm and delete the edges not originally present in G. Though bi-connecting a graph is easy, simultaneously maintaining the outerplanarity and the pathwidth conditions in the process is non-trivial. In Chapter 5, we give algorithm to do this in $O(n \log n)$ time.

1.3 Matchings in TD-Delaunay graphs - Equilateral triangle matchings

A downward equilateral triangle is an equilateral triangle with one of its sides parallel to the x-axis and the corner opposite to this side below the side parallel to the x-axis. Given a point set P, the maximum $\bigtriangledown$-matching problem is to compute a maximum cardinality family $\mathcal{F}$ of downward equilateral triangles such that (i) no point from P belongs to more than one $\bigtriangledown$ in $\mathcal{F}$ and (ii) exactly two points from P lie inside each $\bigtriangledown$ in $\mathcal{F}$. A point set and one of its maximum $\bigtriangledown$-matchings is shown in Figure 1.5. Similar questions with other geometric

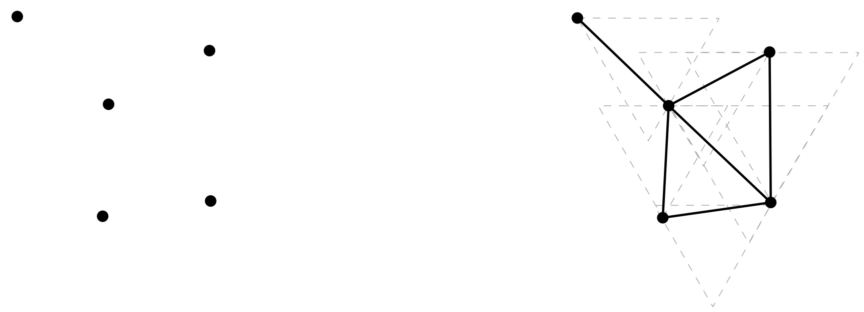
(a) (b)

Figure 1.5: A point set P and a maximum $\bigtriangledown$-matching of P

shapes like circles or axis parallel rectangles instead of downward equilateral triangles have been studied in literature [1, 41, 11].

In Chapter 6, we obtain a lower bound for the cardinality of maximum $\bigtriangledown$-matchings of point sets, in terms of the number of points. To do this, it is convenient to map the problem into a graph theoretic setting, by defining an associated geometric graph as follows. Given a point set P, define $G_{\bigtriangledown}(P)$ to be a geometric graph with vertex set P such that any two vertices p and q are adjacent if and only if there is some downward equilateral triangle containing both p and q but no other point from P. (See Figure 1.6). It is not difficult to see that the cardinality of a maximum $\bigtriangledown$-matching of P is the same as the cardinality of a maximum matching in $G_{\bigtriangledown}(P)$. (Here, a maximum matching in $G_{\bigtriangledown}(P)$ is a maximum cardinality subset M of the edges of $G_{\bigtriangledown}(P)$ such that no two edges in M share a common end-point.)

We prove some structural and geometric properties of the geometric graph mentioned above. In our context, a point set P is said to be in general position, if the line passing through any two points from P does not make angles 0°, 60° or 120° with the horizontal. We show that for point sets P in general position, $G_{\bigtriangledown}(P)$ always contains a matching of size at least $\left\lceil \frac{|P|-1}{3} \right\rceil$. We also give examples of point sets for which this bound is tight.



(a) (b)

Figure 1.6: A point set P and its $G_{\triangledown}(P)$ graph. Edges of $G_{\triangledown}(P)$ are shown using thick lines.

For point sets in general position, the geometric graph we defined above is equivalent to the well known Triangle Distance Delaunay graphs [15]. These are also equivalent to a class of geometric spanners called half θ_6 graphs [15]. Thus $\left\lceil \frac{|P|-1}{3} \right\rceil$ becomes a tight lower bound for the cardinality of maximum matchings in triangle distance Delaunay graphs. In contrast, classical Delaunay graphs for non-degenerate point sets are guaranteed to contain a matching of size at least $\left\lfloor \frac{|P|}{2} \right\rfloor$ [41].

In this chapter we also prove some structural properties of a related class of geometric spanners called θ_6 graphs.

1.4 Heterochromatic paths in edge colored graphs

An edge coloring of graph is a mapping that assigns a color to each edge of the graph. If a graph G has an edge coloring specified, we call G an edge colored graph. The minimum color degree of an edge colored graph G, denoted by $\vartheta(G)$, is the minimum number of distinct colors occurring at edges incident at

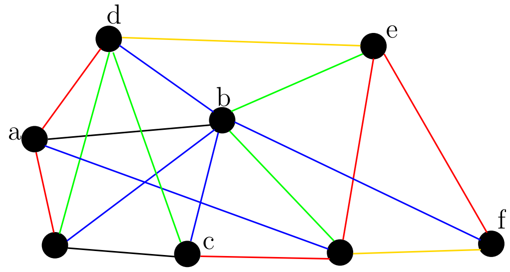


Figure 1.7: An edge colored graph. According to this coloring, the minimum color degree is 3. The path a,b,c,d,e,f is a heterochromatic path of length 5 in G.

any vertex v of G. (See Figure 1.7).

A subgraph H of an edge colored graph G is said to be heterochromatic if edges of H are all distinctly colored. The conditions on the coloring to guarantee the existence of heterochromatic Hamiltonian paths and cycles in edge colored graphs are well studied in literature [59, 44, 8, 45]. A variant of this problem is to obtain conditions that guarantee long heterochromatic paths in edge colored graphs.

The relationship between the minimum color degree $\vartheta(G)$ of an edge colored graph G and the length of its maximum length heterochromatic path $\lambda(G)$ is also well investigated [19, 32, 34, 39]. It is conjectured that for every edge colored graph G, $\lambda(G) \geq \vartheta(G) - 1$ [32]. If this conjecture is true, $\vartheta(G) - 1$ would be a tight lower bound for $\lambda(G)$, since there are graph families for which $\lambda(G) = \vartheta(G) - 1$ under certain colorings.

In Chapter 7, we investigate this conjecture for graphs without small cycles. We show that if G has no cycles of length smaller than $4\log_2(\vartheta(G)) + 2$, then $\lambda(G) \geq \vartheta(G) - 2$, which is only one less than the bound conjectured for the general case. It is also proved that $\lambda(G)$ is at least $\vartheta(G) - o(\vartheta(G))$, if a weaker requirement that G just does not contain four-cycles holds.

Another result in Chapter 7 is an improved lower bound of $\lambda(G)$ for edge colored graphs not containing heterochromatic triangles in it. Other results in this chapter include lower bounds for $\lambda(G)$ in edge colored bipartite graphs and triangle-free graphs. We also give a short and simple proof showing that for any edge colored graph G, $\lambda(G) \geq \left\lceil \frac{2\vartheta(G)}{3} \right\rceil$.

Chapter 2

A constant factor approximation algorithm for the boxicity of circular arc graphs

In this chapter[1], we consider the problem of approximating the boxicity (resp. cubicity) of circular arc graphs - intersection graphs of arcs of a circle. Circular arc graphs are known to have unbounded boxicity, which could be as large as $\Omega(n)$. We give a $\left(2+\frac{1}{k}\right)$-factor (resp. $\left(2+\frac{\lceil \log n \rceil}{k}\right)$-factor) polynomial time approximation algorithm for computing the boxicity (resp. cubicity) of any circular arc graph, where k is the value of the optimum solution. For normal circular arc (NCA) graphs, with an NCA model given, this can be improved to an additive two approximation algorithm. The time complexity of the algorithms to approximately compute the boxicity (resp. cubicity) is $O(mn+n^2)$ in both these cases, where n is the number of vertices of the graph and m is its number of edges. In $O(mn+kn^2) = O(n^3)$ time we get their corresponding box (resp. cube) representations. Our additive two approximation algorithm directly works for any proper circular arc graph, since their NCA models can be computed in polynomial time.

This seems to be the first result obtaining a polynomial time algorithm with a sublinear approximation factor for computing boxicity, of any well known graph class of unbounded boxicity.

[1]Joint work with Abhijin Adiga and L. Sunil Chandran. An initial version of this work was presented in WADS 2011. A complete version is under revision in Discrete Applied Mathematics.

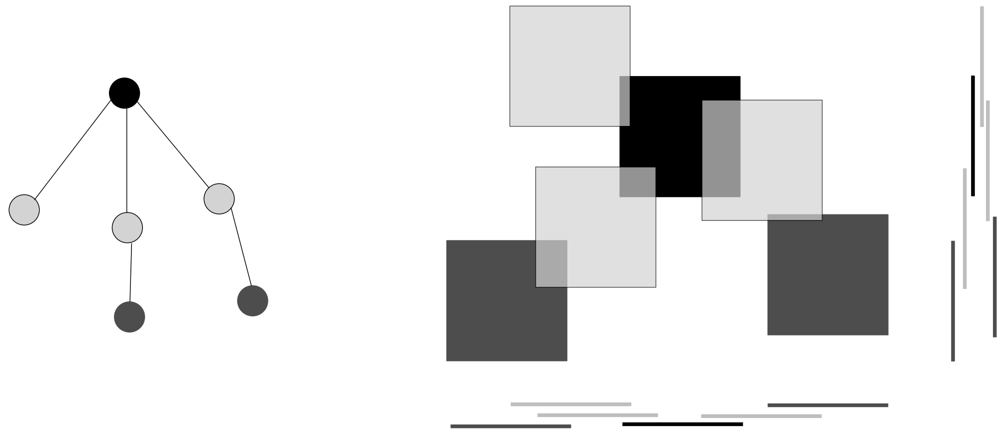


Figure 2.1: A graph and its 2-dimensional cube representation. The projections to X and Y axes give two unit interval graphs.

2.1 Introduction

Let $G(V, E)$ be a graph. Recall that in Section 1.1, we defined a d-dimensional box (resp. cube) representation of G as a geometric representation where each vertex is associated with an axis parallel box (resp. axis parallel unit hypercube) in $\mathbb{R}^k$ so that two boxes (resp. hypercubes) intersect if and only if the corresponding vertices are adjacent in G. It is easy to see that projecting this geometric representation to any of the d coordinate axes gives an interval (resp. unit interval) supergraph of G (see Figure 2.1). Moreover, if $I_1, I_2, \cdots, I_d$ are these interval graphs, we have $V(I_i) = V(G)$ for each $1 \leq i \leq d$ and $E(G) = E(I_1) \cap E(I_2) \cap \cdots \cap E(I_d)$. Conversely, given interval (resp. unit interval) supergraphs $I_1, I_2, \cdots, I_d$ of G satisfying $V(I_i) = V(G)$ for each $1 \leq i \leq d$ and $E(G) = E(I_1) \cap E(I_2) \cap \cdots \cap E(I_d)$, we can also construct a d-dimensional box (cube) representation of G. This leads to a combinatorial re-definition of the terms as follows.

Definition 2.1 (Box (resp. cube) representation [78])**.** If $I_1, I_2, \cdots, I_k$ are interval graphs (resp. unit interval graphs) on the vertex set $V(G)$ such that $E(G) = E(I_1) \cap E(I_2) \cap \cdots \cap E(I_k)$, then $\{I_1, I_2, \cdots, I_k\}$ is called a k-dimensional box (resp. cube) representation of G.

Definition 2.2 (Boxicity (resp. Cubicity)[78])**.** The boxicity (resp. cubicity) of G is the minimum integer k such that G has a k-dimensional box (resp. cube) representation as given by Definition 2.1.

Given a graph G and an integer k, the decision problem BOXICITY (resp. CUBICITY) asks whether $\text{box}(G) \leq k$ (resp. $\text{cub}(G) \leq k$). In 1981, Cozzens [38] showed that this problem is NP-hard. Later Yannakakis [98] proved that deciding $\text{box}(G) \leq 3$ itself is NP-complete. Kratochvil[64] showed that deciding even $\text{box}(G) \leq 2$ is NP-complete. In 2010, Adiga et al. [3] proved that no polynomial time algorithm for approximating the boxicity of bipartite graphs with approximation factor $O(n^{0.5-\epsilon})$ is possible unless NP = ZPP. A very recent work [25] improved this hardness result to $O(n^{1-\epsilon})$ factor. Same non-approximability holds in the case of split graphs and co-bipartite graphs too. Since the cubicity and the boxicity of a co-bipartite graph are always equal, no polynomial time algorithm for approximating cubicity of co-bipartite graphs with approximation factor $O(n^{1-\epsilon})$ is possible unless NP = ZPP.

The boxicity and cubicity of some special graph classes have been studied earlier. It was shown by Scheinerman [81] in 1984 that the boxicity of outer planar graphs is at most two. In 1986, Thomassen [91] proved that the boxicity of planar graphs is at most 3. Recently, an alternate proof of the same result was obtained by Felsner et al. [50]. Boxicity of special classes of graphs was studied in [10, 77] too. Bounds for the cubicity of interval graphs were obtained by Chandran et al. [28] and Adiga et al. [5]. Bhowmic et al. [12] obtained bounds for the boxicity of circular arc graphs.

Circular arc (CA) graphs are intersection graphs of arcs on a circle. That is, an arc of the circle is associated with each vertex and two vertices are adjacent if and only if their corresponding arcs overlap. CA graphs became popular in 1970's with a series of papers from Tucker [94, 95]. A proper circular arc (PCA) graph is a graph which has some CA representation in which no arc is properly contained in another. A normal circular arc (NCA) graph is one which has a CA representation in which there are no pairs of arcs together covering the circumference of the entire circle. For a comprehensive survey on CA graphs, refer to Lin et al. [66].

In this chapter, we study the boxicity and cubicity of circular arc graphs. A fundamental difficulty while dealing with CA graphs in comparison with interval graphs is the absence of Helly property: a set of pairwise intersecting arcs need not share a common intersection point (whereas, a set of pairwise intersecting intervals always has a common intersection point). Some of the well known NP-complete problems like tree-width and path-width are known to be polynomial time solvable in the case of CA graphs [87, 89]. However, unlike interval graphs, problems like minimum vertex coloring [54] and branch-width [71] remain NP-complete for CA graphs. We believe that BOXICITY and CUBICITY belong to the second category. There exist circular arc graphs of arbitrarily high boxicity, including the well known Robert's graph (the complement of a perfect matching on n vertices, with n even) which has boxicity $\frac{n}{2}$.

We consider the problem of approximating the boxicity (resp. cubicity) of circular arc graphs. We give a polynomial time constant factor approximation algorithm for computing the boxicity of circular arc graphs. A polynomial time algorithm for approximating the cubicity of circular arc graphs is also obtained, which gives a constant factor approximation up to an additive error of $\log n$.

This seems to be the first result obtaining a polynomial time algorithm with a sublinear approximation factor for computing the boxicity (resp. cubicity) of any well known graph class of unbounded boxicity.

Our main results in this chapter are:

(a) The boxicity of any circular arc graph can be approximated within a $(2 + \frac{1}{k})$-factor in polynomial time where $k \geq 1$ is the boxicity of the graph.

(b) The boxicity of any normal circular arc graph can be computed within an additive error of two in polynomial time, given a normal circular arc model of the graph. (Note that, for some important subclasses of NCA graphs such as proper circular arc graphs, an NCA model can be computed in polynomial time [86, 95].)

(c) Cubicity of any circular arc graph can be approximated within a $\left(2 + \frac{\lceil \log n \rceil}{k}\right)$-factor in polynomial time, where $k \geq 1$ is the cubicity of the graph and n is its number of vertices.

(d) The time complexity of the algorithms to approximately compute the boxicity (resp. cubicity) is $O(mn + n^2)$ in all the above cases and in $O(mn + kn^2) = O(n^3)$ time we also get their corresponding box(cube) representations, where n is the number of vertices, m is the number of edges and k is the boxicity (resp. cubicity) of the given graph.

A structural result we obtained in this chapter may be of independent interest. The following way of constructing an auxiliary graph H^* of a given graph H is from [2].

Definition 2.3. Given a graph $H = (V, E)$, consider the graph H^* constructed as follows: $V(H^*) = \{\Gamma_e \mid e \in E(H)\}$, and for any pair of edges wx and yz of H, the corresponding vertices Γ_{wx} and Γ_{yz} of H^* are adjacent if and only if the induced subgraph of H on the vertex set $\{w, x, y, z\}$ is a $2K_2$ i.e. a matching of size two. (In other words, H^* is the complement of $[L(H)]^2$, the square of the line graph $L(H)$ of H.)

Structural properties of H^* and its complement are often found useful in relation with problems like largest induced matching problem and minimum chain cover problem. A consolidation of these properties can be found in Golumbic et al. [55] and Cameron et al. [23]. The following is an intermediate structural result we obtained:

(e) In Lemma 2.14, we observe that if H is a bipartite graph whose complement is a CA graph, then H^* is a comparability graph.

This is a generalization of similar results known for smaller classes like convex bipartite graphs and interval bigraphs [2, 99]. This result helps us in reducing the complexity of our polynomial time algorithms mentioned before.

Organization of this chapter. In Section 2.2, we introduce the notations used in this chapter. In Section 2.3, we derive a polynomial time algorithm to compute optimum box representations for a subclass of circular arc graphs, namely co-bipartite CA graphs. In Section 2.4, we describe a polynomial time algorithm to compute a box representation of dimension at most two more than the optimum, of any Normal CA graph, with its normal CA model given. We do this by representing the input graph as the intersection of a co-bipartite CA graph and an interval graph and then use the algorithm for co-bipartite CA graphs as a subroutine.

In Section 2.5, we describe a polynomial time algorithm to construct a box representation of any arbitrary CA graph G, of dimension at most $2\,\mathrm{box}(G)+1$. To do this, we first represent G as the intersection of a co-bipartite graph G_0 and two interval graphs, derived from G. Exploiting a "nice" adjacency structure possessed by CA graphs, we then show that G_0 is a co-bipartite CA graph, and hence its optimal box representation can be computed in polynomial time. Using the optimal box representation of G_0, we construct the required box representation of G.

In Section 2.6, we analyze some structural properties of co-bipartite CA graphs and use them to reduce the time complexity of the algorithm obtained in Section 2.3, for computing optimal box representations of co-bipartite CA graphs. Since this algorithm is a subroutine in our remaining algorithms, the time complexities of the remaining algorithms are also improved. In Section 2.7, we use the algorithm of Section 2.5 and derive a polynomial time algorithm to compute a cube representation of any CA graph G, of dimension at most $2\,\mathrm{cub}(G) + \lceil \log n \rceil$.

2.2 Notations used in this chapter

We denote the vertex set of a given graph G by $V(G)$ and edge set by $E(G)$. We call a graph G as the union of graphs G_1, G_2, $\cdots$, G_k if they are graphs on the same vertex set and $E(G) = E(G_1) \cup E(G_2) \cup \cdots \cup E(G_k)$. Similarly, a graph G is the intersection of graphs G_1, G_2, $\cdots$, G_k if they are graphs on the same vertex set and $E(G) = E(G_1) \cap E(G_2) \cap \cdots \cap E(G_k)$. We use $\chi(G)$ to denote the chromatic number of G. If a vertex v is adjacent to every other vertex in the graph, then we call it a universal vertex in the graph.

A circular-arc (CA) model $M = (C, \mathcal{A})$ consists of a circle C, together with a family $\mathcal{A}$ of arcs of C. It is assumed that C is always traversed in the clockwise direction, unless stated otherwise. The arc A_v corresponding to a vertex v is denoted by $[s(v), t(v)]$, where $s(v)$ and $t(v)$ are the extreme points of A_v on C with $s(v)$ its start point and $t(v)$ its end point respectively, in the clockwise direction. Without loss of generality, we assume that no single arc of $\mathcal{A}$ covers C and no arc is empty or a single point.

An interval model I consists of a family of intervals on real line. An interval I_v corresponding to a vertex v is denoted by a pair $[l_v(I), r_v(I)]$, where $l_v(I)$ and $r_v(I)$ are the left and right end points of the interval I_v. Without loss of generality, we assume that an interval is always non-empty and is not a single point. We may use I to represent both an interval graph and its interval model, when the meaning is clear from the context.

2.3 Computing the boxicity of co-bipartite CA graphs in polynomial time

A graph is called a co-bipartite CA graph if it is a circular arc graph and also a co-bipartite graph (complement of a bipartite graph). Using some theorems in the literature, we show that, if G is a co-bipartite CA graph, then the computation of $\text{box}(G)$ is equivalent to the computation of the chromatic number of an associated perfect graph, which is polynomial time solvable.

A bipartite graph is chordal bipartite, if it does not contain any induced cycle of length ≥ 6. The term edge-asteroid is used only in the statement of the theorem below and we do not require this term later in this chapter.

Theorem 2.1 (Feder, Hell and Huang [47]). *A graph G is a co-bipartite CA graph if and only if its complement is chordal bipartite and contains no edge-asteroids.*

A bipartite graph is called a chain graph if it does not contain any induced $2K_2$ (a matching containing two edges). The minimum chain cover number of G, denoted by $ch(G)$, is the minimum number of chain subgraphs of G such that the union of their edge sets is $E(G)$.

Recall Definition 2.3 of H^* from Section 2.1. If I is an independent set in a graph G and I' is a maximal independent set in G such that $I' \supseteq I$, then we say that I' is a maximal independent set obtained by extending I. The following theorem is just a restatement of some results in Abueida et al. [2].

Theorem 2.2 (Abueida, Busch and Sritharan [2]). *If H is a bipartite graph with no induced cycles on exactly 6 vertices, then*

1. *$ch(H) = \chi(H^*)$.*

2. *Every maximal independent set of H^* corresponds to the edge-set of a chain subgraph of H. Moreover, the family of maximal independent sets obtained by extending the color classes of an optimum coloring of H^* corresponds to a minimum chain cover of H.*

3. *In the more restricted case where H is chordal bipartite, H^* is a perfect graph and therefore, $ch(H)$ and a chain cover of H of minimum cardinality can be computed in polynomial time.*

Proof. The first two parts of this theorem directly follows from the proof[2] of Theorem 1 in Abueida et al. [2]. The third part of the theorem is a direct consequence of Proposition 1 of the same paper together with the fact that H^* is a perfect graph when H is chordal bipartite [2]. □

The following theorem directly follows from the proof[2] of Lemma 5 in Yannakakis [98].

Theorem 2.3 (Yannakakis [98])**.** *Let G be the complement of a bipartite graph H. Then,* $\operatorname{box}(G) = ch(H)$*. Further, if $H_1, H_2, \cdots, H_k$ are chain subgraphs whose union is H, their respective complements $G_1, G_2, \cdots, G_k$ are interval supergraphs of G whose intersection is G.*

By Theorem 2.1, if $G = \overline{H}$ is a co-bipartite CA graph, then H is chordal bipartite. Hence by Theorem 2.2, a chain cover of H of minimum cardinality can be computed in polynomial time and $ch(H) = \chi(H^*)$. Combining this with Theorem 2.3, we get:

Theorem 2.4. *If G is a co-bipartite CA graph, then* $\operatorname{box}(G) = \chi(H^*)$ *and the family of maximal independent sets obtained by extending the color classes of an optimum coloring of H^* corresponds to the complements of interval supergraphs in an optimal box representation of G. Moreover,* $\operatorname{box}(G)$ *and an optimal box representation of G are computable in polynomial time.*

Remark 2.1. *It may be noted that $G_1, G_2, \cdots, G_k$ of Theorem 2.3 are not just interval supergraphs of the co-bipartite CA graph G, but they are unit interval graphs too [98]. Hence,* $\operatorname{cub}(G) = \operatorname{box}(G)$ *and the box representation we obtained in Theorem 2.4 is also an optimal cube representation of G.*

2.4 An additive two approximation algorithm for computing the boxicity of normal CA graphs

In this section, we show how to compute a box representation of a normal CA graph G of dimension at most $\operatorname{box}(G) + 2$ in polynomial time, when a normal CA model $M(C, \mathcal{A})$ of G is given. We do this by constructing three graphs G_0, G_1 and G_2 such that G_0 is a co-bipartite CA graph with $\operatorname{box}(G_0) \leq \operatorname{box}(G)$, G_1 and G_2 are interval graphs, and $G = G_0 \cap G_1 \cap G_2$. It will be clear from our algorithm that G need not necessarily be a normal CA graph for our method to work. The only property needed for our algorithm is the existence of two

[2]Though the statement of this theorem appears differently in its source, the proof given in its source precisely proves the theorem in the way we have stated it.

points p and q on the circle C, such that no arc in $\mathcal{A}$ passes through both p and q.

In our proof, we make use of the fact that introducing universal vertices does not affect the boxicity of a graph and if the graph was originally a co-bipartite CA graph, it remains so after the modification.

Definition 2.4. Let $G'(V', E')$ be a graph obtained by introducing some universal vertices into a graph $G(V, E)$. That is, $V' \supseteq V$ and $E' = E \cup \{(a, b) \mid a \in V' \setminus V \text{ and } b \in V', a \neq b\}$. Then we call G' to be an extension of G on V'.

Lemma 2.5. *Let $G'(V', E')$ be an extension of $G(V, E)$ on V'. If G has a known box representation of dimension k, then in $O(|V'| \cdot k)$ time we can compute a box representation of G' of dimension k. Moreover, if G is a co-bipartite CA graph, so is G'.*

Proof. Let $\mathcal{B} = \{I_1, I_2, \cdots, I_k\}$ be a known box representation of G. For $1 \leq i \leq k$, let $l_i = \min\{l_u(I_i) \mid u \in V\}$ and $r_i = \max\{r_u(I_i) \mid u \in V\}$. For $1 \leq i \leq k$, define I'_i by assigning the interval $[l_i, r_i]$, $\forall u \in V' \setminus V$ and intervals $[l_u(I_i), r_u(I_i)]$, $\forall u \in V$. By the definition of these intervals, in each I'_i, vertices in $V' \setminus V$ are universal and the adjacencies among vertices in V remains as it was in I_i. Since $\mathcal{B}$ is a box representation of G, it can be easily verified that $G' = I'_1 \cap I'_2 \cap \cdots \cap I'_k$ and hence $\mathcal{B}' = \{I'_1, I'_2, \cdots, I'_k\}$ is a valid box representation of G'. The computation of $\mathcal{B}'$ can be done in $O(|V'| \cdot k)$ time.

If G was originally co-bipartite with $V = A \uplus B$, and A and B are cliques, then G' remains co-bipartite with the vertex set V' partitioned into cliques $A' = A \cup (V' \setminus V)$ and $B' = B$. Suppose G was a CA graph with a CA model $M(C, \mathcal{A})$. Then, if we assign the circular arcs corresponding to the vertices in $V' \setminus V$ to be the entire circle C, it gives a CA model of G'. It is also possible to locally modify these universal arcs in the next step, so that they have two distinct endpoints, which are different from the end points of every other arc. Hence, if G was a co-bipartite CA graph, G' remains so. □

Theorem 2.6. *Let $G(V, E)$ be a normal CA graph and let $M(C, \mathcal{A})$ be a normal CA model of G given. The boxicity of G can be approximated within an additive error of two in $O(mn + n^2)$ time and a box representation of G of dimension at most* $\text{box}(G) + 2$ *can be computed in $O(mn + kn^2)$ time, where $m = |E(G)|$, $n = |V(G)|$ and $k =$* $\text{box}(G)$.

Proof. Let p be any arbitrary point on C. Let p_1 be the farthest clockwise end point of any arc passing through p and p_2 be the farthest anticlockwise end point of any arc passing through p. Since $M(C, \mathcal{A})$ is a normal CA model, there is no pair of arcs in $\mathcal{A}$ passing through p such that their union covers the entire circle C. If we choose q to be any point on the arc $[p_1, p_2]$ with $q \neq p_1$,

p_2, it is easy to see that no arc in $\mathcal{A}$ will pass through both p and q. Let A be the clique in G corresponding to the arcs in $\mathcal{A}$ passing through the point p and B be the clique in G corresponding to the arcs in $\mathcal{A}$ passing through the point q. By our observation that no arc in $\mathcal{A}$ passes through both p and q, we get $A \cap B = \emptyset$.

Since A and B are cliques, $G[A \cup B]$, which is the induced subgraph of the CA graph G on the vertex set $A \cup B$, is a co-bipartite CA graph. Let $G_0(V, E_0)$ be the extension of $G[A \cup B]$ on V. The graph G_0 is a supergraph of G, because G_0 is the extension of an induced subgraph of G on $V(G)$. By Lemma 2.5, G_0 is also a co-bipartite CA graph and $\text{box}(G_0) \le \text{box}(G[A \cup B])$. Using the method described in Section 2.3, we can compute an optimal box representation $\mathcal{B}_0$ of G_0 in polynomial time. Since $G[A \cup B]$ is an induced subgraph of G, we have $\text{box}(G[A \cup B]) \le \text{box}(G)$ and hence, $|\mathcal{B}_0| \le \text{box}(G[A \cup B]) \le \text{box}(G)$.

Let A and B be the cliques corresponding to the points p and q respectively, as defined earlier. We define $G_1(V, E_1)$ to be the extension of the induced subgraph $G[V \setminus A]$ on V. Similarly, define $G_2(V, E_2)$ to be the extension of the induced subgraph $G[V \setminus B]$ on V.

We claim that G_1 and G_2 are interval graphs. Imagine the process of removing all the arcs in $\mathcal{A}$ passing through p, and then cutting the circle C at the point p. Then, imagine that we are opening up the cut cycle and stretching it into a straight line, along with the arcs in $\mathcal{A}$ which have not been removed. It is easy to see that this procedure gives an interval representation. Notice that, the adjacencies among the unremoved arcs in $\mathcal{A}$ and the adjacencies among corresponding intervals in the interval graph obtained after the stretching, are the same. Since A corresponds to the set of arcs removed, this implies that the interval graph obtained is the same as $G[V \setminus A]$. The graph G_1 is a supergraph of G, since it is the extension of the induced subgraph $G[V \setminus A]$ on V. Since $G[V \setminus A]$ is an interval graph, by Lemma 2.5 G_1 is also an interval graph. An interval representation of $G[V \setminus A]$ is computable in linear time and by Lemma 2.5 this can be extended to an interval representation of G_1 in $O(n)$ time. In a similar way, in linear time we can compute an interval representation of G_2 also.

Now, we will show that $G = G_0 \cap G_1 \cap G_2$. We already saw that G_0, G_1 and G_2 are supergraphs of G. Therefore, it is enough to prove that, if $(u, v) \notin E$, then $(u, v) \notin E_0 \cap E_1 \cap E_2$. Consider $(u, v) \notin E$. **Case (i)** If $u, v \in V \setminus A$, by construction of G_1, $(u, v) \notin E_1$. **Case (ii)** If $u, v \in V \setminus B$, by construction of G_2, $(u, v) \notin E_2$. Remember that A and B are cliques. Therefore, if both (i) and (ii) are false, then one of $\{u, v\}$ is in A and the other is in B, but (u, v) is not an edge in $G[A \cup B]$. In this case, since G_0 is the extension of $G[A \cup B]$ on V, $(u, v) \notin E_0$. Thus, $G = G_0 \cap G_1 \cap G_2$.

Since $\mathcal{B}_0$ is a box representation of G_0, $(u, v) \notin E_0$ if and only if $(u, v) \notin E(I)$ for some $I \in \mathcal{B}_0$. We saw that the computation of $\mathcal{B}_0$ can be done in polynomial time, where $|\mathcal{B}_0| \leq \text{box}(G)$. We also saw that G_1, G_2 are interval graphs and their interval representations are computable in linear time. Therefore, it immediately follows that $\mathcal{B} = \mathcal{B}_0 \cup \{G_1\} \cup \{G_2\}$ is a valid box representation of G of dimension at most $\text{box}(G)+2$, computable in polynomial time.

Using theorems 2.18 and 2.22, we will later show that $\text{box}(G_0)$ can be computed in $O(\xi n + n^2)$ time and an optimal box representation $\mathcal{B}_0$ of G_0 can be computed in $O(\xi n + k_0 n^2)$ time, where $n = |V(G_0)| = |V(G)|$, $k_0 = \text{box}(G_0) \leq \text{box}(G) = k$ and ξ is a quantity which is at most the number of edges between A and $V \setminus A$ in G_0. From our definition of G_0, we have $\xi \leq m$. Therefore, the time required for computing $\text{box}(G_0)$ and $\mathcal{B}_0$ are respectively within $O(mn + n^2)$ and $O(mn + kn^2)$. From this, we can see that $|\mathcal{B}|$ can be computed in $O(mn + n^2)$ time and $\mathcal{B}$ can be computed in $O(mn + kn^2)$ time, since interval representations of G_1 and G_2 were computed in linear time. □

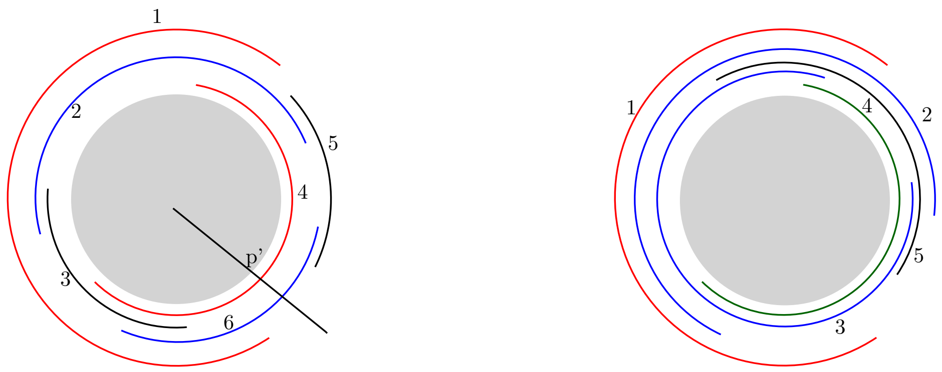
In Theorem 2.6, we assumed that an NCA model of the graph is given. This was required because recognizing NCA graphs in polynomial time is still an open problem. We can observe that though the algorithm of this section is given for normal CA graphs, it can be used for a wider class as stated below.

Theorem 2.7. *If we are given a circular arc model $M(C, \mathcal{A})$ of G with a point p' on the circle C such that the set of arcs passing through p' does not contain a pair of arcs whose union is covering the entire circle (see eg. Figure 2.2(a)), then we can approximate the boxicity of G within an additive error of two in $O(mn + n^2)$ time, where $m = |E(G)|$ and $n = |V(G)|$.*

Proof. In our proof of Theorem 2.6, instead of choosing p to be arbitrary, assign p to be the point p' (guaranteed to exist, by assumption). Such a point p' can be found in $O(n^2)$ time, if it exists. The rest of the algorithm is similar. □

Figure 2.2(a) shows a circular arc model of a graph where Theorem 2.7 is applicable and Figure 2.2(b) shows a circular arc model of a graph where Theorem 2.7 is not applicable. Though a representation, as required by Theorem 2.7, need not exist in general, it does exist for many important subclasses of CA graphs and can be constructed in polynomial time. For any proper CA graph G, the construction of a normal CA (NCA) model of G from the adjacency matrix of G, can be done in polynomial time [86, 95].

Corollary 2.8. *The boxicity of any proper circular arc graph can be approximated within an additive error of two in polynomial time.*



(a) (b)

Figure 2.2: (a) A CA model of a graph is shown, which is not an NCA model. However, the set of arcs passing through point p' of the circle does not contain a pair of arcs whose union is covering the entire circle. (b) A CA model of a graph is shown such that at any point p of the circle, the set of arcs passing through p contain a pair of arcs whose union is covering the entire circle.

2.5 Constant factor approximation algorithm for computing the boxicity of CA graphs

The algorithm of Section 2.4 can be used only when we can find a CA model $M(C, \mathcal{A})$ of G with two points p and q on the circle C, such that no arc in $\mathcal{A}$ passes through both p and q. In this section, we give an algorithm for computing a box representation of any CA graph G, of dimension at most $2\,\mathrm{box}(G)+1$, in polynomial time. From the given CA graph G, in a very natural way, we construct a co-bipartite graph G_0 such that $\mathrm{box}(G_0) \leq 2\,\mathrm{box}(G)$ and an interval graph G_1, such that $G = G_0 \cap G_1$. Using some structural properties of CA graphs, we then show that G_0 is a co-bipartite CA graph and hence, an optimal box representation $\mathcal{B}_0$ of G_0 is computable in polynomial time, using the method given in Section 2.3. Since $G = G_0 \cap G_1$, and G_1 is an interval graph, $\mathcal{B}_0 \cup \{G_1\}$ will be a box representation of G of dimension at most $2\,\mathrm{box}(G) + 1$.

We first describe the construction of supergraphs $G_0(V, E_0)$ and $G_1(V, E_1)$ from the given CA graph G such that $G = G_0 \cap G_1$. We can compute a CA model $M = (C, \mathcal{A})$ of G in linear time [72]. Let p be any point on the circle C and A be the clique in G corresponding to the arcs in $\mathcal{A}$ which pass through p. As in the proof of Theorem 2.6, $G[V \setminus A]$ is an interval graph and its interval representation can be computed in linear time. In the easy case, when $A = \emptyset$, the graph G itself is an interval graph ($\mathrm{box}(G) \leq 1$) and we can compute its optimal box representation in linear time. Therefore, we assume that this is not the case.

The graph $G_1(V, E_1)$ is defined to be the extension of the interval graph $G[V \setminus A]$ on the vertex set V. By Lemma 2.5, G_1 is an interval graph and being the extension of an induced subgraph of G on V, G_1 is a supergraph of G as well. Moreover, the interval representation of $G[V \setminus A]$ can be extended to an interval representation of G_1 in $O(n)$ time.

To construct $G_0(V, E_0)$ from G, we insert additional edges between vertices in $V \setminus A$ to make it a clique. That is, define $E_0 = E \cup \{(u,\ v) \mid u, v \in V \setminus A, u \neq v\}$. Since A was a clique in G to start with, we can see that G_0 is a co-bipartite graph. Since we have only put extra edges in its construction, G_0 is a supergraph of G.

Claim 2.8.1. *Let G_0 and G_1 be the supergraphs of G, as defined above and let $\mathcal{B}_0$ be a box representation of G_0. Then, the graph G is the intersection of graphs G_0 and G_1 (i.e. $V(G) = V(G_1) = V(G_2)$ and $E(G) = E(G_0) \cap E(G_1)$) and hence $\mathcal{B}_0 \cup \{G_1\}$ is a valid box representation of G.*

Proof. Since G_0 and G_1 are supergraphs of G, to prove that $G = G_0 \cap G_1$, it is enough to show that, if $(u,\ v) \notin E$, then $(u,\ v) \notin E_0 \cap E_1$. Consider $(u, v) \notin E$. Remember that A is a clique in G. If one of $\{u,\ v\}$ is in A and the other is in $V \setminus A$, by construction of G_0, $(u,\ v)$ is not an edge in G_0. On the other hand, if $u,\ v \in V \setminus A$, then, $(u,\ v)$ is not an edge in $G[V \setminus A]$, and since G_1 is the extension of $G[V \setminus A]$ on V, $(u, v) \notin E_1$. Thus, $G = G_0 \cap G_1$.

Since $\mathcal{B}_0$ is a box representation of G_0 and $G = G_0 \cap G_1$, where G_1 is an interval graph, it is straightforward to conclude that $\mathcal{B}_0 \cup \{G_1\}$ is a valid box representation of G. □

Claim 2.8.1 implies that if we can compute an optimal box representation of G_0, it can be used to get a box representation of G of dimension $\mathrm{box}(G_0) + 1$. However, this method will be useful in computing a near optimal box representation of G, only if $\mathrm{box}(G_0)$ is not too big compared to $\mathrm{box}(G)$. The following general lemma shows that $\mathrm{box}(G_0) \leq 2\,\mathrm{box}(G)$. This lemma is an adaptation of a similar one given in [3].

Lemma 2.9. *Let $G(V,\ E)$ be a graph with a partition (A, B) of its vertex set V with $A = \{1,\ 2,\ \cdots,\ n_1\}$ and $B = \{1',\ 2',\ \cdots,\ n_2'\}$. Let $G_0(V,\ E_0)$ be its supergraph such that $E_0 = E \cup \{(a',\ b') \mid a',\ b' \in B, a' \neq b'\}$. Then,* $\mathrm{box}(G_0) \leq 2\,\mathrm{box}(G)$ *and this bound is tight.*

Proof. Let k be the boxicity of G and $\mathcal{B} = \{I_1, I_2, \cdots, I_k\}$ be an optimal box representation of G. For each $1 \leq i \leq k$, let $l_i = \min\{l_u(I_i) \mid u \in V\}$ and $r_i = \max\{r_u(I_i) \mid u \in V\}$. Let I_{i_1} be the interval graph obtained from I_i by assigning the interval $[l_u(I_i), r_u(I_i)]$, $\forall u \in A$ and the interval $[l_i, r_{v'}(I_i)]$, $\forall v' \in B$. Let I_{i_2} be the interval graph obtained from I_i by assigning the interval $\Big[l_u(I_i),\ r_u(I_i)\Big]$, $\forall u \in A$ and the interval $\Big[l_{v'}(I_i),\ r_i\Big]$, $\forall v' \in B$.

Note that, in constructing I_{i_1} and I_{i_2} we have only extended some of the intervals of I_i and therefore, I_{i_1} and I_{i_2} are supergraphs of I and in turn of G. By construction, B induces cliques in both I_{i_1} and I_{i_2}, and thus they are supergraphs of G_0 too.

We will show that $E_0 = \bigcap_{i=1}^{k} E(I_{i_1}) \cap E(I_{i_2})$. Consider $(u, v') \notin E_0$ with $u \in A$, $v' \in B$. This implies that $(u, v') \notin E$ as well. Since $\mathcal{B}$ is a box representation of G, for some $1 \leq i \leq k$, we have $(u, v') \notin E(I_i)$. This implies that either $r_{v'}(I_i) < l_u(I_i)$ or $r_u(I_i) < l_{v'}(I_i)$. If $r_{v'}(I_i) < l_u(I_i)$, then clearly the intervals $[l_i, r_{v'}(I_i)]$ and $[l_u(I_i), r_u(I_i)]$ do not intersect and thus $(u, v') \notin E(I_{i_1})$. Similarly, if $r_u(I_i) < l_{v'}(I_i)$, then $(u, v') \notin E(I_{i_2})$. If both $u, v \in A$ and $(u, v) \notin E_0$, then also $(u, v) \notin E$. Then, $\exists i$ such that $(u, v) \notin E(I_i)$ for some $1 \leq i \leq k$ and clearly by construction, $(u, v) \notin E(I_{i_1})$ and $(u, v) \notin E(I_{i_2})$.

It follows that $G_0 = \bigcap_{i=1}^{k} I_{i_1} \cap I_{i_2}$ and therefore, $\text{box}(G_0) \leq 2\,\text{box}(G)$. For a simple tight example, let G be a graph on $2n$ vertices such that $V(G) = A \cup B$ where A is a clique on n vertices and B is an independent set on n vertices and the missing edges between A and B form a matching of size n. Trotter [93] showed that $\text{box}(G)$ is $\left\lceil \frac{n}{2} \right\rceil$. If we add edges making B into a clique to form G_0, then G_0 is the same as a complete graph on $2n$ vertices from which a perfect matching has been removed. It is well known that this graph has boxicity n [93]. In this example, when n is even, we have $\text{box}(G_0) = 2\,\text{box}(G)$. □

By Lemma 2.9, an optimal box representation $\mathcal{B}_0$ will be of dimension at most $2\,\text{box}(G)$ and by Claim 2.8.1, this can be used to derive a box representation of G of dimension at most $2\,\text{box}(G) + 1$. In the remaining parts of this section, we will show that an optimal box representation $\mathcal{B}_0$ of G_0 can indeed be computed in polynomial time, using the algorithm of Section 2.3, because G_0 is not just a co-bipartite graph but it is also a circular arc graph. For proving that G_0 is a co-bipartite CA graph, we will first prove some structural properties of CA graphs.

We use the following definition subsequently, while describing some special adjacency properties of CA graphs.

Definition 2.5 (Bi-consecutive adjacency property)**.** Let the vertex set $V(G)$ of a graph G be partitioned into two sets A and B with $|A| = n_1$ and $|B| = n_2$. A numbering scheme where vertices of A are numbered as $1, 2, \cdots, n_1$ and vertices of B are numbered as $1', 2', \cdots, n_2'$ satisfies the bi-consecutive adjacency property between A and B, if the following condition holds:
For any $i \in A$ and $j' \in B$, if i is adjacent to j', then either
(a) j' is adjacent to all k such that $1 \leq k \leq i$ or
(b) i is adjacent to all k' such that $1 \leq k' \leq j'$.

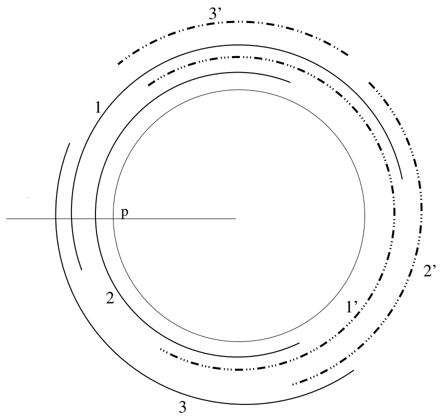


Figure 2.3: Example for numbering of vertices of a CA graph

Lemma 2.10. *Let G be a circular arc graph. Given a CA model $M(C, \mathcal{A})$ of G and a point p on the circle C, let A be the clique corresponding to the arcs in $\mathcal{A}$ passing through the point p. Then,*

1. *We can define a numbering scheme $NS(M, p)$ of vertices of G such that it satisfies the bi-consecutive adjacency property between A and $V \setminus A$.*

2. *$NS(M, p)$ can be computed in $O(n^2)$ time.*

Proof. Let A be the clique corresponding to the arcs passing through p and let $B = V \setminus A$. Let $|A| = n_1$ and $|B| = n_2$. Number the vertices in A as $1, 2, \cdots, n_1$ such that the vertex v with its $t(v)$ farthest (in the clockwise direction) from p gets number 1 and so on. Similarly, number the vertices in B as $1', 2', \cdots, n_2'$ such that the vertex v' with its $t(v')$ farthest (in the clockwise direction) from p gets number $1'$ and so on. In both cases, break ties (if any) between vertices arbitrarily, while assigning numbers. See Figure 2.3 for an illustration of the numbering scheme. Now, observe that in G, if a vertex $i \in A$ is adjacent to a vertex $j' \in B$, then at least one of the following is true: (a) the point $t(i)$ is contained in the arc $[s(j'), t(j')]$ or (b) the point $t(j')$ is contained in the arc $[s(i), t(i)]$. This implies that if $i \in A$ is adjacent to $j' \in B$, then either (a) j' is adjacent to all k such that $1 \leq k \leq i$ or (b) i is adjacent to all k' such that $1 \leq k' \leq j'$. Thus the numbering scheme defined above, satisfies bi-consecutive adjacency property between A and $B = V \setminus A$. Given the CA model $M(C, \mathcal{A})$, and a point p on C, this numbering scheme can be computed in $O(n^2)$ time. □

Claim 2.10.1. *Let $G_0(V, E_0)$ be the supergraph of $G(V, E)$ constructed at the beginning of this section. Consider the numbering scheme $NS(M, p)$ of vertices G, as obtained by Lemma 2.10. The same numbering of vertices will satisfy the bi-consecutive adjacency property between A and $V \setminus A$ in the graph G_0 as well.*

Proof. Recall our construction of the supergraph $G_0(V, E_0)$ of $G(V, E)$. For any pair of vertices $i \in A$ and $j' \in V \setminus A$, $(u, v') \in E$ if and only if $(u, v') \in E_0$. Since the numbering scheme $NS(M, p)$ of vertices of G satisfies bi-consecutive adjacency property between A and $V \setminus A$ by Lemma 2.10, and the edges across A and $V \setminus A$ are the same in both G and G_0, the same numbering of vertices will satisfy the bi-consecutive adjacency property between A and $V \setminus A$ in G_0 as well. □

Recall that G_0 is constructed to be a co-bipartite graph, where A and $V \setminus A$ are cliques. The following lemma explains how bi-consecutive adjacency property between A and $V \setminus A$ gives G_0 the additional structure of being a circular arc graph.

Lemma 2.11. *Let G be a co-bipartite graph with a partitioning of vertex set into cliques A and $B = V \setminus A$ with $|A| = n_1$ and $|B| = n_2$. Suppose there exist a numbering scheme of vertices of G which satisfies the bi-consecutive adjacency property between A and B. Then G is a CA graph.*

Proof. The proof is by construction of a CA model $M(C, \mathcal{A})$ for G.
Step 1: Choose four distinct points a, b, c, d in the clockwise order on C. Initially fix $s(i) = a$ for all $i \in A$ and $s(j') = c$ for all $j' \in B$. Choose n_1 distinct points $p_{n_1}, p_{n_1-1}, \cdots, p_1$ in the clockwise order on the arc (a, b) and set $t(i) = p_i$ for all $i \in A$. Choose n_2 distinct points $p_{n_2'}, p_{n_2-1'}, \cdots, p_{1'}$ in the clockwise order on the arc (c, d) and set $t(j') = p_{j'}$ for all $j' \in B$. As of now, the family of arcs that we have constructed represents two disjoint cliques corresponding to A and B.
Step 2: Now we will modify the start points of each arc as follows: Consider vertex $i \in A$. If $j' \in B$ is the highest numbered vertex in B such that i is adjacent to all k' with $1' \leq k' \leq j'$, then set $s(i) = t(j') = p_{j'}$. Similarly, Consider vertex $j' \in B$. If $i \in A$ is the highest numbered vertex in A such that j' is adjacent to all k with $1 \leq k \leq i$, then set $s(j') = t(i) = p_i$. Notice that we are not making any adjacencies not present in G between vertices of A and B in this step.

Since A and B are cliques, what remains to prove is that if a vertex $i \in A$ is adjacent to a vertex $j' \in B$, their corresponding arcs overlap. Consider such an edge (i, j'). If j' is adjacent to all k such that $1 \leq k \leq i$, we would have extended $s(j')$ to meet $t(i)$ in Step 2 above. If this does not occur, then by assumed bi-consecutive adjacency property, i is adjacent to all k' such that $1 \leq k' \leq j'$. In this case, we would have extended $s(i)$ to meet $t(j')$ in Step 2. In both cases, the arcs corresponding to vertices i and j' overlap. We got a CA model of G proving that G is a CA graph. □

Remark 2.2. *A different presentation of Lemma 2.11 and an independent proof was obtained by Shrestha et al. [84], while studying a class of graphs called*

2-directional orthogonal ray graph (2DORG). Shrestha et al. [84] showed that a bipartite graph G is a 2DORG if and only if its complement $\overline{G}$ is a co-bipartite CA graph. They also showed that a bipartite graph G is a 2DORG if and only if G satisfies a certain property called weakly orderability. It is easy to see that the notions of weakly orderability of G and Bi-Consecutive Adjacency Property of $\overline{G}$ coincide, giving an alternative proof of Lemma 2.11.

By Claim 2.10.1, a numbering scheme of vertices of the co-bipartite graph G_0 is computable in $O(n^2)$ time such that it satisfies the bi-consecutive adjacency property between cliques A and $V \setminus A$ in G_0. By Lemma 2.11, this implies that G_0 is a co-bipartite CA graph. Hence, using the algorithm of Section 2.3, we can compute an optimal box representation $\mathcal{B}_0$ in polynomial time. By Lemma 2.9, $|\mathcal{B}_0| \leq 2\,\mathrm{box}(G)$. Since $G = G_0 \cap G_1$, by Claim 2.8.1, $\mathcal{B} = \mathcal{B}_0 \cup \{G_1\}$ is a valid box representation of G of dimension $|\mathcal{B}_0| + 1 \leq 2\,\mathrm{box}(G) + 1$. We already saw that we can compute G_1 and its interval representation in linear time. Thus, $\mathcal{B}$ is a box representation of G of dimension at most $2\,\mathrm{box}(G) + 1$ and it is computable in polynomial time.

As in the proof of Theorem 2.6, using Theorems 2.18 and 2.22 which will be proved later, we can compute $\mathrm{box}(G_0)$ in $O(\xi n + n^2)$ time and an optimal box representation $\mathcal{B}_0$ of G_0 can be computed in $O(\xi n + k_0 n^2)$ time, where $n = |V(G_0)| = |V(G)|$, $k_0 = \mathrm{box}(G_0) \leq \mathrm{box}(G) = k$ and ξ is a quantity which is at most the number of edges between A and $V \setminus A$ in G_0. From our definition of G_0, in this case also we have $\xi \leq m$. Therefore, the time required for computing $\mathrm{box}(G_0)$ and $\mathcal{B}_0$ are respectively within $O(mn + n^2)$ and $O(mn + kn^2)$. From this, we can see that $|\mathcal{B}|$ can be computed in $O(mn + n^2)$ time and $\mathcal{B}$ can be computed in $O(mn + kn^2)$ time, since the interval representation of G_1 was computed in linear time. Thus, we have the following theorem.

Theorem 2.12. *Let G be a CA graph. A $\left(2 + \frac{1}{k}\right)$-factor approximation for $\mathrm{box}(G)$ can be computed in $O(mn + n^2)$ time and a box representation of G of dimension at most $2\,\mathrm{box}(G) + 1$ can be computed in $O(mn + kn^2)$ time, where $m = |E(G)|$, $n = |V(G)|$ and $k = \mathrm{box}(G)$.*

2.6 Complexity of computing the boxicity and optimal box representation of co-bipartite CA graphs

In Section 2.3, we gave a polynomial time algorithm to compute an optimal box representation of a co-bipartite CA graph. In this section, we will analyze the time complexity of this algorithm and using some structural properties, show how this method can be made more efficient. First, let us do a preliminary analysis of our algorithm of Section 2.3.

Let $G(V,\, E)$ be a co-bipartite CA graph with $|E| = m$ and $|V| = n$. Let $H = \overline{G}$. Recall that by Theorem 2.4, $\mathrm{box}(G) = \chi(H^*)$. Let $C_1,\, C_2,\, \cdots,\, C_k$ be the color classes in an optimal coloring of H^*. For $1 \le i \le k$, let C'_i be a maximal independent set containing C_i and $E_i = \{e \in E(H) \mid \Gamma_e \in C'_i\}$. By Theorem 2.4, $\{G_i = \overline{H_i} \mid H_i = (V, E_i),\, 1 \le i \le k\}$ gives an optimal box representation of G. Our aim is to reduce the complexity of computing an optimal proper coloring of H^*, which is a crucial step in our algorithm. We also require an efficient method to extend the color classes of H^* to maximal independent sets.

By Theorem 2.2, H^* is a perfect graph. Let t be the number edges of H or equivalently, the number of vertices in H^*. Using the standard perfect graph coloring methods, $\chi(H^*)$ can be computed, as done in [2]. However, this method takes $O(t^3)$ time, which could be as bad as $O(n^6)$ in the worst case, where n is the number of vertices of G. In [2], for the restricted case when H is an interval bigraph, they succeeded in reducing the complexity to $O(tn)$, using the zero partitioning property of the adjacency matrix of interval bigraphs. Unfortunately, since the zero partitioning property is the defining property of interval bigraphs, we cannot use the method used in [2] in our case, because the complements of CA co-bipartite graphs form a strict superclass of interval bigraphs [84]. Hence to bring down the complexity of the algorithm from $O(t^3)$, we have to go for a new method.

2.6.1 An $O(n^4)$ time algorithm for computing $\chi(H^*)$

Our method proceeds by computing a numbering of the vertices of G such that bi-consecutive adjacency property is satisfied between the clique partitions of G. This numbering scheme is then used to prove that H^* is a comparability graph and hence time required for computing an optimal proper coloring of H^* can be brought down to $O(t^2) = O(n^4)$. Later, we will see that the same numbering scheme can be used to reduce the time complexity of our algorithm further.

The following property holds for any co-bipartite CA graph.

Lemma 2.13. *If $G(V,\, E)$ is a co-bipartite CA graph, then we can find a partition $A \cup B$ of V where A and B induce cliques, having a numbering scheme of the vertices of A and B such that it satisfies bi-consecutive adjacency property between A and B. Moreover, the numbering scheme can be computed in $O(n^2)$ time.*

Proof. Let G be a co-bipartite CA graph. Recall that a circular arc model of G is constructible in linear time [72]. In any circular arc model $M(C,\, \mathcal{A})$ of a co-bipartite CA graph G, there are two points p_1 and p_2 on the circle C such that every arc passes through at least one of them [95, 66]. It is easy to see that these points can be identified in $O(n^2)$ time. Let the clique corresponding to p_1 be denoted as A. Let $B = V \setminus A$, which is clearly a clique, since the arcs corresponding to all vertices in B pass through p_2. Let $|A| = n_1$ and $|B| = n_2$. Then, by Lemma 2.10, we can compute a numbering scheme $NS(M, p_1)$ in $O(n^2)$ time, such that the vertices of A are numbered $1, 2, \cdots, n_1$ and vertices of B are numbered $1', 2', \cdots, n_2'$ and it satisfies bi-consecutive adjacency property between A and B. $\square$

In order to show that H^* is a comparability graph, we define a binary relation on $V(H^*)$.

Definition 2.6. Let $A \cup B$ be a partitioning of the vertex set $V(G)$ as described in Lemma 2.13, where A and B are cliques in G and $A = \{1, 2, \cdots, n_1\}$ and $B = \{1', 2', \cdots, n_2'\}$ is the associated numbering of vertices. We define a relation $\prec$ on $E(H)$ as: $ab' \prec cd'$ if and only if $a, c \in A$, $b', d' \in B$ with $a < c$ and $b' < d'$ and $\{a, b', c, d'\}$ induces a $2K_2$ (i.e. a matching containing two edges) in H. Correspondingly, we also define a relation $\prec^*$ on $V(H^*)$ as: $\Gamma_{ab'} \prec^* \Gamma_{cd'}$ if and only if $ab' \prec cd'$.

From the definition of H^* and the definition of $\prec^*$, it follows that if $\Gamma_{ab'} \prec^* \Gamma_{cd'}$, then $\Gamma_{ab'}$ and $\Gamma_{cd'}$ are adjacent vertices in H^*. We claim that the converse is also true.

Claim 2.13.1. *If vertices $\Gamma_{ab'}$ and $\Gamma_{cd'}$ are adjacent in H^*, then they are comparable with respect to the relation $\prec^*$.*

Proof. Let $\Gamma_{ab'}$ and $\Gamma_{cd'}$ be two adjacent vertices of H^* corresponding to the edges ab' and cd' of H where $a, c \in A$, $b', d' \in B$. From the definition of H^*, it follows that $\{a, b', c, d'\}$ induces a $2K_2$ in H. Equivalently, these vertices induce a 4-cycle in G with edges ac, cb', $b'd'$ and $d'a$. We have either $a < c$ or $c < a$.

We claim that $a < c$ if and only if $b' < d'$. To see this, assume that $a < c$. Since $cb' \in E(G)$, by the Bi-Consecutive property of the numbering scheme (Lemma 2.10), if $d' < b'$, $cd' \in E(G)$ or $ab' \in E(G)$, a contradiction. Hence, $b' < d'$. From this, it follows that if $a < c$, then $ab' \prec cd'$ and therefore, $\Gamma_{ab'} \prec^* \Gamma_{cd'}$. Using similar arguments, we can show that if $c < a$, then $\Gamma_{cd'} \prec^* \Gamma_{ab'}$. $\square$

Claim 2.13.2. *The binary relation $\prec^*$ on $V(H^*)$ is antisymmetric and transitive.*

Proof. It is clear from Definition 2.6 that the relations $\prec$ and $\prec^*$ are antisymmetric.

To show that $\prec^*$ is transitive, let $\Gamma_{ab'} \prec^* \Gamma_{cd'}$ and $\Gamma_{cd'} \prec^* \Gamma_{ef'}$. From the definition of $\prec^*$, the vertex set $\{a, b', c, d'\}$ induces a $2K_2$ in H with edges ab' and cd'. Equivalently the vertex set $\{a, b', c, d'\}$ induces 4-cycle in G with edges ac, cb', $b'd'$ and $d'a$. Similarly, the vertex set $\{c, d', e, f'\}$ induces a 4-cycle in G with edges ce, ed', $d'f'$ and $f'c$. We also have $a < c < e$ and $b' < d' < f'$, by the definition of the relation $\prec^*$. By the Bi-Consecutive property of the numbering scheme (Lemma 2.10), $cf' \in E(G)$ and $cd' \notin E(G)$ implies that $af' \in E(G)$. Similarly, $ed' \in E(G)$ and $cd' \notin E(G)$ implies that $eb' \in E(G)$. Edges ae and $b'f'$ are parts of cliques A and B. Hence, we have an induced 4-cycle in G with edges ae, eb', $b'f'$ and $f'a$. We can conclude that $ab' \prec ef'$ which implies $\Gamma_{ab'} \prec^* \Gamma_{ef'}$. Thus the relation $\prec^*$ is transitive. □

A transitive orientation of edges of H^*. Consider any pair of adjacent vertices $\Gamma_{ab'}$ and $\Gamma_{cd'}$ of H^*. From Claim 2.13.1, we know that $\Gamma_{ab'}$ and $\Gamma_{cd'}$ are comparable with respect to $\prec^*$ and by Claim 2.13.2, we know that $\prec^*$ is antisymmetric. Based on the relation $\prec^*$, we can associate an orientation for the edge in H^* between the vertices $\Gamma_{ab'}$ and $\Gamma_{cd'}$ as follows: If $\Gamma_{ab'} \prec^* \Gamma_{cd'}$, orient the edge from $\Gamma_{ab'}$ to $\Gamma_{cd'}$; on the other hand if $\Gamma_{cd'} \prec^* \Gamma_{ab'}$, orient the edge from $\Gamma_{cd'}$ to $\Gamma_{ab'}$.

It follows from Claim 2.13.2 that if each edge of H^* is oriented in this manner, we get a transitive orientation of H^*. The following lemma is a direct consequence of this fact and is a generalization of similar results obtained in [2, 99] for smaller graph classes.

Lemma 2.14. *If the complement of graph H is a co-bipartite CA graph, then H^* is a comparability graph.*

Since the number of edges in H^* may be of $O(t^2)$, where $t = |E(H)|$, using the standard algorithm for the vertex coloring of comparability graphs, we can compute an optimal proper coloring of H^* in $O(t^2) = O(n^4)$ time. Since G was any arbitrary co-bipartite CA graph to start with, we can make the following inference:

Lemma 2.15. *Minimum vertex coloring is polynomial time solvable for any graph that is the square of the line graph of the complement of a co-bipartite CA graph.*

2.6.2 An improved algorithm for computing an optimal proper coloring of H^*

Let t denote the number of edges of H, as earlier. Let $m_{AB} = n_1n_2 - t$, the number of edges between A and B in G. We call ab' a *non-edge* of G, if it is an edge of H. In this section, we utilize the structure of G along with the relation $\prec$ on the set of non-edges of G, and compute the boxicity of G in $O(\xi n + n^2)$ time, where ξ is $\min(m_{AB}, t)$. The improved running time is obtained by a suitable implementation of the greedy algorithm for the vertex coloring of comparability graphs, fine tuned for this special case, and its careful amortized analysis. Due to the structural differences with interval bigraphs as explained before, this turns out to be much different from the method used in [2].

A greedy algorithm for optimally coloring comparability graphs. There is a well-known greedy algorithm to compute an optimal coloring of comparability graphs (for reference, see e.g. [68]). We make a note of some relevant points of that algorithm here. A topological ordering $<$ of a directed graph is a linear order of its vertices such that if an edge uv is oriented from u to v, then $u < v$. If the graph is a comparability graph, then the associated transitive orientation is always acyclic and a topological ordering respecting the transitive orientation always exists. The greedy algorithm for coloring the comparability graph with colors $1, 2, \ldots,$ is the following: Consider the vertices of the comparability graph in a topological order that respects its transitive orientation. Color the first vertex in the order with color 1 and at each vertex v, color v with the minimum color not used by any neighbor of v that is already colored before coloring v. This algorithm produces an optimum coloring of the comparability graph.

A topological ordering that respects the transitive orientation of H^*. In Section 2.6.1, we saw that for any pair of vertices $\Gamma_{ab'}$ and $\Gamma_{cd'}$ of H^*, $\Gamma_{ab'}$ and $\Gamma_{cd'}$ are comparable with respect to the relation $\prec^*$ if and only if they are adjacent in H^*. From the transitive orientation given to the edges of H^* and the definition of $\prec^*$, it is straightforward to see that any linear extension of $\prec^*$ is a topological ordering that respects the transitive orientation of H^* and any linear ordering of vertices of H^* that ensures that whenever $a < c$, the vertex $\Gamma_{ab'}$ appears in the linear order before the vertex $\Gamma_{cd'}$ will serve as a linear extension of $\prec^*$. This leads to the following observation, using the description in the paragraph above.

Observation 2.1. *In order to get an optimal coloring of H^*, it is enough to greedily color the vertices of H^* according to a linear order such that whenever $a < c$, the vertex $\Gamma_{ab'}$ appears in this linear order before the vertex $\Gamma_{cd'}$.*

We use the method suggested above, for producing an optimal coloring of H^*. According to the greedy coloring strategy, while coloring a vertex $\Gamma_{xy'}$ of H^*, we need to use the minimum color not used by any neighbor of $\Gamma_{xy'}$ that is already colored before coloring $\Gamma_{xy'}$. However, while coloring $\Gamma_{xy'}$, its neighbor $\Gamma_{ab'}$ is already colored if and only if $\Gamma_{ab'} \prec^* \Gamma_{xy'}$. Moreover, we also know that if $\Gamma_{ab'} \prec^* \Gamma_{xy'}$, then $\Gamma_{ab'}$ and $\Gamma_{xy'}$ are adjacent in H^*. Therefore, while coloring a vertex $\Gamma_{xy'}$, the set of already colored neighbors of a vertex $\Gamma_{xy'}$ is $\{\Gamma_{ab'} \in V(H^*) \mid \Gamma_{ab'} \prec^* \Gamma_{xy'}\}$. Therefore, for coloring each vertex $\Gamma_{xy'}$ of H^*, the color used for greedy coloring is given by $1 + \max\{Color(\Gamma_{ab'}) \mid \Gamma_{ab'} \prec^* \Gamma_{xy'}\}$, where the maximum taken over the empty set is assumed to be zero. Our task is to implement this coloring efficiently.

A note to the reader. If the reader is not interested to know the details of the implementation of the algorithm, (s)he may take a note of Theorem 2.18 and Theorem 2.22 and skip forward directly to Section 2.7.

Re-formulating the coloring of H^* in terms of coloring of non-edges of G. Since $V(H^*) = \{\Gamma_e \mid e \in E(H)\}$, a coloring of vertices of H^* can be thought of as an equivalent coloring of non-edges of G. Note that, by the definition of H^*, a proper coloring of the vertices of H^* is equivalent to a coloring of the edges of H (i.e. non-edges of G) such that no two edges get the same color if their end points induce a $2K_2$ in H or equivalently a 4 cycle in G. Since it makes our presentation easier, we describe the vertex coloring algorithm of H^* in terms of its equivalent coloring of non-edges of G.

Some basic data structures. Recall that $A \cup B$ is a partitioning of $V(G)$ where A and B induce cliques, with $A = \{1, 2, \ldots, n_1\}$ and $B = \{1', 2', \ldots, n_2'\}$ such that the numbering satisfies the bi-consecutive adjacency property between A and B. The following definitions are with respect to G. For $X \subseteq V$, let $N_X(v)$ represent the set of neighbors of v in X and $\widehat{N}_X(v) = X \setminus N_X(v)$. For $S \subseteq V$, $N_X(S) = \bigcup_{v \in S} N_X(v)$ and $\widehat{N}_X(S) = \bigcup_{v \in S} \widehat{N}_X(v)$. Let $deg_X(v)$ denote $|N_X(v)|$. The linked lists corresponding to $N_B(v)$ and $\widehat{N}_B(v)$ for each $v \in A$ and $N_A(v')$ and $\widehat{N}_A(v')$ for each $v' \in B$, with their entries sorted with respect to the numbering scheme described above, can be constructed from the adjacency list of G. This can be done in overall $O(n^2)$ time. We will assume that lists N_A, $\widehat{N}_A$, N_B, $\widehat{N}_B$ are global data structures. In the remaining parts of this section, we assume that the maximum taken over an empty set is zero.

The coloring algorithm. Assume that the colors available are $1, 2, \cdots$. Notice that, if xy' and xz' are two non-edges of G incident at a vertex $x \in A$, then xy' and xz' are mutually incomparable under $\prec$ and the relative order of coloring them is immaterial for the coloring produced using the method suggested by Observation 2.1. Therefore, to implement this method, it is safe to split the coloring algorithm into $|A| = n_1$ stages such that for each $1 \leq i \leq n_1 - 1$, stage i is followed by stage $i + 1$ and for $1 \leq i \leq n_1$, all the non-edges xy' of G with $x = i$ are colored in stage i.

Our coloring algorithm considers $x = 1, 2, \cdots, n_1$ in that order and invokes Algorithm 1 in order to color the non-edges of G incident at $x \in A$ using the color suggested by the greedy strategy. For the convenience of our analysis, we refer to an invocation of Algorithm 1 for vertex x as *the processing of x*. The non-edges incident at x are colored only during the processing of x and once the processing of x is finished they never get recolored.

Before getting into the finer details of Algorithm 1, we will try to understand the objective of Algorithm 1 a bit closely. Let xy' be a non-edge in G incident at x. Consider a non-edge tu' such that $tu' \prec xy'$. By the definition of $\prec$, we have $t < x$ and therefore, the processing of vertex t is finished before we started processing x. Therefore, we have the following observation.

Observation 2.2. *Let xy' be a non-edge in G incident at x. When the processing of x is about to begin, all non-edges tu' of G such that $tu' \prec xy'$ are already colored and they will not be recolored in future.*

By Observation 2.1 and Observation 2.2, to produce an optimal coloring it suffices to ensure that when the processing of x finishes, the non-edge xy' is assigned the color suggested by the greedy strategy. Consider a non-edge xy' of G. Let $F_{xy'} = F_{y'} = \{ab' \in E(H) \mid ab' \prec xy'\}$ and $maxcolor(F_{y'}) = \max\{Color(ab') \mid ab' \in F_{y'}\}$. From our discussions so far, we know that the color suggested by the greedy strategy for the non-edge xy' is $maxcolor(F_{y'})+1$.

Therefore, our task involved in the processing of x reduces to efficiently compute $maxcolor(F_{y'}) + 1$, for all non-edges xy' incident at x, using Algorithm 1. To understand how Algorithm 1 does this, let us first look at the set $F_{y'}$ a bit more closely. Let $P = \{a \in N_A(\widehat{N}_B(x)) \mid a < x\}$, i.e., P is a subset of A consisting of the vertices that are smaller than x and are neighbors of some non-neighbor of x in B. Let $Q = \{b' \in N_B(x) \mid b' < \min \widehat{N}_B(x)\}$, i.e., Q is a subset of B consisting of the neighbors of x whose number is smaller than the minimum numbered non-neighbor of x in B.

Claim 2.15.1. $F_{y'} = \biguplus_{a \in N_A(y') \cap P}\{ab' \in E(H) \mid b' \in Q\} = \{ab' \in E(H) \mid a \in N_A(y') \cap P$ *and* $b' \in Q\}$.

Proof. Since $F_{y'} = \{ab' \in E(H) \mid ab' \prec xy'\}$, we need to show that for any $ab' \in E(H)$, $ab' \prec xy'$ if and only if $a \in N_A(y') \cap P$ and $b' \in Q$. Recall that $ab' \prec xy'$ if and only if $a < x$, $b' < y'$ and $\{a, b', x, y'\}$ induces a 4-cycle in G. Observe that $N_A(y') \cap P = \{a \in N_A(y') \mid a < x\}$. It is easy to see that, if $ab' \in E(H)$ with $a \in N_A(y') \cap P$ and $b' \in Q$, then $ab' \prec xy'$.

To prove the other direction, assume that $ab' \prec xy'$. Then we have $a \in N_A(y')$, $a < x$ and therefore, $a \in N_A(y') \cap P$. Similarly, $b' \in N_B(x)$, $b' < y'$. Since b' is a neighbor of x, we know that $\min \widehat{N}_B(x) \neq b'$. Suppose $\min \widehat{N}_B(x) < b'$. Since the numbering scheme satisfies bi-consecutive adjacency property, $xb' \in E(G)$ implies that either $(x, \min \widehat{N}_B(x)) \in E(G)$ or $ab' \in E(G)$, which is a contradiction. Therefore $b' < \min \widehat{N}_B(x)$ and therefore, $b' \in Q$. $\square$

By the above claim, we have

$$maxcolor(F_{y'}) = \max_{a \in N_A(y') \cap P}\{\max_{b' \in Q, ab' \in E(H)}\{Color(ab')\}\}$$

But, if we have to do this computation separately for each $y' \in \widehat{N}_B(x)$, then for any $a \in P$ which is in $N_A(y')$ of more than one y', the computation of

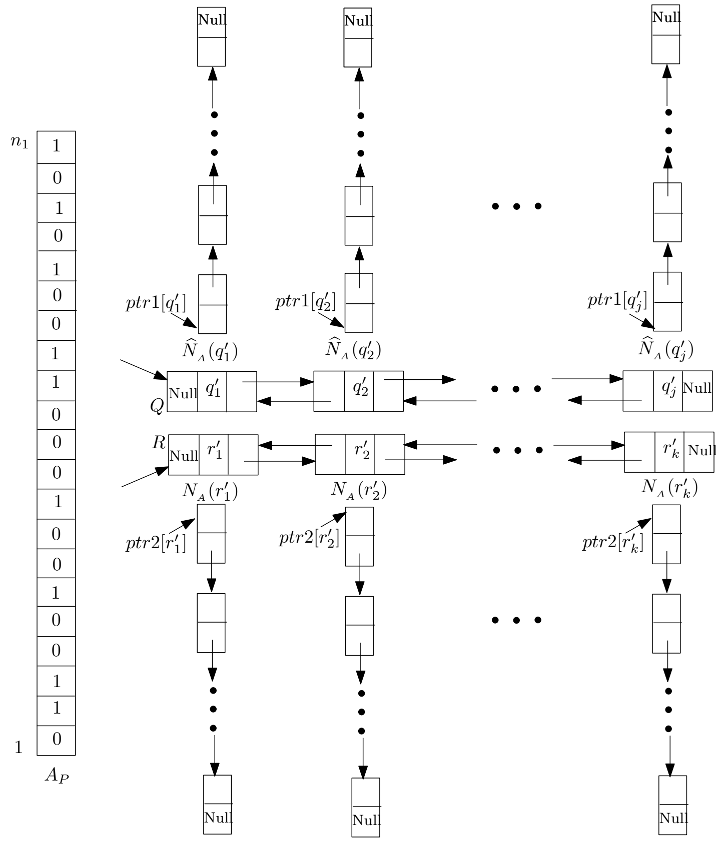


Figure 2.4: Basic data structures used in Algorithm 1 and Algorithm 2. For $1 \leq i \leq n_1$, $A_P[i] = 1$ if and only if $i \in P \subseteq A$. Sets Q and R are subsets of B and are represented as doubly linked lists. For each $q' \in Q$, the linked list $\widehat{N}_A(q')$ is sorted in the ascending order. Similarly, for each $r' \in R$, the linked list $N_A(r')$ is sorted in the ascending order.

$maxcolor(a) = \max\{Color(ab') \mid b' \in Q, ab' \in E(H)\}$ has to be repeated. To avoid this repetition, Algorithm 1 first computes $maxcolor(a)+1$ for each $a \in P$ and then uses these values while computing $maxcolor(F_{y'})$, for $y' \in \widehat{N}_B(x)$.

Here is a simple description of Algorithm 1. To make it easier to follow this description, the reader may refer to Figure 2.4.

- Type 0 work (Lines 1 to 3): Here we do some initializations. Recall the definitions of P and Q. The algorithm computes Q and $R = \widehat{N}_B(x)$ and also computes an indicator array A_P of P such that $A_P[a] = 1$ if $a \in P$ and is zero otherwise. The lists Q and R can be represented as doubly linked lists. For each $b' \in Q$, a pointer $ptr1[b']$ is initialized to point to the start of the linked list $\widehat{N}_A(b')$. For each $r' \in R$, a pointer $ptr2[r']$ is initialized to point to the start of the linked list $N_A(r')$ and $Color(xr')$ is initialized to one. For each $p \in P$, $color[p]$ is initialized to one.

- Type 1 work (Lines 6 to 12 performed for elements of P considered according to their ascending order): By these lines, for each $p \in P$ the algorithm sets $color[p] = maxcolor(p) + 1$. To achieve this, for each $q' \in Q$ such that $p \in \widehat{N}_{A}(q')$, the algorithm updates $color[p] = Color(pq') + 1$, in case $color[p] < Color(pq') + 1$.

For checking whether $p \in \widehat{N}_{A}(q')$, the algorithm traverses the sorted list $\widehat{N}_{A}(q')$ in the forward direction by repeatedly updating $ptr1[q']$ from its current position until it points to the next element which is greater than or equal to p, if such an element exists. If no such element exists in $\widehat{N}_{A}(q')$, the pointer $ptr1[q']$ reaches the end of the list $\widehat{N}_{A}(q')$. In that case, the element q' is deleted from Q to make sure that no more Type 1 work is done on $q' \in Q$ or the list $\widehat{N}_{A}(q')$ for elements of P considered in future. Note that, for each $q' \in Q$, the list $\widehat{N}_{A}(q')$ is traversed only once during one invocation of Algorithm 1.

- Type 2 work (Lines 13 to 19 performed for elements of P considered according to their ascending order): By these lines, for each $r' \in \widehat{N}_{B}(x)$ the algorithm computes $maxcolor(F_{r'}) + 1$ using the values of $color[p] = maxcolor(p) + 1$ already computed as part of Type 1 work and assigns $Color(xr') = maxcolor(F_{r'})$ +1. To achieve this, for each $r' \in R$ such that $p \in N_{A}(r')$, the algorithm updates $Color(xr') = color[p]$, in case $Color(xr') < color[p]$. Thus, each non-edge xr' incident at x gets the color which is the same as the color suggested by the greedy coloring.

  For checking whether $p \in N_{A}(r')$ the algorithm traverses the sorted list $N_{A}(r')$ by updating $ptr2[r']$. This is done in a similar way as we operated with $ptr1[q']$ for doing the Type 1 work. This ensures that for each $r' \in R$, the list $N_{A}(r')$ is traversed only once during one invocation of Algorithm 1.

Thus, by invoking Algorithm 1 for each vertex $x \in A$, according to the increasing order of the numbers assigned to vertices in A, each non-edge of G gets the color required by the greedy coloring. As explained earlier in this section, this implies the following.

Lemma 2.16. *Invoking Algorithm 1 for each vertex $x \in A$, according to the increasing order of the numbers assigned to vertices in A, gives an optimal proper coloring of vertices of H^*.*

Lemma 2.17. *Time spent over all invocations of Algorithm 1 is $O(\xi n + n^2)$.*

Proof. The algorithm is invoked once for each vertex $x \in A$. Recall that $m_{AB} = |\{ab' \in E(G) \mid a \in A \text{ and } b' \in B\}|$, $t = n_1 n_2 - m_{AB} = |E(\overline{G})|$ and $\xi = \min(m_{AB}, t)$.

Type 0 work (Lines 1 to 3): Initializations in Line 1 can be achieved in $O(\xi n + n^2)$ time as follows. A_P can be initialized to zero in $O(n)$ time during

Algorithm 1: Computing colors of non-edges incident on vertex $x \in A$

Input: $x \in A$

Output: $Color(xy')$ for each $y' \in \widehat{N}_B(x)$

/* Type 0 work : Lines 1 to 3 - Initializations */

/* Let $P = \{a \in N_A(\widehat{N}_B(x)) \mid a < x\}$ */

1 For $1 \le a \le n_1$, let $A_P[a] = 0$ initially. For each $a \in P$, set $A_P[a] = 1$ and $color[a] = 1$

2 Compute $Q = \{b' \in N_B(x) \mid b' < p'\}$, where $p' = \min(\widehat{N}_B(x))$, which is the first element of $\widehat{N}_B(x)$. For each $b' \in Q$, initialize $ptr1[b'] =$ NULL if $\widehat{N}_A(b') = \emptyset$, and $ptr1[b'] =$ start of $\widehat{N}_A(b')$ otherwise

3 Assign $R = \widehat{N}_B(x)$ and for each $r' \in R$ initialize $Color(xr') = 1$ and initialize $ptr2[r'] =$ NULL if $N_A(r') = \emptyset$ and $ptr2[r'] =$ start of $N_A(r')$ otherwise

4 **for** $cur = 1$ *to* n_1 **do**

5   **if** $A_P[cur] = 1$ **then**

    /* Type 1 work : Lines 6 to 12 - Computing $color[cur] = 1+$ the maximum color given to a non-edge between cur and Q */

6   **for** *each* q' *in* Q **do**

7   **while** $ptr1[q']$ *is not NULL and* $\widehat{N}_A(q')[ptr1[q']] < cur$ **do**

8   Increment the pointer $ptr1[q']$ /* $ptr1[q']$ becomes NULL if it is incremented past the last element in $\widehat{N}_A(q')$ */

9   **if** $ptr1[q']$ *is NULL* **then**

10   delete q' from Q

11   **else if** $\widehat{N}_A(q')[ptr1[q']] = cur$ **then**

12         $color[cur] = \max(color[cur], Color(cur\ q') + 1)$ /* non-edge $(cur\ q')$ is already colored */

    /* Type 2 work : Lines 13 to 19 - Identify non-edges at x affected by non-edges between cur and Q and update their colors if necessary */

13   **for** *each* r' *in* R **do**

14   **while** $ptr2[r']$ *is not NULL and* $N_A(r')[ptr2[r']] < cur$ **do**

15   Increment the pointer $ptr2[r']$ /* $ptr2[r']$ becomes NULL if it is incremented past the last element in $N_A(r')$ */

16   **if** $ptr2[r']$ *is NULL* **then**

17   delete r' from R

18   **else if** $N_A(r')[ptr2[r']] = cur$ **then**

19   **if** $Color(xr') < color[cur]$ **then** $Color(xr') = color[cur]$

the processing of each $x \in A$. The total time for this work is $O(n^2)$, over all invocations of Algorithm 1. Recall that $P = \{a \in N_A(\widehat{N}_B(x)) \mid a < x\}$. We are not computing the set P explicitly in Algorithm 1. The initialization of non-zero entries of A_P in Line 1 can be implemented in the following way: Checking whether $\widehat{N}_B(x)$ is empty during the processing of an $x \in A$ can be done in unit time, requiring $O(n)$ time over all invocations of Algorithm 1. If $\widehat{N}_B(x)$ is not empty, then for each $y' \in \widehat{N}_B(x)$, traverse the list $N_A(y')$ and for each $a \in N_A(y')$ set $A_P[a] = 1$, if $a < x$. We split the time required for this into two. To detect that the end of the list $\widehat{N}_B(x)$ is reached requires only unit time per $x \in A$, which amounts to $O(n)$ time over all invocations of Algorithm 1. The remaining time is spent in actually traversing the list $N_A(y')$ for each $y' \in \widehat{N}_B(x)$ and setting $A_P[a] = 1$ for each $a \in N_A(y')$ with $a < x$. For calculating the time required for this, we think from the perspective of y': Compute the time spent by y' over all the invocations of the algorithm and sum this up over all elements $y' \in B$. A vertex $y' \in B$ can account for setting $A_P[a] = 1$ for every $a \in N_A(y')$ when the list $N_A(y')$ is traversed and this happens during the processing of each $x \in A$ such that $y' \in \widehat{N}_B(x)$ or in other words during the processing of $x \in A$ such that $x \in \widehat{N}_A(y')$. Note that if $x \notin \widehat{N}_A(y')$, the vertex y' is not involved in this initialization work during the processing of x. Thus, there are $|\{x \in \widehat{N}_A(y')\}| = n_1 - deg_A(y')$ invocations of Algorithm 1 during which y' does this work and in each such invocation, y' does $|N_A(y')| = deg_A(y')$ initializations and an additional one unit of time is required to detect that the end of the list $N_A(y')$ is reached. Therefore, counting together all invocations of Algorithm 1, the total time spent by elements of B for this initialization is $\sum_{y' \in B} (deg_A(y') + 1)(n_1 - deg_A(y')) = O(n + n \min(m_{AB}, t)) = O(n + \xi n)$. During the processing of x, the initialization of the doubly linked list Q and the pointers in Line 2 can be done in $O(deg_B(x))$ time. Summing over all $x \in A$, this amounts to $O(m_{AB}) = O(m)$ time, over all invocations. For initializing the doubly linked list R and the pointers in Line 3, we need $O(n_2 - deg_B(x))$ time during the processing of x. Summed over all $x \in A$, this amounts to $O(t)$ time over all invocations of the algorithm. Adding all the above, total time spent on Type 0 work (over all invocations of Algorithm 1) is $O(n + \xi n + n^2 + m + t) = O(\xi n + n^2)$, since $m + t = O(n^2)$.

Type 1 work (Lines 6 to 12): To make it easier to follow the algorithm, Line 6 is written as a for-loop. But to implement this efficiently, we consider that a pointer is used for storing the current traversal position in the doubly linked list Q and this pointer is made to point to the next element in Q each time this step is executed.

Let us calculate the time spent in Type 1 work. Recall that $Q \subseteq N_B(x)$. Therefore, if $q' \notin N_B(x)$, then during the processing of x, there is no Type 1 work associated with q'. When $q' \in Q$, the algorithm remembers the traversal position in the linked list $\widehat{N}_A(q')$ using $ptr1[q']$. This means that $ptr1[q']$ continues from where it stopped in the current iteration, while doing the Type 1 work of the next element of P. Therefore, the pointer $ptr1[q']$ moves at most $n_1 - deg_A(q')$ times for each $q' \in Q \subseteq N_B(x)$. When $ptr1[q']$ reaches the end of list $\widehat{N}_A(q')$, the element q' is deleted from the doubly linked list Q. This makes sure that no more Type 1 work is done on q' or the list $\widehat{N}_A(q')$ during the current invocation of Algorithm 1. Thus, during the processing of each x such that $q' \in N_B(x)$, Line 7 is repeated only $O(n_1 - deg_A(q'))$ times.

There are at most $|\{x \mid q' \in N_B(x)\}| = deg_A(q')$ invocations of Algorithm 1 during which $q' \in Q$ and in each such invocation, q' can account for the execution of Line 7 for $O(n_1 - deg_A(q'))$ times. Therefore, over all invocations of Algorithm 1, the number of times Line 7 is executed is $O\left(\sum_{q' \in B}(n_1 - deg_A(q'))deg_A(q')\right) = O(n \min(m_{AB}, t)) = O(\xi n)$.

It is possible that Q is found empty when the algorithm is trying to traverse the linked list Q in Line 6. However, this can happen only $O(|P|) = O(n)$ times during an invocation of Algorithm 1. Therefore, over all invocations of Algorithm 1, the time spent in Line 6 with Q being found empty is $O(n^2)$. It is easy to see that the number of times Line 6 gets executed with Q being found non-empty is at most the number of times Line 7 is executed and Lines 9 - 12 get executed at most once for each such execution of Line 6. The deletion of q' in Line 10 can be done in unit time, since the pointer storing the current traversal position of the doubly linked list Q points to q'. Hence the total time spent for Type 1 work over all invocations of Algorithm 1 is $O(\xi n + n^2)$.

Type 2 work (Lines 13 to 19): As in the case of Type 1 work, a pointer is used for storing the current traversal position in the doubly linked list R and it is made to point to the next element in R each time Line 13 is executed. For each element $r' \in R$, the algorithm remembers the traversal position in the linked list $N_A(r')$ using $ptr2[r']$. This means that $ptr2[r']$ continues from where it stopped in the current iteration while doing the Type 2 work of the next element of P. Therefore, pointer $ptr2[r']$ moves at most $deg_A(r')$ times for each $r' \in R = \widehat{N}_B(x)$. When $ptr2[r']$ reaches the end of list $N_A(r')$, the element r' is deleted from the doubly linked list R. This makes sure that during the processing each x such that $r' \in \widehat{N}_B(x)$, Line 14 is repeated only $O(deg_A(r'))$ times. If $r' \notin \widehat{N}_B(x)$, there is no Type 2 work associated with r'. Thus, there are exactly $|\{x \mid r' \in \widehat{N}_B(x)\}| = n_1 - deg_A(r')$ invocations of Algorithm 1 such that $r' \in \widehat{N}_B(x)$ and in each such invocation, r' can account for the execution of Line 14 for $O(deg_A(r'))$ times. Therefore, over all invocations of Algorithm 1, the number of times Line 14 is executed is $O\left(\sum_{b' \in B} deg_A(b')(n_1 - deg_A(b'))\right) = O(n \min(m_{AB}, t)) = O(\xi n)$. It is possible that R is found empty when the algorithm is trying to traverse the linked list R in Line 13. However, this can happen only $O(|P|) = O(n)$ times during an invocation of Algorithm 1. Therefore, over all invocations of Algorithm 1, the time spent in Line 13 with R being found empty is $O(n^2)$. It is easy to see that the number of times Line 13 gets executed with R being found non-empty is at most the number of times Line 14 is executed and Lines 16 - 19 get executed at most once for each such execution of Line 13. The deletion of r' in Line 17 can be done in unit time, since the pointer storing the current traversal position of the doubly linked list R points to r'. Hence the total time spent for Type 2 work (over all invocations of Algorithm 1) is $O(\xi n + n^2)$.

Thus the total time spent over all invocations of Algorithm 1 is $O(\xi n + n^2)$, as claimed. $\square$

Since $\text{box}(G) = \chi(H^*)$ by Theorem 2.4, from Lemma 2.16 and Lemma 2.17, we can conclude:

Theorem 2.18. *If G is a co-bipartite circular arc graph with cliques A and $V \setminus A$, then,* $\text{box}(G)$ *can be computed in $O(\xi n + n^2)$ time, where $\xi = \min($number of edges between A and $V \setminus A$ in G, number of edges between A and $V \setminus A$ in $\overline{G})$.*

2.6.3 Expanding color classes of H^* to maximal independent sets

For $1 \leq i \leq k$, let C_i be the i^{th} color class in the optimal coloring of H^* obtained by the algorithm of Section 2.6.2 and let C'_i be a maximal independent set containing C_i. Recall from the beginning of Section 2.6, that we can compute an optimal box representation $\mathcal{B} = \{G_1, G_2, \cdots, G_k\}$ of G, by computing C'_i, for $1 \leq i \leq k$. The following lemma suggests one way to compute these maximal independent sets.

Lemma 2.19. *Let C_i be the i^{th} color class in the optimal coloring of H^* obtained by the algorithm of Section 2.6.2. Let $S_i = C_1 \cup C_2 \cup \cdots \cup C_i$ and let $MaxS_i$ be the set of maximal elements of $(S_i, \prec^*)$, i.e, $MaxS_i = \{\Gamma_{ab'} \in S_i \mid \nexists \Gamma_{cd'} \in S_i$ with $\Gamma_{ab'} \prec^* \Gamma_{cd'}\}$. Then $MaxS_i$ is a maximal independent set in H^* containing C_i.*

Proof. $MaxS_i$, being the set of maximal elements of $(S_i, \prec^*)$, forms an independent set in H^*. Recall that, as per our algorithm, for any $ab' \in E(H)$, $Color(ab') = \max\{Color(e) + 1 \mid e \in E(H)$ such that $e \prec ab'\}$. Consider $\Gamma_{ab'} \in C_i$. If $\exists cd'$ such that $ab' \prec cd'$, then $Color(cd') > Color(ab') = i$ and therefore, $\Gamma_{cd'} \notin S_i$. Hence, by the definition of $MaxS_i$, $\Gamma_{ab'} \in MaxS_i$. Thus, $C_i \subseteq MaxS_i$.

Consider any $\Gamma_{ab'} \notin MaxS_i$. Either $\Gamma_{ab'} \in (S_i \setminus MaxS_i)$ or $\Gamma_{ab'} \notin S_i$. In the former case, $\exists \Gamma_{cd'} \in MaxS_i$ with $\Gamma_{ab'} \prec^* \Gamma_{cd'}$. In the latter case, when $\Gamma_{ab'} \notin S_i$, $Color(ab') > i$ and it is easy to see from our coloring strategy that $\exists \Gamma_{cd'} \in C_i \subseteq MaxS_i$ with $\Gamma_{cd'} \prec^* \Gamma_{ab'}$. Therefore, in both cases, if $\Gamma_{ab'}$ is added to $MaxS_i$, it will no longer be an independent set. Thus, $MaxS_i$ is a maximal independent set containing C_i. $\square$

The next question is to efficiently compute $MaxS_i$, for $1 \leq i \leq k$. For this purpose, we introduce the following definition.

Definition 2.7. For each $ab' \in E(H)$, let

$$Next(ab') = \begin{cases} \min\limits_{e \in E(H), ab' \prec e} \{Color(e)\}, & \text{if } \exists e \in E(H) \text{ such that } ab' \prec e \\ k+1, & \text{otherwise} \end{cases}$$

Lemma 2.20. *For $1 \leq i \leq k$, $MaxS_i = \{\Gamma_{ab'} \in S_i \mid Next(ab') > i\}$*

Proof. If $\Gamma_{ab'} \notin MaxS_i$, then $\exists \Gamma_{cd'} \in S_i$ with $\Gamma_{ab'} \prec^* \Gamma_{cd'}$. It will follow that $Next(ab') \leq Color(cd') \leq i$. Conversely, if $Next(ab') \leq i$, then $\exists \Gamma_{cd'} \in S_i$ with $\Gamma_{ab'} \prec^* \Gamma_{cd'}$ and hence $\Gamma_{ab'} \notin MaxS_i$. $\square$

Our method is to first compute $Next(ab')$ for each $ab' \in E(H)$ and then, use Lemma 2.20 to compute $MaxS_i$, for $1 \leq i \leq k$.

Computing $Next(ab')$ for all $ab' \in E(H)$

Here we describe an algorithm to compute $Next(ab')$ for all $ab' \in E(H)$ in $O(\xi n + n^2)$ time. Let $Next(ab')$ for all $ab' \in E(H)$ be initialized to $k+1$. This can be done in $O(|E(H)|) = O(n^2)$.

Consider the following strategy. Take a non-edge $e \in E(H)$ and update $Next(ab')$ of all $ab' \prec e$ with $\min(Next(ab'), Color(e))$. When we have repeated this for all $e \in E(H)$, it is easy to see that the values of $Next(ab')$ for every $ab' \in E(H)$ will satisfy Definition 2.7.

In order to do this efficiently, we process the non-edges incident at a vertex $x \in A$ together, in an invocation of Algorithm 2 - hereafter called the processing of x. During the processing of x, each non-edge xy' incident at x updates $Next(ab')$ of all $ab' \prec xy'$ with $\min(Next(ab'), Color(xy'))$. We will process $x = 1, 2, \cdots, n_1$ in that order. The data structures used are similar to those used for Algorithm 1.

Consider an $x \in A$. As in Section 2.6.2, let $P = \{a \in N_A(\widehat{N}_B(x)) \mid a < x\}$, $Q = \{b' \in N_B(x) \mid b' < \min \widehat{N}_B(x)\}$ and $F_{y'} = \{ab' \in E(H) \mid ab' \prec xy'\}$. Let $T_x = \bigcup_{y' \in \widehat{N}_B(x)} \{ab' \in E(H) \mid ab' \prec xy'\} = \bigcup_{y' \in \widehat{N}_B(x)} F_{y'}$. Notice that, by the definition of T_x and $Next(ab')$, the value of $Next(ab')$ is dependent on the colors assigned to some non-edge incident at x, only if $ab' \in T_x$. By the claim proved in Section 2.6.2, $F_{y'} = \{ab' \in E(H) \mid a \in N_A(y') \cap P$ and $b' \in Q\}$. Hence, $T_x = \{ab' \in E(H) \mid a \in P$ and $b' \in Q\}$. Therefore, during the processing of $x \in A$, we just need to update the $Next$ values of non-edges between P and Q only.

Consider any $ab' \in T_x$. The set of non-edges incident at x whose colors can affect the value of $Next(ab')$ belong to the set $\{xy' \mid y' \in \widehat{N}_B(x)$ and $ab' \prec xy'\}$. We claim that this set is the same as $\{xy' \mid y' \in \widehat{N}_B(x) \cap N_B(a)\}$.

Since $ab' \prec xy'$ implies $y' \in \widehat{N}_B(x) \cap N_B(a)$, we have $\{xy' \mid y' \in \widehat{N}_B(x)$ and $ab' \prec xy'\} \subseteq \{xy' \mid y' \in \widehat{N}_B(x) \cap N_B(a)\}$. To prove the reverse direction of inclusion, assume that xy' is such that $y' \in \widehat{N}_B(x) \cap N_B(a)$. In the previous paragraph, we saw that $ab' \in T_x$ implies $ab' \in E(H)$ with $a \in P$ and $b \in Q$. From the definitions of P and Q and the assumption that $y' \in \widehat{N}_B(x) \cap N_B(a)$, it follows that $a < x$, $ay' \in E(G)$, $xb' \in E(G)$, $xy' \in E(H)$ and $b' < y'$. Moreover, $ab' \in E(H)$ and A and B are cliques in G. Therefore, by the definition of $\prec$, we get $ab' \prec xy'$. Thus, $\{xy' \mid y' \in \widehat{N}_B(x)$ and $ab' \prec xy'\} \supseteq \{xy' \mid y' \in \widehat{N}_B(x) \cap N_B(a)\}$.

Thus, the set of non-edges incident at x whose colors can affect the value of $Next(ab')$ belong to the set $\{xy' \mid y' \in \widehat{N}_B(x) \cap N_B(a)\}$. Notice that for any fixed $a \in P$, this set is independent of any particular $b' \in Q$. Let us denote this set by U_a. For any non-edge $ab' \in E(H)$ with $a \in P$ and $b' \in Q$, $Next(ab') \leq \min\{Color(e) \mid e \in U_a\}$. Hence, we can make the following inference, which is critical for the efficiency of Algorithm 2:

Fact. For a fixed vertex $a \in P$, for any non-edge $ab' \in E(H)$ between a and Q, we just need to update $Next(ab')$ with $\min(Next(ab'), MinColor[a])$, where $MinColor[a] = \min\{Color(e) \mid e \in U_a\}$, irrespective of which $b' \in Q$ is involved. (If $U_a = \emptyset$, we take $MinColor[a] = k + 1$.)

Here is a short description of Algorithm 2. To make it easier to follow this description, the reader may refer to Figure 2.4.

- Type 0 work (Lines 1 to 3): This is similar to Type 0 work of Algorithm 1. In these lines, the algorithm computes Q, R and the indicator array A_P of P and initializes the pointer $ptr1[b']$ for each $b' \in Q$ and the pointer $ptr2[r']$ for each $r' \in R$. The lists Q and R are represented as doubly linked lists. For each $a \in P$, $MinColor[a]$ is initialized to $k + 1$.

- Type 1 work (Lines 6 to 12 performed for elements of P considered according to their ascending order): By these lines, for each $p \in P$ the algorithm computes $MinColor[p] = \min\{Color(xy') \mid y' \in \widehat{N}_B(x) \cap N_B(p)\}$. To achieve this, for each $r' \in R = \widehat{N}_B(x)$ such that $p \in N_A(r')$, the algorithm updates $MinColor[p] = \min(MinColor[p], Color(xr'))$. For checking whether $p \in N_A(r')$ the algorithm traverses the sorted list $R = N_A(r')$ by updating $ptr2[r']$, as we did for the Type 2 work of Algorithm 1.

- Type 2 work (Lines 13 to 19 performed for elements of P considered according to their ascending order): By these lines, the algorithm updates $Next(ab')$, for each $ab' \in T_x$ with $\min(Next(ab'), MinColor[a])$. (Recall that $T_x = \{ab' \in E(H) \mid a \in P$ and $b' \in Q\}$.) To achieve this, for each $q' \in Q$ such that $p \in \widehat{N}_A(q')$, the algorithm updates $Next(pb') = \min(Next(pb'), MinColor[p])$. For checking whether $p \in \widehat{N}_A(q')$, the al-

Algorithm 2: Each non-edge xy' incident at vertex $x \in A$ updates $Next(ab')$ of all non-edges $ab' \prec xy'$

Input: $x \in A$

Output: The updated $Next(ab')$ for each non-edges $ab' \prec xy'$ where $y' \in \widehat{N}_B(x)$

`/* Type 0 work : Lines 1 to 3 - Initializations */`

`/* Let` $P = \{a \in N_A(\widehat{N}_B(x)) \mid a < x\}$ `*/`

1 For $1 \le a \le n_1$, let $A_P[a] = 0$ initially. For each $a \in P$, set $A_P[a] = 1$ and $MinColor[a] = k + 1$

2 Compute $Q = \{b' \in N_B(x) \mid b' < p'\}$, where $p' = \min(\widehat{N}_B(x))$, which is the first element of $\widehat{N}_B(x)$. For each $b' \in Q$ initialize $ptr1[b'] =$ NULL if $\widehat{N}_A(b') = \emptyset$, and $ptr1[b'] =$ start of $\widehat{N}_A(b')$ otherwise.

3 Assign $R = \widehat{N}_B(x)$ and for each $r' \in R$ initialize $ptr2[r'] =$ NULL if $N_A(r') = \emptyset$, and $ptr2[r'] =$ start of $N_A(r')$ otherwise.

4 **for** $cur = 1$ *to* n_1 **do**

5   **if** $A_P[cur] = 1$ **then**

        `/* Type 1 work : Lines 6 to 12 - Computing` $MinColor[cur] =$ `the minimum color given to a non-edge between` x `and` $N_B(cur) \cap R$ `*/`

6   **for** *each* r' *in* R **do**

7   **while** $ptr2[r']$ *is not NULL and* $N_A(r')[ptr2[r']] < cur$ **do**

8   Increment the pointer $ptr2[r']$ `/*` $ptr2[r']$ `becomes NULL if it is incremented past the last element in` $N_A(r')$ `*/`

9   **if** $ptr2[r']$ *is NULL* **then**

10   delete r' from R

11   **else if** $N_A(r')[ptr2[r']] = cur$ **then**

12                 $MinColor[cur] = \min(MinColor[cur], Color(xr'))$

        `/* Type 2 work : Lines 13 to 19 - Update` $Next$ `of non-edges between` cur `and` Q `*/`

13   **for** *each* q' *in* Q **do**

14   **while** $ptr1[q']$ *is not NULL and* $\widehat{N}_A(q')[ptr1[q']] < cur$ **do**

15   Increment the pointer $ptr1[q']$ `/*` $ptr1[q']$ `becomes NULL if it is incremented past the last element in` $\widehat{N}_A(q')$ `*/`

16   **if** $ptr1[q']$ *is NULL* **then**

17   delete q' from Q

18   **else if** $\widehat{N}_A(q')[ptr1[q']] = cur$ **then**

19                 $Next(cur\ q') = \min(Next(cur\ q'), MinColor[cur])$

gorithm traverses the sorted list $\widehat{N}_A(q')$ by updating $ptr1[q']$ as we did in Type 1 work of Algorithm 1.

By the time we have processed all $x \in A$ in their increasing order, all non-edges $e \in E(H)$ get processed and hence $Next(ab')$ for each $ab' \in E(H)$ is correctly computed as explained in the beginning of this section.

Lemma 2.21. *Time spent over all invocations of Algorithm 2 is $O(\xi n + n^2)$.*

Proof. Type 0 work done by Algorithm 2 (Lines 1 to 3) is similar to the Type 0 work of Algorithm 1 and hence the total time spent in Type 0 work over all invocations of Algorithm 2 is $O(\xi n + n^2)$.

Let us calculate the total time spent in Type 1 work. By similar arguments that were used to count the number of times Line 14 of Algorithm 1 is executed, we can show that over all invocations of Algorithm 2, the number of times Line 7 is executed is $O\left(\sum_{b' \in B} deg_A(b')(n_1 - deg_A(b'))\right) = O(\xi n)$. Similar to the analysis of Line 13 of Algorithm 1, over all invocations of Algorithm 2 the time spent in Line 6 with R being found empty is $O(n^2)$. It is easy to see that the number of times Line 6 gets executed with R being found non-empty is at most the number of times Line 7 is executed and Lines 9 - 12 get executed at most once for each such execution of Line 6. Also, the deletion of r' in Line 10 can be done in unit time. Hence the total time spent for Type 1 work over all invocations of Algorithm 2 is $O(\xi n + n^2)$.

Now consider Type 2 work. By similar arguments that were used to count the number of times Line 7 of Algorithm 1 is executed, we can show that over all invocations of Algorithm 2, the number of times Line 14 is executed is $O\left(\sum_{b' \in B} (n_1 - deg_A(b'))deg_A(b')\right) = O(\xi n)$. Similar to the analysis of Line 6 of Algorithm 1, over all invocations of Algorithm 2, the time spent in Line 13 with Q being found empty is $O(n^2)$. It is easy to see that the number of times Line 13 gets executed with Q being found non-empty is at most the number of times Line 14 is executed and Lines 16 - 19 get executed at most once for each such execution of Line 13. Also, the deletion of q' in Line 17 can be done in unit time. Hence the total time spent for Type 2 work over all invocations of Algorithm 2 is $O(\xi n + n^2)$.

Thus the total time spent over all invocations of Algorithm 2 is $O(\xi n + n^2)$ as claimed. □

Computing $MaxS_i$ and obtaining an optimal box representation of G. Once $Next(ab')$ for each $ab' \in E(H)$ is correctly computed by invoking Algorithm 2 for each $x \in A$, we compute $MaxS_i = \{\Gamma_{ab'} \in S_i \mid Next(ab') > i\} = \{\Gamma_{ab'} \mid ab' \in E(H) \text{ and } Color(ab') \leq i \text{ and } Next(ab') > i\}$, for $1 \leq i \leq k$. This can be done in overall $O(k \cdot |E(H)|) = O(kn^2)$ time.

For $1 \leq i \leq k$, let $E_i = \{e \in E(H) \mid \Gamma_e \in MaxS_i\}$. As mentioned in the beginning of Section 2.6, $\{G_i = \overline{H_i} \mid H_i = (V, E_i), 1 \leq i \leq k\}$, gives an optimal box representation of G. Since each G_i can be computed from E_i in $O(n^2)$, the above box representation can be obtained in $O(kn^2)$ time. Thus, we have the following theorem.

Theorem 2.22. *If G is a co-bipartite circular arc graph with cliques A and $V \setminus A$, then, an optimal box representation of G can be computed in $O(\xi n + kn^2)$ time, where $\xi = \min($number of edges between A and $V \setminus A$ in G, number of edges between A and $V \setminus A$ in $\overline{G})$ and $k = \text{box}(G)$.*

2.7 An approximation algorithm for the cubicity of circular arc graphs

Given any interval graph I on n vertices, we can represent it as the intersection of at most $\lceil \log n \rceil$ unit interval graphs and such a representation can be computed in polynomial time [28]. Therefore, it is easy to observe that our algorithm for computing a $\left(2 + \frac{1}{k}\right)$ factor optimal box representation of CA graphs immediately gives an algorithm to get a $\left(2 + \frac{1}{k}\right) \cdot \lceil \log n \rceil$ factor optimal cube representation of CA graphs. However, we can improve this to a $(2 + \frac{\lceil \log n \rceil}{k})$ factor as stated below.

Theorem 2.23. *Let G be a CA graph. A $\left(2 + \frac{\lceil \log n \rceil}{k}\right)$-factor approximation for $\text{cub}(G)$ can be computed in $O(mn + n^2)$ time and a cube representation of G of dimension at most $2 \cdot \text{cub}(G) + \lceil \log n \rceil$ can be computed in $O(mn + kn^2)$ time, where $m = |E(G)|$, $n = |V(G)|$ and $k = \text{cub}(G)$.*

Proof. In Section 2.5, we saw that for every CA graph G, we can construct two supergraphs $G_0(V, E_0)$ and $G_1(V, E_1)$ such that G_0 is a co-bipartite CA graph and G_1 is an interval graph and $G = G_0 \cap G_1$. As mentioned in Remark 2.1 at the end of Section 2.3, the optimal box representation $\mathcal{B}_0 = \{I'_1, I'_2, \cdots, I'_b\}$ of the co-bipartite CA graph G_0 obtained using the algorithm of Section 2.6, is also an optimal cube representation of G_0 because each I'_i, $1 \le i \le b$ is a unit interval graph.

Using the method of [28], we can compute a cube representation of the interval graph G_1 of dimension $\lceil \log n \rceil$ in $O((m' + n) \log n)$ time, where $m' = |E(G_1)|$ and $n = |V(G_1)| = |V(G)|$. However, since vertices in A are universal vertices in G_1, we can do this computation in $O((m + n) \log n)$ time, where $m = |E(G)|$. For this, we will first compute a cube representation of the graph G'_1, which is the extension of $G[V \setminus A]$ on the vertex set $(V \setminus A) \cup \{x\}$, where x is an arbitrarily chosen representative vertex in A. Since $|E(G'_1)| \le m + n$, we can compute a cube representation $\mathcal{B}'_1$ of G'_1 of dimension $\lceil \log n \rceil$, in $O((m + n) \log n)$ time, using the method of [28]. In each interval graph in $\mathcal{B}'_1$, assign the interval of each vertex of A to be the same as the interval corresponding to the representative vertex x. This will give us a cube representation $\mathcal{B}_1$ of G_1, because in G_1, every vertex y in A is adjacent to x and the neighborhoods of x and y are the same.

Since $G = G_0 \cap G_1$, $\mathcal{B}'' = \mathcal{B}_0 \cup \mathcal{B}_1$ is a cube representation of G. The dimension of $\mathcal{B}''$ is $b + \lceil \log n \rceil$, where $b = \mathrm{box}(G_0) = \mathrm{cub}(G_0)$. By Lemma 2.9, $\mathrm{box}(G_0) \leq 2\,\mathrm{box}(G) \leq 2\,\mathrm{cub}(G)$, implying that $\mathcal{B}''$ is of dimension at most $2\,\mathrm{cub}(G) + \lceil \log n \rceil$. The time complexity of this algorithm is $O(mn + kn^2)$, because the time complexity is dominated by the time taken to compute $\mathcal{B}_0$. □

2.8 Conclusion

We showed that, for a co-bipartite CA graph G, an optimal box representation of G can be obtained in polynomial time. Later, using some structural properties of co-bipartite CA graphs, we made this algorithm more efficient and showed that $\mathrm{box}(G)$ can be computed in $O(mn+n^2)$ time and an optimal box representation of G can be obtained in $O(mn+kn^2)$ time, where $m = |E(G)|$, $n = |V(G)|$ and $k = \mathrm{box}(G)$. The algorithms developed for co-bipartite CA graphs are used as subroutines in all the remaining algorithms in this chapter. We gave an algorithm to compute a box representation of an arbitrary CA graph G, of dimension at most $2\,\mathrm{box}(G) + 1$. We also explained how to compute box representations of proper CA graphs, of dimension at most two more than the optimum. We also gave an algorithm to compute a cube representation of a CA graph G of dimension at most $2\,\mathrm{cub}(G) + \lceil \log n \rceil$. The time required for approximating the boxicity (resp. cubicity) is $O(mn + n^2)$ and the time required for computing the box (resp. cube) representation is $O(mn + kn^2)$, in all the above algorithms.

Chapter 3

Approximating the cubicity of trees

It is NP-hard to decide whether cubicity of a graph is at most k, even for $k = 2$ or $k = 3$. Moreover, cubicity is known to be inapproximable in polynomial time, within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$, unless NP = ZPP.

In this chapter[1] we present a randomized algorithm that runs in polynomial time and computes cube representations of trees, of dimension within a constant factor of the optimum. If we do not insist for a cube representation, then the cubicity of trees can be approximated within a constant factor in polynomial time, without using any randomization. As far as we know, this is the first constant factor approximation algorithm for computing the cubicity of trees. It is not known whether computing the cubicity of trees is NP-hard or not.

3.1 Introduction

Recall that in Section 1.1 we defined a d-dimensional cube representation of a graph G as a geometric representation of G as an intersection graph of d-dimensional axis-parallel unit hypercubes and the cubicity of G, $\mathrm{cub}(G)$, as the smallest dimension d such that G can be represented as an intersection graph of d-dimensional axis-parallel unit hypercubes. In other words, $\mathrm{cub}(G)$ is the smallest dimension d for which G is a unit disc graph in $\mathbb{R}^d$, under the l^∞ metric.

In Chapter 2, we also saw a combinatorial redefinition that $\mathrm{cub}(G)$ is the smallest integer d such that G can be represented as the intersection of d

[1]Joint work with Manu Basavaraju, L. Sunil Chandran, Deepak Rajendraprasad and Naveen Sivadasan.

unit interval graphs on the same vertex set $V(G)$; i.e there exist unit interval graphs $I_1, I_2, \ldots, I_d$ with $V(I_i) = V(G)$ for each $1 \leq i \leq d$ and $E(G) = E(I_1) \cap E(I_2) \cap \cdots \cap E(I_d)$. If the requirement of unit interval graphs is relaxed to interval graphs the corresponding parameter was defined as boxicity.

It is known that $\text{box}(G) \leq \text{cub}(G) \leq \text{box}(G)\lceil \log \alpha(G) \rceil$, where $\alpha(G)$ is the cardinality of a maximum independent set in G [5]. Boxicity (resp. cubicity) of a graph on n vertices is at most $\left\lfloor \frac{n}{2} \right\rfloor$ (resp. $\left\lceil \frac{2n}{3} \right\rceil$) [78]. By convention, cubicity and boxicity of a complete graph are zero. It follows from the definitions that $\text{cub}(G) \leq 1$, if and only if G is a unit interval graph and $\text{box}(G) \leq 1$, if and only if G is an interval graph.

Since unit interval graphs are polynomial time recognizable, whether $\text{cub}(G) \leq 1$ is polynomial time decidable. However, deciding whether a graph has cubicity at most k is NP-hard in general. Yannakakis [98] showed that deciding whether $\text{cub}(G) \leq 3$ is NP-hard, even for co-bipartite graphs. Later, while studying unit disc graph recognition problems, Breu et al. [18] showed that deciding $\text{cub}(G) \leq 2$ is also NP-hard. Adiga et al. [3] showed that boxicity and cubicity problems are inapproximable in polynomial time, within an $O(n^{0.5-\epsilon})$ factor for any $\epsilon > 0$, unless NP = ZPP, even for graph classes like bipartite, co-bipartite, and split graphs. Recently, Chalermsook et al. [25] improved this hardness result by bettering the $O(n^{0.5-\epsilon})$ factor to an $O(n^{1-\epsilon})$ factor. Even for special classes of graphs, there were no good approximation algorithms known to exist for these problems; an exception being the case of circular arc graphs which was discussed in Chapter 2.

In this chapter, we present a randomized algorithm that runs in polynomial time, for computing cube representations of trees. Our algorithm computes cube representations of trees of dimension within a constant factor of the optimum. If we do not require a corresponding cube representation, then the cubicity of trees can be approximated within a constant factor in polynomial time, without using any randomization. As far as we know, the algorithm presented here is the first constant factor approximation algorithm for computing the cubicity of trees. It is not known whether computing the cubicity of trees is NP-hard or not.

Our randomized procedure borrows its ideas from the randomized algorithm devised by Krauthgamer et al. [65], for approximating the intrinsic dimensionality of trees. As we will see, this parameter is fundamentally different and is incomparable with cubicity in general. However, it comes as a surprise that their proof technique works more or less the same way for cubicity of trees, with some problem specific modifications to handle the details and the base cases. This is more surprising because Krauthgamer et al. [65] devised an $O(\log \log n)$ factor approximation for intrinsic dimensionality of general graphs by extending the the proof techniques used for trees; whereas cubicity for general graphs is inapproximable within $O(n^{1-\epsilon})$ factor for any

$\epsilon > 0$, unless NP=ZPP.

3.2 Preliminaries

In this chapter, we are dealing with only finite graphs, without self loops or multi edges. Unless specified otherwise, logarithms are taken to the base 2. A unit hypercube in $\mathbb{R}^d$ is a hypercube whose sides are of unit length in the usual Euclidean metric, i.e it is a disc in $\mathbb{R}^d$ of radius $\frac{1}{2}$ under the l^∞ metric. We consider our trees as rooted trees in which the root vertex is considered to be at depth zero and for any other vertex, its depth is given by its distance from the root. For any two vertices u and v of a tree T, the least common ancestor of u and v is the vertex with the minimum depth on the path between u and v in T. If u, v are two vertices in a graph G, we use $d_{uv}(G)$ to denote the distance between u and v in G and when it clear which graph we are talking out, we just use d_{uv}.

3.2.1 Cube representations, embeddings and weight-vector assignments to edges

Let G be a graph and suppose $f : V(G) \mapsto \mathbb{R}^d$ is such that $\|f(v) - f(u)\|_\infty \leq 1$ if and only if u and v are adjacent in G. If we consider unit hypercube corresponding to a vertex v as the unit hypercube centered at $f(v)$, then it is easy to see that the hypercubes corresponding to u and v intersect if and only if $\|f(v) - f(u)\|_\infty \leq 1$. Conversely, given a cube representation of G in d dimensions, for any $v \in V(G)$ we can define $f(v)$ as the vector corresponding to the center of the hypercube associated with v. Since we derived f from a cube representation of G, it follows from the definition that $\|f(v) - f(u)\|_\infty \leq 1$ if and only if u and v are adjacent in G. Thus, cubicity of a graph G is also the minimum dimension d such that there exist a function $f : V(G) \mapsto \mathbb{R}^d$ such that $\|f(v) - f(u)\|_\infty \leq 1$ if and only if u and v are adjacent in G.

Now we will turn our attention to the special case of trees and show that there is a correspondence between the maps from $V(T)$ to $\mathbb{R}^d$ as discussed above, and weight-vector assignments to edges $E(T) \mapsto [-1,1]^d$ with some nice properties. Let r denote an arbitrarily chosen root vertex of T and let h be the height of the rooted tree T. Suppose we have a weight-vector assignment $W : E(T) \mapsto [-1,1]^d$. For any vertex $v \neq r$, let $S_W(v)$ be the sum of weight-vectors of edges along the path in T from r to v, under the weight-vector assignment W and let $S_W(r)$ be the zero vector. Note that if u and v are adjacent in T, then $\|S_W(u) - S_W(v)\|_\infty \leq 1$.

Definition 3.1. Let W be a weight-vector assignment such that $W : E(T) \mapsto [-1,1]^d$ and S_W be defined with respect to W, as above. We say that W is a separating weight-vector assignment for a pair u, v of non-adjacent vertices of T, if $\|S_W(u) - S_W(v)\|_\infty > 1$.

If $W : E(T) \mapsto [-1,1]^d$ is a separating weight-vector assignment for every pair u, v of non-adjacent vertices of T, then the function $f : V(T) \mapsto \mathbb{R}^d$ defined as $f(v) = S_W(v)$ corresponds to a d-dimensional cube representation of T.

Conversely, given $f : V(T) \mapsto \mathbb{R}^d$ such that $\|f(v) - f(u)\|_\infty \leq 1$ if and only if u and v are adjacent in G, we can also get a corresponding weight-vector assignment $W : E(T) \mapsto [-1,1]^d$ such that $\|S_W(u) - S_W(v)\|_\infty > 1$, if and only if u and v are non-adjacent. If $uv \in E(T)$ such that u is the child vertex of v, then define $W(uv) = f(u) - f(v)$, which will be a vector belonging to $[-1,+1]^d$. From this, it is immediate that whenever u and v are adjacent, $\|S_W(u) - S_W(v)\|_\infty \leq 1$. If u and v are non-adjacent vertices, we had $\|f(u) - f(v)\|_\infty > 1$. Suppose a is the least common ancestor of u and v in T and $u = v_0, v_1, v_2, \ldots, v_{j-1}, a = v_j, v_{j+1}, v_k, v_{k+1} = v$ is the path in T between u and v. Since the path from $v_j = a$ to the root vertex is common to both the path from u to r and v to r, it is easy to see that $S_W(u) - S_W(v) = W(v_0, v_1) + W(v_1, v_2) + \cdots + W(v_{j-1}, v_j) - W(v_j, v_{j+1}) - \cdots - W(v_k, v)$. Therefore, $\|S_W(u) - S_W(v)\|_\infty = \|W(u, v_1) + W(v_1, v_2) + \cdots + W(v_{j-1}, v_j) - W(v_j, v_{j+1}) - \cdots - W(v_k, v)\|_\infty$. Since for any edge (v_i, v_{i+1}) in the uv path $W(v_i, v_{i+1}) = f(v_i) - f(v_{i+1})$, the RHS is equal to $\|f(u) - f(v)\|_\infty > 1$. We note down the following simple property, since it is used in later parts of this chapter as well.

Property 3.1. *Let T be a tree and $W : E(T) \mapsto [-1,1]^d$ and for any vertex v, let $S_W(v)$ be the sum of weight-vectors on the edges along the path in T from the root of T to v, under the weight-vector assignment W. Suppose $u = v_0, v_1, v_2, \ldots, v_k, v_{k+1} = v$ is the path in T between u and v. Then, $S_W(u) - S_W(v) = W(u, v_1) + W(v_1, v_2) + \cdots + W(v_{j-1}, v_j) - W(v_j, v_{j+1}) - \cdots - W(v_{k-1}, v_k) - W(v_k, v)$, where v_j is the least common ancestor of u and v in T.*

Our discussion is summarized below:

Lemma 3.1. *Given a cube representations of T of dimension d, in polynomial time we can compute weight-vector assignment $W : E(T) \mapsto [-1,1]^d$ that is a separating weight-vector assignment for every pair of non-adjacent vertices u and v of T. Conversely, given weight-vector assignment $W : E(T) \mapsto [-1,1]^d$ that is a separating weight-vector assignment for every pair of non-adjacent vertices u and v of T, then in polynomial time, we can obtain a d-dimensional cube representation of T.*

3.2.2 A lower bound for cubicity

In this section we demonstrate an important lower bound for the cubicity of general graphs and derive a lower bound in the special case of trees, using this general lower bound.

Lemma 3.2 ([26]). *If G is a graph of diameter $d > 0$, on n vertices, then* $\mathrm{cub}(G) \geq \left\lceil \frac{\log \alpha(G)}{\log(d+1)} \right\rceil$*, where $\alpha(G)$ is the cardinality of a maximum independent set in G.*

Proof. Suppose $\mathrm{cub}(G) = k$. This means that G can be represented as the intersection graph of axis parallel hypercubes in k dimensions. This cube representation, when projected to the k fundamental directions, give k unit interval supergraphs of G, say $I_1, I_2, \ldots, I_k$. Clearly, each I_i, $1 \leq i \leq k$ has diameter at most d and in any interval representation of I_i, the distance between the left end point of the left most unit interval and the right end point of the rightmost unit interval is at most $d+1$. This implies that the total volume occupied by the cube representation, in the k-dimensional Euclidean space is at most $(d+1)^k$. But we know that there are $\alpha(G)$ vertices such that unit volume hypercubes corresponding to no two of them share a common point. Therefore, the volume occupied by the cube representation is at least $\alpha(G)$ units. Thus we have, $(d+1)^k \geq \alpha(G)$. □

Definition 3.2. Let G be a connected graph of diameter d and for each $1 \leq r \leq d$ and $v \in V(T)$, let $B_{v,r}$ represent the set of vertices in G, which are at a distance at most r from v. Then, we define $\rho(G) = \max\limits_{v \in V, 1 \leq r \leq d} \dfrac{\log \frac{|B_{v,r}|}{2}}{\log(2r+1)}$. Note that, if G has at least three vertices, then $\lceil \rho(T) \rceil \geq 1$.

The following lemma is a direct consequence of the above definition.

Lemma 3.3. *Let G be a connected graph. For any $v \in V(G)$ and $1 \leq r \leq$ diameter(G), $|B_{v,r}| \leq 2(2r+1)^{\rho(G)}$.*

Theorem 3.4. *For any connected bipartite graph G,* $\mathrm{cub}(G) \geq \lceil \rho(G) \rceil$*. In particular, for any tree T,* $\mathrm{cub}(T) \geq \lceil \rho(T) \rceil$.

Proof. This directly follows from Lemma 3.2, because the subgraph of G induced on $B_{v,r}$ has an independent set of size at least $\frac{|B_{v,r}|}{2}$ and diameter at most $2r$. □

Remark 3.1. *Note that, though for a bipartite graph G its cubicity is at least $\rho(G)$, this need not be true in the case of general graphs. An easy counter example would be the case of cliques.*

3.2.3 Cube representations of short trees

In this section, we describe a way of constructing low dimensional cube representations of trees having relatively small height.

Lemma 3.5. *For any tree T on n vertices,* $\mathrm{cub}(T) \leq 1 + \lceil \log n \rceil$ *and a cube representation of T of dimension* $1 + \lceil \log n \rceil$ *can be constructed in polynomial time.*

Proof. Shah [83] describes a polynomial time algorithm for constructing two interval supergraphs I_1 and I_2 of T such that $V(T) = V(I_1) = V(I_2)$, I_1 is a unit interval graph and $E(T) = E(I_1) \cap E(I_2)$. Since we also know that any interval graph has $\lceil \log n \rceil$-dimensional cube representation and in polynomial time we can construct $\lceil \log n \rceil$ unit interval graphs on the same vertex set $V(T) = V(I_2)$ such that the intersection of their edge sets is $E(I_2)$ [28]. From this, the statement follows. □

Lemma 3.6. *Let T be a tree with* $\mathrm{cub}(T) \geq 2$ *and T_i be a subtree of T of height at most 2^{2^4}. Then, a cube representation of T_i of dimension* $\lceil c \times \rho(T) \rceil + 2 \leq (c+1) \times \mathrm{cub}(T)$ *or more can be constructed in polynomial time, where* $c = 22.77$*.*

Proof. If $\mathrm{cub}(T) \leq 1$, T should be path; otherwise, it has an induced star on four vertices, denoted as $K_{1,3}$, which forces $\mathrm{cub}(T) \geq 2$ [28]. Since we assumed that $\mathrm{cub}(T) \geq 2$, T contains an induced $K_{1,3}$ and therefore, $\lceil \rho(T) \rceil \geq 1$. If $\mathrm{cub}(T_i) \geq 2$, by Lemma 3.3, $|V(T_i)| \leq 2(2^{17}+1)^{\rho(T)}$. By Lemma 3.5, a cube representation of T of dimension $d \leq 2 + \lceil \rho(T) \log(2^{17}+1) \rceil$ can be constructed in polynomial time.

After getting a cube representation of T_i in a lower dimension d_1, it is a trivial job to extend it to a higher dimension d_2. Consider the cube representation as a mapping $f : V(T) \mapsto \mathbb{R}^{d_1}$, as described in Section 3.2.1 and for each $v \in V(T)$, append the vector $f(v)$ with $d_2 - d_1$ additional coordinates each of whose value is zero. By Lemma 3.4, the statement follows. □

3.2.4 Cubicity and intrinsic dimensionality

Let Z denote the set of integers and $\|\|_\infty$ denote the l^∞ norm. Let Z^d_∞ be the infinite graph with vertex set Z^d and an edge (u, v) for two vertices u and v if and only if $\|u - v\|_\infty = 1$. The intrinsic dimensionality of a graph G, $\dim(G)$ is the smallest d such that G can be injectively embedded on to Z^d_∞. This means that G occurs as a (not necessarily induced) subgraph of Z^d_∞.

Though both cubicity and intrinsic dimensionality are parameters related to graph embeddings, there are several fundamental differences between these two.

(1) **Injectivity:** Intrinsic dimensionality requires the mapping from $V(G)$ to

Z^d to be injective. Thus, dense graphs will have relatively high intrinsic dimensionality compared to sparse graphs. A clique on n vertices has intrinsic dimensionality $\log_2 n$. In contrast, the injectivity constraint is absent for cubicity. In a cube representation, the hypercubes corresponding to two distinct vertices are permitted to occupy the same space. Recall that a clique has cubicity zero.

(2) **Vertex Positioning:** In the case of intrinsic dimensionality, we should map the vertices of the graph to points in Z^d. However, as we saw in the previous section, the mappings associated with cubicity are from $V(G)$ to $\mathbb{R}^d$, giving us more freedom to place the hypercubes corresponding to the vertices. There are some graphs for which there is a cube representation in $\mathbb{R}^2$ even when its vertices have their neighborhoods different from each other, forcing an injective embedding from $V(G)$ to $\mathbb{R}^2$. For example, we can show that a graph having two cliques on n vertices and a matching connecting the corresponding pairs of vertices in both the cliques has cubicity two. Thus, even when cube representations have their corresponding vertex embedding injective, cubicity can be very low, due to the flexibility in vertex positioning.

(3) **Treatment of non-adjacency and monotonicity:** In the case of intrinsic dimensionality, it is possible to map even non-adjacent vertices u and v to points in Z^d which are at unit distance from each other. Because of this freedom, if a graph has intrinsic dimensionality k, its subgraphs will have intrinsic dimensionality at most k. Thus, intrinsic dimensionality is monotone, with respect to subgraph relation. In particular, all graphs of n vertices have intrinsic dimensionality at most that of a clique on n vertices, namely $\log_2 n$.

However, in the case of cube representations, we require the hypercubes corresponding to non-adjacent vertices to be non-intersecting and as we discussed in Section 3.2.1, the centers of hypercubes of non-adjacent vertices are required to be mapped to points in $\mathbb{R}^d$ which are at distance strictly more than one. For this reason, the monotonicity we observed in the case of intrinsic dimensionality does not happen for cubicity. A clique has cubicity zero, but almost all graphs on n vertices have cubicity $\Omega(n)$ [6].

(4) **Parameter value range:** As we noted, almost all graphs on n vertices have cubicity $\Omega(n)$. There are graphs on n vertices with cubicity $\left\lceil \frac{2n}{3} \right\rceil$. But the intrinsic dimensionality of a graph on n vertices is at most $\log_2 n$.

(5) **Good polynomial time approximations:** Krauthgamer et al. [65] defined a parameter called *growth rate* of a graph G, defined as

$$\eta(G) = sup \left\{ \frac{\log |B_{v,r}|}{\log r} \mid v \in V(G),\, r > 1 \right\}$$

Note that, this parameter is computable in polynomial time and for a graph on n vertices, the value of this parameter is at most $\log n$. Krauthgamer et al. [65] showed that for any graph G, its growth rate is a lower bound for its intrinsic dimensionality. They also showed that $dim(G)$ is $O(\eta(G) \log \eta(G))$

in general and in the special case of trees, $dim(G)$ is $O(\eta(G))$. This leads to an $O(\log\log n)$ factor approximation algorithm for the intrinsic dimensionality of general graphs and a constant factor approximation algorithm in the case of trees. For cubicity, the bound given by Lemma 3.2 is the only non-trivial polynomial time computable lower bound known. However, notice that this parameter can only go up to $\log_2 n$, whereas almost all graphs on n vertices have cubicity is $\Omega(n)$ [6]. Moreover, cubicity is known to be inapproximable in polynomial time, within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$, unless NP = ZPP.

Thus, cubicity and intrinsic dimensionality are two graph parameters, not directly comparable with each other in general. There are graphs for which cubicity exceeds intrinsic dimensionality, and for some others it is the other way. Even in the special case of trees, the intrinsic dimension and cubicity can be different. For example, a star graph $K_{1,n}$ has intrinsic dimension $\log_3(n+1)$, whereas the same graph has cubicity $\log_2 n$[67, 28].

In spite of all these contrasts between cubicity and intrinsic dimension, they share an interesting similarity: The injectivity requirement places a lower bound on the volume required for injectively embedding a graph on to Z_∞^d and this is the reason for having growth rate as a lower bound for intrinsic dimensionality. In the case of cubicity, cubes corresponding to non-adjacent pairs of vertices need to be non-intersecting, giving a lower bound to the volume required for placing the cubes. This fact was exploited to obtain the lower bounds given by Lemma 3.2 and Theorem 3.4. In a retrospective analysis, it appears that this similarity is what helped us to use the techniques developed by Krauthgamer et al. [65] in developing our algorithm. We will be showing that cubicity of a tree T is $O(\rho(T))$.

However, as we noted under item (5) above, this similarity between the parameters is not powerful enough to be useful in the case of general graphs, because of the approximation hardness results. The techniques do not seem to scale up even in other special cases, for example, for graphs without long induced simple cycles. Using the result obtained for trees, Krauthgamer et al. [65] had showed that graphs without induced simple cycles of length greater than λ have intrinsic dimensionality $O(\eta(G)\log^2(\lambda+2))$. But we know that even chordal graphs can have cubicity as high as $\Omega(n)$, whereas the lower bound obtained from Lemma 3.2 can be at most $\log n$. Moreover, even for split graphs which form a subclass of chordal graphs, cubicity is known to be NP-hard to approximate within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$, unless NP = ZPP.

3.3 Constructing the cube representation

Only cliques have cubicity zero. If a tree has a vertex of degree three, its cubicity is greater than one, since it has an induced $K_{1,3}$ [28]. Therefore, a tree of cubicity one can be only a path, whose unit interval representation is easy to construct. Hence, for the remaining parts of this chapter, we assume that $\text{cub}(T) \geq 2$. This also means that $n \geq 4$ and $\lceil \rho(T) \rceil \geq 1$.

In the previous section, we saw that for a tree T, $\lceil \rho(T) \rceil$ is a lower bound for $\text{cub}(T)$. Since $\rho(T)$ can be computed in polynomial time by its definition, if we can show the existence of a constant c such that $\text{cub}(T) \leq c\lceil \rho(T) \rceil$ for any tree T, then $c\lceil \rho(T) \rceil$ will serve as a polynomial time computable c factor approximation for $\text{cub}(T)$. The existence and determination of such a constant is proved using probabilistic arguments and the techniques we describe below are essentially derived from the techniques used in Krauthgamer et al. [65]. The method also gives a randomized algorithm to compute the corresponding cube representation.

3.3.1 A recursive decomposition of trees

We first define a recursive decomposition of the rooted tree T into rooted subtrees.

Let h denote the height of the tree T. Let $k = \lceil \log \log h \rceil$ and $\Gamma = 2^{2^k}$. Clearly, $\sqrt{\Gamma} = 2^{2^{k-1}} < h \leq 2^{2^k} = \Gamma$. For each $0 \leq i \leq k-1$, let $h_i = \Gamma^{\frac{1}{2^i}}$. Thus, $h_0 = \Gamma$ and $h_{i+1} = \sqrt{h_i}$. Let e denote the minimum even integer such that $h_e \leq 2^{16}$ and o denote the minimum odd integer such that $h_o \leq 2^{16}$. (This means $\{h_e, h_o\} = \{2^{2^3}, 2^{2^4}\}$).

For each integer i such that $\max(e, o) \geq i \geq 0$ we define two sets of rooted subtrees of T as follows: If we delete all edges of T that connect vertices at depth j and $j+1$ for each j which is a positive integer multiple of $3h_i$, the tree T gets decomposed into several vertex disjoint subtrees. We consider each such subtree as a rooted subtree with its root being the vertex in the subtree of smallest depth with respect to T. We denote this family of rooted subtrees of T as $\mathcal{A}_i$. In a similar way, let $\mathcal{B}_i$ denote the family of rooted subtrees of T, obtained by deleting all edges of T that connect vertices at depth j and $j+1$ for each j such that $j \equiv h_i \mod 3h_i$. Let O_i^A denote the the set of edges deleted from T to form $\mathcal{A}_i$ and let O_i^B denote the set of edges deleted from T to form $\mathcal{B}_i$. Let $\mathcal{L}_i = \mathcal{A}_i \cup \mathcal{B}_i$.

Lemma 3.7. *For each i such that $\max(e, o) \geq i \geq 0$:*

1. *The rooted trees in $\mathcal{L}_i$ have height at most $3h_i$.*
2. *Trees in $\mathcal{A}_{i+1}$ are subtrees of trees in $\mathcal{A}_i$ and trees in $\mathcal{B}_{i+1}$ are subtrees of trees in $\mathcal{B}_i$. This is because $O_i^A \subseteq O_{i+1}^A$ and $O_i^B \subseteq O_{i+1}^B$.*
3. *Vertex sets of trees in $\mathcal{A}_i$ partition $V(T)$. Same is the case with $\mathcal{B}_i$s.*
4. *If u and v are two vertices such that $d_{uv} \leq h_i$, then there exist at least one subtree $F \in \mathcal{L}_i$ such that both u and v belong to $V(F)$.*

Proof. The first three parts of the lemma follow directly from the definitions. Here we will prove the last part of the lemma.

Assume that $d_{uv} \leq h_i$ in T and let x be the least common ancestor of u and v in T. Without loss of generality, let $d_{vx} \leq d_{ux} \leq h_i$. Let F be the tree in $\mathcal{A}_i$ such that $x \in V(F)$ and let r be the root of F. We know that $d_{rx} \leq 3h_i$, by construction of F. If $d_{rx} \leq 2h_i$, then $d_{rv} \leq d_{ru} = d_{rx} + d_{xu} \leq 2h_i + h_i \leq 3h_i$ and therefore, $v, u \in V(F)$, by construction.

On the other hand, if $d_{rx} > 2h_i$, then $\exists y \in V(F)$ such that $d_{ry} = h_i + 1$ and y is on the path from r to x in T. By our construction, y becomes the root of a tree $F' \in \mathcal{B}_i$. Since $d_{rx} \leq 3h_i$ by construction of F and $d_{ry} = h + 1$, we have $d_{xy} = d_{rx} - d_{ry} < 2h_i$. This gives $d_{uy} = d_{ux} + d_{xy} < h_i + 2h_i = 3h_i$ Similarly, $d_{vy} = d_{vx} + d_{vy} < h_i + 2h_i = 3h_i$. Therefore, $v, u \in V(F')$, by construction. $\square$

Definition 3.3. If $T_1, T_2, \ldots, T_k$ are trees with disjoint vertex sets and for $1 \leq j \leq k$, $W_j : E(T_j) \mapsto [-1,1]^d$, then a weight-vector assignment $W : E(T_1) \cup E(T_2) \cup \cdots \cup E(T_k) \mapsto [-1,1]^d$ can be obtained by assigning $W(e) = W_j(e)$, where T_j is the tree containing the edge e. Then, W is the weight-vector assignment for $E(T_1) \cup E(T_2) \cup \cdots \cup E(T_k)$ derived from $W_1, W_2, \ldots, W_k$.

3.3.2 A randomized algorithm for constructing the cube representation

From our definitions, $\{h_e, h_o\} = \{2^{2^3}, 2^{2^4}\}$. The idea of recursive decomposition of trees and extending the weight-vector assignments of smaller trees to weight-vector assignments of bigger trees was used by Krauthgamer et al. [65] to attain injectivity while embedding the vertices in Z^d_∞. As we will explain soon, the same technique helps us to make sure that the hypercubes corresponding to non-adjacent vertex pairs do not intersect. The algorithm for constructing a weight-vector assignment for $E(T)$ that separates every pair of non-adjacent vertices of T is given below:

1. Using Lemma 3.6, construct cube representations of dimension $t = \lceil 22.77 \times \rho(T) \rceil + 2$ for each of the subtrees belonging to $\mathcal{L}_e \cup \mathcal{L}_o$.

2. Using the correspondence given in Section 3.2.1 between cube representations and weight-vector assignments, for each tree $F \in \mathcal{A}_e \cup \mathcal{B}_e \cup \mathcal{A}_o \cup \mathcal{B}_o$, compute a weight-vector assignment $W_e^F : E(F) \mapsto [-1,1]^t$. Notice that $\bigcup_{F \in \mathcal{A}_e} E(F) = E(T) \setminus O_e^A$. Combine the weight-vector assignments of trees in $\mathcal{A}_e$ as in Definition 3.3 and obtain $W_e^A : E(T) \setminus O_e^A \mapsto [-1,1]^t$. Similarly, obtain $W_e^B : E(T) \setminus O_e^B \mapsto [-1,1]^t$ from weight-vector assignments of trees in $\mathcal{B}_e$, $W_o^A : E(T) \setminus O_o^A \mapsto [-1,1]^t$ from weight-vector assignments of trees $F \in \mathcal{A}_o$ and $W_o^B : E(T) \setminus O_o^B \mapsto [-1,1]^t$ from weight-vector assignments of trees in $\mathcal{B}_o$.

3. Set $i = \max(e, o)$ and repeat steps 3a to 3d while $i > 1$.

   (a) For each edge uv belonging to $E(T) \setminus O_i^A$, assign $W_{i-2}^A(uv) = W_i^A(uv)$ and for each edge uv belonging to $E(T) \setminus O_i^B$, assign $W_{i-2}^B(uv) = W_i^B(uv)$.

   (b) For each tree $F \in \mathcal{A}_{i-2}$, do the following: For each edge uv of F such that $uv \in O_i^A \setminus O_{i-2}^A$, $W_{i-2}^A(uv)$ is assigned a weight-vector from $\{-1, 1\}^t$, chosen uniformly at random. Now, each edge uv of F has got a weight-vector under W_{i-2}^A. For each vertex v of F, compute $S(v)$ as the sum of weight-vectors on edges of the path in F from the root of F to v, as given by W_{i-2}^A. For each pair of non-adjacent vertices u and v of F such that $d_{uv} \geq h_{i-1}$, check whether $\|S(v) - S(u)\|_\infty > 1$. Repeat Step 3b, until the above condition becomes true simultaneously for all pair of non-adjacent vertices u and v of F such that $d_{uv} \geq h_{i-1}$.

   (c) For each tree $F \in \mathcal{B}_{i-2}$, do the following: For each edge uv of F such that $uv \in O_i^B \setminus O_{i-2}^B$, $W_{i-2}^B(uv)$ is assigned a weight-vector from $\{-1, 1\}^t$, chosen uniformly at random. Now, each edge uv of F has got a weight-vector under W_{i-2}^B. For each vertex v of F, compute $S(v)$ as the sum of weight-vectors on edges of the path in F from the root of F to v, as given by W_{i-2}^B. For each pair of non-adjacent vertices u and v of F such that $d_{uv} \geq h_{i-1}$, check whether $\|S(v) - S(u)\|_\infty > 1$. Repeat Step 3c, until the above condition becomes true simultaneously for all pair of non-adjacent vertices u and v of F such that $d_{uv} \geq h_{i-1}$.

   (d) Set $i = i - 1$.

4. For each edge uv belonging to $E(T) \setminus O_1^A$, assign $W_0'^A(uv) = W_1^A(uv)$ and for each edge uv belonging to O_1^A, assign the all zeros vector to $W_0'^A(uv)$. Similarly, for each edge uv belonging to $E(T) \setminus O_1^B$, assign $W_0'^B(uv) = W_1^B(uv)$ and for each edge uv belonging to O_1^B, assign the all zeros vector to $W_0'^B(uv)$.

5. Output $W_0^A \circ W_0^B \circ W_0'^A \circ W_0'^B$, a weight-vector assignment from $E(T)$ to $[-1, 1]^{4t}$ obtained by concatenating the components of weight assignments W_0^A, W_0^B, $W_0'^A$ and $W_0'^B$ together.

Property 3.2. *$W_0^A \circ W_0^B$ is a separating weight-vector assignment for every non-adjacent pair of vertices u and v of T such that $d_{uv} \leq h_e$ or $h_{i-1} \leq d_{uv} \leq h_{i-2}$, for any even integer i such that $e \geq i \geq 2$. Similarly, $W_0'^A \circ W_0'^B$ is a separating weight-vector assignment for every non-adjacent pair of vertices u and v of T such that $d_{uv} \leq h_o$ or $h_{i-1} \leq d_{uv} \leq h_{i-2}$, for any odd integer i such that $o \geq i \geq 3$.*

Proof. Let u and v be two non-adjacent vertices in T. If $d_{uv} \leq h_e$, by part 4 of Lemma 3.7, there exist at least one subtree $F \in \mathcal{L}_e$ such that both u and v belong to $V(F)$. In step 2 of the algorithm, we computed W_e^F from a cube representation of F, which is a separating weight-vector assignment by the correspondence given in Lemma 3.1. If $F \in \mathcal{A}_e$, then W_e^F is one of the weight-vector assignment from which W_e^A is derived, and on each edge of the path from u to v in T, the weight-vector assigned by W_e^A is the same as the weight-vector assigned by W_e^F. Since each edge xy of the path from u to v in T belongs to $E(T) \setminus O_e^A$, in Step 3a the algorithm assigns $W_{i-2}^A(xy) = W_i^A(xy)$ for each even integer i where $e \geq i \geq 2$. Thus, finally we will have $W_0^A(xy) = W_e^A(xy) = W_e^F(xy)$. Therefore, by Property 3.1 it follows that W_0^A will be a separating weight-vector assignment for u and v. By similar reasons, if $F \in \mathcal{B}_e$, W_0^B will be a separating weight-vector assignment for u and v.

Similarly, if $h_{i-1} \leq d_{uv} \leq h_{i-2}$, for any even integer i such that $e \geq i \geq 2$, then by part 4 of Lemma 3.7 there exist at least one subtree $F \in \mathcal{L}_{i-2}$ such that both u and v belong to $V(F)$. If $F \in \mathcal{A}_{i-2}$, in step 3a of the algorithm we would have made sure that W_{i-2}^A is a separating weight-vector assignment for u and v. As in the earlier case, for each edge xy of the path from u to v in T, $W_0^A(xy) = W_{i-2}^A(xy)$ and by Property 3.1, W_0^A will be a separating weight-vector assignment for u and v. Similarly, if $F \in \mathcal{B}_{i-2}$, W_0^B will be a separating weight-vector assignment for u and v.

Thus, for every non-adjacent pair of vertices u and v of T such that $d_{uv} \leq h_e$ or $h_{i-1} \leq d_{uv} \leq h_{i-2}$ for any even integer i such that $e \geq i \geq 2$ one of W_0^A and W_0^B is a separating weight-vector assignment, which implies that $W_0^A \circ W_0^B$ is a separating weight-vector assignment for u and v.

The proof of the second part of the lemma is similar. If u and v are non-adjacent pairs of vertices of T such that $d_{uv} \leq h_o$ or $h_{i-1} \leq d_{uv} \leq h_{i-2}$, for any odd integer i such that $o \geq i \geq 3$, then there exist at least one subtree $F \in \mathcal{L}_{i-2}$ such that both u and v belong to $V(F)$. If $F \in \mathcal{A}_{i-2}$, we get $W_0'^A(xy) = W_1^A(xy) = W_{i-2}^A(xy)$ and if $F \in \mathcal{B}_{i-2}$, we get $W_0'^B(xy) = W_1^B(xy) = W_{i-2}^B(xy)$, for each edge xy of the path from u to v in T. This implies that $W_0'^A \circ W_0'^B$ is a separating weight-vector assignment for u and v. □

The following is a direct consequence of Property 3.2.

Theorem 3.8. *$\mathcal{W} = W_0^A \circ W_0^B \circ W_0'^A \circ W_0'^B$ is a separating weight-vector assignment for each non-adjacent pair of vertices u and v of T. Here, $W : E(T) \mapsto [-1,1]^{4t}$, where $t = \lceil 22.77 \times \rho(T) \rceil + 2$.*

The following lemma will help us to calculate the expected number of times the algorithm repeats Step 3b (or 3c) till it obtains a suitable weight-vector assignment for a tree $F \in \mathcal{L}_{i-2}$, where $e \geq i \geq 2$.

Lemma 3.9. *Let i be such that $h_i \geq 2^{2^3}$ and $i \geq 2$. Let $W_i : E(T) \setminus O_i^A \mapsto [-1,1]^t$, where $t = \lceil 22.77 \times \rho(T) \rceil + 2$ and $F \in \mathcal{A}_{i-2}$. Suppose for each edge uv of F such that $uv \in E(T) \setminus O_i^A$, we set $W_{i-2}(uv) = W_i(uv)$ and for each edge uv of F such that $uv \in O_i^A \setminus O_{i-2}^A$, we assign $W_{i-2}(uv)$ to be a vector from $\{-1,1\}^t$ chosen independently and uniformly at random. For each vertex v of F, let $S_{W_{i-2}}(v)$ be the sum of edge weights of the edges belonging to the path from the root of F to v, as given by W_{i-2}. Then, with probability at least $p = 0.64$, for every pair of non-adjacent vertices u and v of F such that $d_{uv} \geq h_{i-1}$, $\|S_{W_{i-2}}(v) - S_{W_{i-2}}(u)\|_\infty > 1$.*

Proof. Consider a pair of non-adjacent vertices u and v belonging to the vertex set of the same rooted subtree $F \in \mathcal{A}_{i-2}$ and $d_{uv} \geq h_{i-1}$. Let r be the root of F. Since u and v both belong to the same subtree $F \in \mathcal{A}_{i-2}$, all the edges in the uv path fall in $E(T) \setminus O_{i-2}^A$. Therefore, all the edges in the uv path get their weight-vectors assigned under W_{i-2}. But since $d_{uv} \geq h_{i-1} = {h_i}^2$ and $h_i \geq 2^8$ and each subtree in $\mathcal{A}_i$ has height at most $3h_i$, among the edges in the uv path, at least $\frac{h_i}{4}$ edges should belong to $O_i^A \setminus O_{i-2}^A$ and got their weights assigned independently and uniformly at random from $\{-1,1\}^t$, as stated in the lemma. The other edges on the uv path were already assigned values in W_i and these values remain the same in W_{i-2}. Let $u = v_0, v_1, v_2, \ldots, v_q, v_{q+1} = v$ be the path in T between u and v, where v_j is the least common ancestor of u and v in T. Also let S^k denote the k^{th} coordinate function of $S_{W_{i-2}}$ and W^k denote the k^{th} coordinate function of W_{i-2}. By property 3.1, for each $1 \leq k \leq t$, $S^k(u) - S^k(v) = X_k + c_k$, where $X_k = \sum_{\{0 \leq i \leq j-1 \text{ and } v_i v_{i+1} \in O_{i-2}^A \setminus O_i^A\}} W^k(xy) - \sum_{\{j \leq i \leq q \text{ and } v_i v_{i+1} \in O_{i-2}^A \setminus O_i^A\}} W^k(xy)$ and $c_k \in \mathbb{R}$ is a constant, depending on the weight vectors fixed by W_i for edges in the uv path that belong to $E(T) \setminus O_i^A$. Let l be the number of edges in the uv path that belong to $E(T) \setminus O_i^A$.

We will bound the probability that $|S^k(v) - S^k(u)| \leq 1$. Note that X_k is the sum of l iid random variables, each of which is -1 or $+1$ with equal probability. Therefore,

$$Pr(|S^k(v) - S^k(u)| \leq 1) = Pr(X_k \text{ falls in the interval } [-c_k - 1, -c_k + 1])$$

But since X_k can take only integer values and X_k can take at most two possible values in $[-c_k - 1, -c_k + 1]$ irrespective of whether l is even or odd, because any interval of length two can contain at most two integers of the same parity. Therefore, $Pr(|S^k(v) - S^k(u)| \leq 1) \leq 2\binom{l}{\lceil \frac{l}{2} \rceil} 2^{-l}$. Since $l \geq \frac{h_i}{4} \geq 2^6$, using Sterling's approximation formula,

$$Pr(|S^k(v) - S^k(u)| \leq 1) \leq \frac{1.61}{\sqrt{l}} \leq \frac{1.61}{\sqrt{\frac{h_i}{4}}}$$

$$Pr(\|S_{W_{i-2}}(v) - S_{W_{i-2}}(u)\|_\infty \leq 1) \leq \left(\frac{1.61}{\sqrt{\frac{h_i}{4}}} \right)^t$$

Since the height of F is at most $3h_{i-2}$, by Lemma 3.3, there are at most $2(6h_{i-2}+1)^{\rho(T)}$ vertices in T_i and the number of non-adjacent pairs $u, v \in V(F)$ such that $h_{i-1} \leq d_{uv} \leq 2 \times h_{i-1}$, is at most $4(6h_{i-2}+1)^{\rho(T)}\,(2 \times 2h_{i-1}+1)^{\rho(T)}$.

For each integer l where $1 \leq l \leq \log(h_{i-1})$, let $\mathcal{P}_l$ denote the set consisting of the non-adjacenct pairs $u, v \in V(F)$ such that $2^{l-1}h_{i-1} \leq d_{uv} \leq 2^l h_{i-1}$. Using Lemma 3.3, it is easy to see that for each integer l where $1 \leq l \leq \log(h_{i-1})$, $|\mathcal{P}_l| \leq 4(6h_{i-2}+1)^{\rho(T)}\left(2^l 2h_{i-1}+1\right)^{\rho(T)}$. Using similar arguments as given in the previous paragraph, we also get the following:

For each pair $(u, v) \in \mathcal{P}_l$,

$$Pr(\|S_{W_{i-2}}(v) - S_{W_{i-2}}(u)\|_\infty \leq 1) \leq \left(\frac{1.61}{\sqrt{(2^{l-1} \times \frac{h_i}{4})}}\right)^t$$

Applying union bound,

$$Pr(\exists u, v \in V(F) \text{ with } d_{uv} \geq h_{i-1} \text{ and } \|S_{W_{i-2}}(v) - S_{W_{i-2}}(u)\|_\infty \leq 1)$$

$$\begin{aligned}
&\leq \sum_{l=1}^{\log(h_{i-1})} |\mathcal{P}_l| \left(\frac{1.61}{\sqrt{(2^{l-1} \times \frac{h_i}{4})}}\right)^t \\
&\leq \sum_{l=1}^{\log(h_{i-1})} 4(6h_{i-2}+1)^{\rho(T)}\left(2^l 2h_{i-1}+1\right)^{\rho(T)} \left(\frac{1.61}{\sqrt{(2^{l-1} \times \frac{h_i}{4})}}\right)^t \\
&\leq 8(6h_{i-2}+1)^{\rho(T)}\,(2 \times 2h_{i-1}+1)^{\rho(T)} \left(\frac{1.61}{\sqrt{\frac{h_i}{4}}}\right)^t \\
&\leq 0.33, \text{since } t \geq \lceil 22.77 \times \rho(T) \rceil + 2,\ h_i \geq 2^{2^3} \text{ and } h_{i-2} = h_{i-1}^2 = h_i^4.
\end{aligned}$$

Therefore, with probability at least 0.67, for every pair of non-adjacent vertices u and v of F such that $d_{uv} \geq h_{i-1}$, $|S_{w_i}(v) - S_{w_i}(u)| > 1$ for some k such that $1 \leq k \leq t$. ∎

Lemma 3.10. *The expected number of times the algorithm repeats Step 3b (or 3c) till it obtains a suitable weight-vector assignment for a tree $F \in \mathcal{L}_{i-2}$ is at most $\frac{1}{0.67}$ for any i such that $e \geq i \geq 2$.*

Theorem 3.11. *For any T, we can compute a $4(\lceil 22.77 \times \rho(T) \rceil + 2)$-dimensional cube representation using a randomized algorithm which runs in time polynomial in expectation. Cubicity of trees can be approximated within a constant factor in deterministic polynomial time.*

Proof. The second part of the theorem follows from the first part, because $\rho(T)$ is a polynomial time computable function. Since by Lemma 3.1, in polynomial time we can construct a d-dimensional cube representation of T from a weight-vector assignment $W : E(T) \mapsto [-1,1]^d$, it is enough to show that the randomized algorithm we described here, for computing a weight-vector assignment $W : E(T) \mapsto [-1,1]^{4t}$, where $t = \lceil 22.77 \times \rho(T) \rceil + 2$ runs in time polynomial in expectation.

In any partition of the rooted tree T into smaller trees, there can be at most $O(n)$ rooted subtrees. Therefore, by Lemma 3.6, step 1 of the algorithm runs in polynomial time. In step 2 of the algorithm, the weight-vector assignments can be computed in polynomial time, by Lemma 3.1. The operation in step 2 of combining the weight assignments on smaller trees as given in Definition 3.3 can easily be done in polynomial time. By the definition of the recursive decomposition, Step 3 is executed at most $O(\log \log h)$ rounds, where h is the height of the tree T. It is easy to see that the assignments in step 3a can be done in polynomial time. By Lemma 3.10, for each round of execution of step 3, steps 3b and 3c are repeated only constantly many times in expectation. In each repetition, the algorithm does only a polynomial time operation. Steps 4 and 5 are simple assignments, which can be done in polynomial time. □

3.4 Conclusion

In this chapter, we show that cubicity of trees can be approximated within a constant factor, in deterministic polynomial time. As far as we know, this is the first constant factor approximation algorithm known for cubicity of trees. A corresponding cube representation of the tree can also be computed by a randomized algorithm which runs in time polynomial in expectation. The basic techniques for the randomized algorithm are borrowed from the techniques given by Krauthgamer et al. [65], for approximating the intrinsic dimensionality of trees. We feel that this is a surprising coincidence because as we explained in Section 3.2.4, intrinsic dimensionality is quite different from cubicity and neither the bounds of these parameters nor the proof techniques for these problems work for each other in general. As far as we know, till now there are no works connecting the parameters cubicity and intrinsic dimension.

Chapter 4

Approximation algorithms for boxicity and cubicity

The problem of computing boxicity (resp. cubicity) is known to be inapproximable in polynomial time even for graph classes like bipartite, co-bipartite and split graphs, within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$, unless NP = ZPP. We[1] prove that if a graph G on n vertices has a clique on $n - k$ vertices, then $\text{box}(G)$ can be computed in time $n^2 2^{O(k^2 \log k)}$. Using this fact, various FPT approximation algorithms for boxicity are derived. The parameter used is the vertex (or edge) edit distance of the input graph from certain graph families of bounded boxicity - like interval graphs and planar graphs. Using the same fact, we also derive an $O\left(\frac{n\sqrt{\log \log n}}{\sqrt{\log n}}\right)$ factor approximation algorithm for computing the boxicity and an $O\left(\frac{n(\log \log n)^{\frac{3}{2}}}{\sqrt{\log n}}\right)$ factor approximation algorithm for computing the cubicity. To our knowledge, these are the first $o(n)$ factor approximation algorithms for computing boxicity and cubicity of general graphs. As a consequence of this result, a $o(n)$ factor approximation algorithm for computing the partial order dimension of finite posets and a $o(n)$ factor approximation algorithm for computing the threshold dimension of split graphs would follow. We also present an FPT approximation algorithm for computing the cubicity of graphs, with vertex cover number as the parameter.

4.1 Introduction

Let $G(V,\ E)$ be a graph. Recall that a set of interval graphs (resp. unit interval graphs) $\{I_1,\ I_2,\ \ldots,\ I_k\}$ is called a box (resp. cube) representation of

[1]Joint work with Abhijin Adiga and L. Sunil Chandran. An initial version of this work was presented in IPEC 2012.

G of dimension k if $I_1, I_2, \ldots, I_k$ have the same vertex set V and $E(G) = E(I_1) \cap E(I_2) \cap \cdots \cap E(I_k)$ (See Definition 2.1). Also, the boxicity (resp. cubicity) of an incomplete graph G, $\mathrm{box}(G)$ (respectively $\mathrm{cub}(G)$), was defined as the minimum integer k such that G has a box (resp. cube) representation of dimension k. For a complete graph, it is defined to be zero.

The decision problem BOXICITY takes a graph on n vertices and an integer b as inputs and asks whether $\mathrm{box}(G) \leq b$. Cozzens [38] proved that this problem is NP-hard. In fact, determining whether $\mathrm{box}(G) \leq 2$ (resp. $\mathrm{cub}(G) \leq 2$) is itself NP-hard ([64, 18]). Moreover, it is not possible to approximate boxicity and cubicity within a factor of $O(n^{1-\epsilon})$ for any $\epsilon > 0$ in polynomial time unless NP = ZPP [25]. In this work, we present $o(n)$ factor approximation algorithms for computing boxicity and cubicity - the first of their kind, to our knowledge.

Since NP-hard problems are often impractical to solve, it is natural to introduce parameters along with the input, and design algorithms which run in polynomial time for small values of the parameter. We say that a decision problem with input size n and a parameter k is Fixed Parameter Tractable (FPT) if the problem can be decided in time $f(k) \cdot n^{O(1)}$, for some computable function f. Often, a similar terminology is used in the case of optimization problems too. An FPT approximation algorithm is an approximation algorithm that runs in $f(k) \cdot n^{O(1)}$ time. For an introduction to parameterized complexity, please refer to [74].

The standard parameterization of BOXICITY using boxicity itself as the parameter k is meaningless since the problem is NP-hard even for $k = 2$. Parameterizations with vertex cover number (MVC), minimum feedback vertex set size (FVS) and max leaf number as parameters were studied by Adiga et al. [7]. With vertex cover number as the parameter k, they gave an algorithm which computes boxicity exactly in $2^{O(2^k k^2)} n$ time, and another algorithm which gives an additive one approximation for boxicity in $2^{O(k^2 \log k)} n$ time, where n is the number of vertices in the graph. Using FVS as the parameter k, they gave a $2 + \frac{2}{\mathrm{box}(G)}$ factor approximation algorithm to compute boxicity that runs in $2^{O(2^k k^2)} n^{O(1)}$ time. With max leaf number as the parameter k, they gave an additive two approximation algorithm for boxicity that runs in $2^{O(k^3 \log k)} n^{O(1)}$ time. In 2011, Ganian [52] showed that the FPT algorithms and approximations for boxicity with parameter vertex cover can be easily generalized for the parameter twin cover. Very recently, Bruhn et al. [20] provided additive one FPT algorithms for boxicity with the parameter pathwidth and also with the parameter cluster vertex deletion number. Their algorithm for boxicity with parameter pathwidth improves the previously known (including one of our results in this chapter) approximation guarantee bound for boxicity with the parameter maximum leaf number from additive two to additive one.

In this work, we consider vertex and edge edit distance from families of graphs of bounded boxicity as parameters. The notion of edit distance refers, in general, to the smallest number of some well-defined modifications to be applied to the input graph so that the resultant graph possesses some desired properties. Edit distance from graph classes is a well-studied problem in parameterized complexity [21, 57, 70, 97].

Cai [22] introduced a framework for parameterizing problems with edit distance as the parameter. For a family $\mathcal{F}$ of graphs, and $k \geq 0$ an integer, the author used $\mathcal{F}+ke$ (respectively, $\mathcal{F}-ke$) to denote the family of graphs that can be converted to a graph in $\mathcal{F}$ by deleting (respectively, adding) at most k edges, and $\mathcal{F}+kv$ to denote the family of graphs that can be converted to a graph in $\mathcal{F}$ by deleting at most k vertices. Cai [22] considered the parameterized complexity of the vertex coloring problem on $\mathcal{F}-ke$, $\mathcal{F}+ke$ and $\mathcal{F}+kv$ for various families $\mathcal{F}$ of graphs, with k as the parameter. This was further studied by Marx [69].

In the same framework, we consider the parameterized complexity of computing the boxicity of $\mathcal{F}+k_1e-k_2e$ and $\mathcal{F}+kv$ graphs for families $\mathcal{F}$ of bounded boxicity graphs, using k_1+k_2 and k as parameters. We will see that many relevant parameters for the boxicity problem, including MVC and FVS considered by Adiga et al. [7], are special cases of our parameters. We provide an improved FPT algorithm with the parameter FVS and give FPT approximation algorithms with some parameters smaller than MVC. With the parameter max leaf number, our method achieves the same result as obtained in Adiga et al. [7]. (See corollaries 1-7 for more details.)

We also give a factor-2 FPT approximation algorithm for cubicity, using vertex cover number as the parameter. This can be improved to a $(1+\epsilon)$ factor algorithm for any $\epsilon > 0$, by sacrificing more on the running time.

4.2 Prerequisites

In this section, we give some basic facts necessary for the later part of this chapter. For a vertex $v \in V$ of a graph G, we use $N_G(v)$ to denote the set of neighbors of v in G. We use $G[S]$ to denote the induced subgraph of $G(V,E)$ on the vertex set $S \subseteq V$. If I is an interval representation of an interval graph $G(V,E)$, as in Chapter 2 we use $l_v(I)$ and $r_v(I)$ respectively to denote the left and right end points of the interval corresponding to $v \in V$ in I. The interval corresponding to v is denoted as $\Big[l_v(I), r_v(I)\Big]$.

Lemma 4.1 (Roberts [78]). *Let $G(V,\ E)$ be any graph. For any $x \in V$, $\mathrm{box}(G) \leq 1 + \mathrm{box}(G \setminus \{x\})$.*

Lemma 4.2 and Lemma 4.3 given below are just restatements respectively of Lemma 2.5 and Lemma 2.9 of Chapter 2, presented in a slightly different format suitable for this chapter. For easy reference, their proofs are also included here with slight modifications.

Lemma 4.2. *Let $G(V, E)$ be a graph on n vertices. Let $S \subseteq V$ be such that $\forall v \in V \setminus S$ and $u \in V$ such that $u \neq v$, $(u, v) \in E$. If a k-dimensional box representation $\mathcal{B}_S$ of $G[S]$ is known, then, in $O(kn)$ time we can construct a box representation $\mathcal{B}$ of G of dimension $|\mathcal{B}_S|$. Moreover,* $\operatorname{box}(G) = \operatorname{box}(G[S])$.

Proof. Let $\mathcal{B}_S = \{I_1, I_2, \ldots, I_p\}$ be a box representation of $G[S]$. For $1 \leq i \leq p$, let $l_i = \min\limits_{u \in S} l_u(I_i)$ and $r_i = \max\limits_{u \in S} r_u(I_i)$. For $1 \leq i \leq p$ define I'_i by the interval assignment

$$[l_v(I'_i), r_v(I'_i)] = \begin{cases} [l_v(I_i), r_v(I_i)] & \text{if } v \in S, \\ [l_i, r_i] & \text{if } v \in V \setminus S. \end{cases}$$

It is easy to see that $\mathcal{B}_2 = \{I'_1, I'_2, \ldots, I'_p\}$ is a box representation of G and $\operatorname{box}(G) \leq \operatorname{box}(G[S])$. Since $G[S]$ is an induced subgraph of G, we also have $\operatorname{box}(G) \geq \operatorname{box}(G[S])$. The whole construction can be done in $O(kn)$ time. □

Lemma 4.3. *Let $G(V, E)$ be a graph on n vertices and let $A \subseteq V$. Let $G_1(V, E_1)$ be a supergraph of G with $E_1 = E \cup \{(x, y) \mid x, y \in A,\ x \neq y\}$. If a box representation $\mathcal{B}$ of G is known, then in $O(b \cdot n)$ time we can construct a box representation $\mathcal{B}_1$ of G_1 of dimension $2|\mathcal{B}|$. In particular,* $\operatorname{box}(G_1) \leq 2 \operatorname{box}(G)$.

Proof. Let $\mathcal{B} = \{I_1, I_2, \ldots, I_b\}$ be a box representation of G. For each $1 \leq i \leq b$, let $l_i = \min\limits_{u \in V} l_u(I_i)$ and $r_i = \max\limits_{u \in V} r_u(I_i)$. For $1 \leq i \leq b$, let I_{i_1} be the interval graph obtained from I_i by assigning the intervals

$$[l_v(I_{i_1}), r_v(I_{i_1})] = \begin{cases} [l_i, r_v(I_i)] & \text{if } v \in A, \\ [l_v(I_i), r_v(I_i)] & \text{if } v \in V \setminus A. \end{cases}$$

and let I_{i_2} be the interval graph obtained from I_i by assigning the intervals

$$[l_v(I_{i_2}), r_v(I_{i_2})] = \begin{cases} [l_v(I_i), r_i] & \text{if } v \in A, \\ [l_v(I_i), r_v(I_i)] & \text{if } v \in V \setminus A. \end{cases}$$

It is easy to see that this construction can be done in $O(b \cdot n)$ time.

Note that, in constructing I_{i_1} and I_{i_2} we have only extended some of the intervals of I_i and therefore, I_{i_1} and I_{i_2} are supergraphs of I_i and in turn of G. By construction, A induces cliques in both I_{i_1} and I_{i_2}, and thus they are supergraphs of G_1 too.

Now, consider $(u, v) \notin E$ with $u \in V \setminus A$, $v \in A$. Then $\exists i \in \{1, 2, \ldots, b\}$ such that either $r_v(I_i) < l_u(I_i)$ or $r_u(I_i) < l_v(I_i)$. If $r_v(I_i) < l_u(I_i)$, then clearly the intervals $[l_i, r_v(I_i)]$ and $[l_u(I_i), r_u(I_i)]$ do not intersect and thus $(u, v) \notin E(I_{i_1})$. Similarly, if $r_u(I_i) < l_v(I_i)$, then $(u, v) \notin E(I_{i_2})$. If both $u, v \in V \setminus A$

and $(u, v) \notin E$, then $\exists i$ such that $(u, v) \notin E(I_i)$ for some $1 \leq i \leq b$ and clearly by construction, $(u, v) \notin E(I_{i_1})$ and $(u, v) \notin E(I_{i_2})$.

It follows that $G_1 = \bigcap_{1 \leq i \leq b} I_{i_1} \cap I_{i_2}$ and $\mathcal{B}_1 = \{I_{1_1}, I_{1_2}, I_{2_1}, I_{2_2}, \ldots, I_{b_1}, I_{b_2}\}$ is a box representation of G_1 of dimension $2b$. If $|\mathcal{B}| = \operatorname{box}(G)$ to start with, then we get $|\mathcal{B}'| \leq 2 \operatorname{box}(G)$. Therefore, $\operatorname{box}(G_1) \leq 2 \operatorname{box}(G)$. □

We know that there are at most $2^{O(nb \log n)}$ distinct b-dimensional box representations of a graph G on n vertices and all these can be enumerated in time $2^{O(nb \log n)}$ [7, Proposition 1]. In linear time, it is also possible to check whether a given graph is a unit interval graph and if so, generate a unit interval representation of it [16]. Hence, a similar result holds for cubicity as well.

Proposition 4.1. *Let $G(V, E)$ be a graph on n vertices of boxicity (resp. cubicity) b. Then an optimal box (resp. cube) representation of G can be computed in $2^{O(nb \log n)}$ time.*

If $S \subseteq V$ induces a clique in G, then it is easy to see that the intersection of all the intervals in I corresponding to vertices of S is nonempty. This property is referred to as the *Helly property of intervals* and we refer to this common region of intervals as the *Helly region* of the clique S.

Definition 4.1. Let $G(V, E)$ be a graph in which $S \subseteq V$ induces a clique in G. Let $H(V, E')$ be an interval supergraph of G. Let p be a point on the real line. If H has an interval representation I satisfying the following conditions:

(1) p belongs to the Helly region of S in I.

(2) The end points of intervals corresponding to vertices of $V \setminus S$ are all distinct in I.

(3) For each $v \in S$,
$$l_v(I) = \min\left(p, \min_{u \in N_G(v) \cap (V \setminus S)} r_u(I)\right) \text{ and}$$
$$r_v(I) = \max\left(p, \max_{u \in N_G(v) \cap (V \setminus S)} l_u(I)\right)$$

then we call I a nice interval representation of H with respect to S and p. If H has a nice interval representation with respect to clique S and some point p, then H is called a nice interval supergraph of G with respect to clique S.

Lemma 4.4. *Let G be a graph on n vertices, with its vertices arbitrarily labeled as $1, 2, \ldots, n$. If G contains a clique of size $n - k$ or more, then :*

(a) A subset $A \subseteq V$ such that $|A| \leq k$ and $G[V \setminus A]$ is a clique, can be computed in $O(n2^k)$ time.

(b) There are at most $2^{O(k \log k)}$ nice interval supergraphs of G with respect to the clique $V \setminus A$. These can be enumerated in $n^2 2^{O(k \log k)}$ time.

(c) *If G has a box representation $\mathcal{B}$ of dimension b, then it has a box representation $\mathcal{B}'$ of the same dimension, in which $\forall I \in \mathcal{B}'$, I is a nice interval supergraph of G with respect to the clique $V \setminus A$.*

(d) *By construction, vertices of the nice interval supergraphs obtained in (b) and (c) retain their original labels as in G.*

Proof. (a) We know that, if G contains a clique of size $n-k$ or more, then the complement graph $\overline{G}$ has a vertex cover of size at most k. We can compute a minimum vertex cover A of $\overline{G}$ in $O(n2^k)$ time [74]. We have $|A| \leq k$ and $G[V \setminus A]$ is a clique because $V \setminus A$ is an independent set in $\overline{G}$.

(b) Let H be any nice interval supergraph of G with respect to $V \setminus A$. Let I be a nice interval representation of H with respect to $V \setminus A$ and a point p. Let P be the set of end points (both left and right) of the intervals corresponding to vertices of A in H. Clearly $|P| = 2|A| \leq 2k$. The order of end points of vertices of A in I from left to right corresponds to a permutation of elements of P and therefore, there are at most $(2k)!$ possibilities for this ordering. Moreover, note that the points of P divide the real line into $|P|+1$ regions and that p can belong to any of these regions. From the definition of nice interval representation, it is clear that, once the point p and the end points of vertices of A are fixed, the end points of vertices in $V \setminus A$ get automatically decided.

Thus, to enumerate every nice interval supergraph H of G with respect to clique $V \setminus A$, it is enough to enumerate all the $(2k)! = 2^{O(k \log k)}$ permutations of elements of P and consider $|P|+1 \leq 2k+1$ possible placements of p in each of them. Some of these orderings may not produce an interval supergraph of G though. In $O(k^2)$ time, we can check whether the resultant graph is an interval supergraph of G and output the interval representation in $O(n)$ time. The number of supergraphs enumerated is only $(2k+1)2^{O(k \log k)} = 2^{O(k \log k)}$.

(c) Let $\mathcal{B} = \{I_1, I_2, \ldots, I_b\}$ be a box representation of G. Without loss of generality, we can assume that all $2|V|$ interval end points are distinct in I_i, for $1 \leq i \leq b$. (Otherwise, we can always alter the end points locally and make them distinct.) Let $p_i \in \mathbb{R}$ be a point belonging to the Helly region corresponding to $V \setminus A$ in I_i. For $1 \leq i \leq b$, let I'_i be the interval graph defined by the interval assignments given below. Vertices of I'_i are assigned their original labels as in I_i.

$$[l_v(I'_i), r_v(I'_i)] = \begin{cases} [l_v(I_i), r_v(I_i)] & \text{if } v \in A, \\ [l'_v(i), r'_v(i)] & \text{if } v \in V \setminus A. \end{cases}$$

where $l'_v(i) = \min\left(p_i, \min\limits_{u \in N_G(v) \cap A} r_u(I_i)\right)$ and $r'_v(i) = \max\left(p_i, \max\limits_{u \in N_G(v) \cap A} l_u(I_i)\right)$.

Claim 4.4.1. *$\mathcal{B}' = \{I'_1, I'_2, \ldots, I'_b\}$ is a box representation of G such that $\forall I'_i \in \mathcal{B}'$, I'_i is a nice interval supergraph of G with respect to clique $V \setminus A$.*

Proof. Consider any $I'_i \in \mathcal{B}'$. For $u, v \in A$, intervals corresponding to u and v are the same in both I_i and I'_i. If $(u, v) \in E(G)$, with $u, v \in A$, then the intervals corresponding to u and v intersect in I'_i because they were intersecting in I_i. For any $(u, v) \in E(G)$, with $u \in A$ and $v \in V \setminus A$, the interval of v intersects the interval of u in I'_i, by the definition of $[l'_v(i), r'_v(i)]$. Vertices of $V \setminus A$ share the common point p_i. Thus, I'_i is an interval supergraph of G. It is easy to see that I'_i is a nice interval supergraph of G with respect to clique $V \setminus A$ and point p_i.

Since $\mathcal{B}$ is a valid box representation of G, for each $(u, v) \notin E(G)$, $\exists I_i \in \mathcal{B}$ such that $(u, v) \notin E(I_i)$. Observe that for any vertex $v \in V$, the interval of v in I_i contains the interval of v in I'_i. Therefore, if $(u, v) \notin E(I_i)$, then $(u, v) \notin E(I'_i)$ too. Thus, $\mathcal{B}'$ is also a valid box representation of G. □

(d) Since vertices of G are labeled $1, 2, \ldots, n$ initially (specified at the beginning of the statement of this lemma), we just need to retain the same labeling of vertices during the definition and construction of nice interval supergraphs of G. (We have included this obvious fact in the statement of the lemma, just to give better clarity.) □

4.3 Boxicity of graphs with large cliques

One of the central ideas in this chapter is the following theorem about computing the boxicity of graphs which contain very large cliques. Using this theorem, in Section 4.4 we derive $o(n)$ factor approximation algorithms for computing the boxicity and cubicity of graphs. Further, it is used in Section 4.5 to derive parameterized approximation algorithm for the boxicity problem parameterized by vertex edit distance from a family of graphs of bounded boxicity.

Theorem 4.5. *Let G be a graph on n vertices, containing a clique of size $n-k$ or more. Then,* $\mathrm{box}(G) \leq k$ *and an optimal box representation of G can be found in time $n^2 2^{O(k^2 \log k)}$.*

Proof. Let $G(V, E)$ be a graph on n vertices containing a clique of size $n - k$ or more. Arbitrarily label the vertices of G as $1, 2, \ldots, n$. Using part (a) of Lemma 4.4, we can compute in $O(n2^k)$ time, $A \subseteq V$ such that $|A| \leq k$ and $G[V \setminus A]$ is a clique. It is easy to infer from Lemma 4.1 that $\mathrm{box}(G) \leq \mathrm{box}(G \setminus A) + |A| = k$, since $\mathrm{box}(G \setminus A) = 0$ by definition.

From part (c) of Lemma 4.4, we get that, if $\mathrm{box}(G) = b$, then there exists a box representation $\mathcal{B}' = \{I'_1, I'_2, \ldots, I'_b\}$ of G in which each I'_i is a nice interval supergraph of G with respect to clique $V \setminus A$. We call such a representation a nice box representation of G with respect to clique $V \setminus A$. To construct a nice box representation of G with respect to clique $V \setminus A$ and of dimension d, we choose d of the $2^{O(k \log k)}$ nice interval supergraphs of G with respect to clique

$V \setminus A$ (guaranteed by part (b) of Lemma 4.4) and check if this gives a valid box representation of G. This validation is straightforward because vertices in supergraphs being considered retain their original labels as in G by part (d) of Lemma 4.4. All possible nice box representations of dimension d can be computed and validated in $n^2 2^{O(k \cdot d \log k)}$ time. We might have to repeat this process for $1 \leq d \leq b$ in that order, to obtain an optimal box representation. Hence the total time required to compute an optimal box representation of G is $bn^2 2^{O(k \cdot b \log k)}$, which is $n^2 2^{O(k^2 \log k)}$, because $b \leq k$ by the first part of this theorem. □

Remark 4.1. *Theorem 4.5 gives an FPT algorithm for computing the boxicity of G, with the parameter $k = MVC(\overline{G})$, where $\overline{G}$ is the graph complement of G.*

4.4 Approximation algorithms for computing boxicity and cubicity

In this section, we use Theorem 4.5 and derive $o(n)$ factor approximation algorithms for boxicity and cubicity. Let $G(V, E)$ be the given graph with $|V| = n$. Without loss of generality, we can assume that G is connected. Let $k = \frac{\sqrt{\log n}}{\sqrt{\log \log n}}$ and $t = \lceil \frac{n}{k} \rceil$. The algorithm proceeds by defining t supergraphs of G and computing their optimal box representations. Let the vertex set V be partitioned arbitrarily into t sets $V_1, V_2, \ldots, V_t$ where $|V_i| \leq k$, for each $1 \leq i \leq t$. We define supergraphs $G_1, G_2, \ldots, G_t$ of G with $G_i(V, E_i)$ defined by setting $E_i = E \cup \{(x, y) | x, y \in V \setminus V_i\}$, for $1 \leq i \leq t$.

Lemma 4.6. *Let G_i be as defined above, for $1 \leq i \leq t$. An optimal box representation $\mathcal{B}_i$ of G_i can be computed in $n^{O(1)}$ time, where $n = |V|$.*

Proof. Noting that $G[V \setminus V_i]$ is a clique and $|V_i| \leq k = \frac{\sqrt{\log n}}{\sqrt{\log \log n}}$, by Theorem 4.5, we can compute an optimal box representation $\mathcal{B}_i$ of G_i in $n^2 2^{O(k^2 \log k)} = n^{O(1)}$ time, where $n = |V|$. □

Lemma 4.7. *Let $\mathcal{B}_i$ be as computed above, for $1 \leq i \leq t$. Then, $\mathcal{B} = \bigcup_{1 \leq i \leq t} \mathcal{B}_i$ is a valid box representation of G such that $|\mathcal{B}| \leq t' \operatorname{box}(G)$, where t' is $O\left(\frac{n\sqrt{\log \log n}}{\sqrt{\log n}}\right)$. The box representation $\mathcal{B}$ is computable in $n^{O(1)}$ time.*

Proof. We can compute optimal box representations $\mathcal{B}_i$ of G_i, for $1 \leq i \leq t = \left\lceil \frac{n\sqrt{\log \log n}}{\sqrt{\log n}} \right\rceil$ as explained in Lemma 4.6 in total $n^{O(1)}$ time. Observe that $E(G) = E(G_1) \cap E(G_2) \cap \cdots \cap E(G_t)$. Therefore, it is a trivial observation that the union $\mathcal{B} = \bigcup_{1 \leq i \leq t} \mathcal{B}_i$ gives us a valid box representation of G.

We will prove that this representation gives the approximation ratio as required. By Lemma 4.3 we have, $|\mathcal{B}_i| = \operatorname{box}(G_i) \leq 2 \operatorname{box}(G)$. Hence, $|\mathcal{B}| =$

$\sum_{i=1}^{t} |\mathcal{B}_i| \leq 2t \operatorname{box}(G)$. Substituting $t = \left\lceil \frac{n\sqrt{\log\log n}}{\sqrt{\log n}} \right\rceil$ in this inequality gives the approximation ratio as required. □

The box representation $\mathcal{B}$ obtained from Lemma 4.7 can be extended to a cube representation $\mathcal{C}$ of G as stated in the following lemma.

Lemma 4.8. *A cube representation $\mathcal{C}$ of G, such that $|\mathcal{C}| \leq t' \operatorname{cub}(G)$, where t' is $O\left(\frac{n(\log\log n)^{\frac{3}{2}}}{\sqrt{\log n}}\right)$, can be computed in $n^{O(1)}$ time.*

Proof. We can compute optimal box representations $\mathcal{B}_i$ of G_i, for $1 \leq i \leq t = \left\lceil \frac{n\sqrt{\log\log n}}{\sqrt{\log n}} \right\rceil$ as explained in Lemma 4.6 in $O(n^4)$ time. By [5, Corollary 2.1] we know that, from an optimal box representation $\mathcal{B}_i$ of G_i, in $O(n^2)$ time, we can construct a cube representation $\mathcal{C}_i$ of G_i of dimension $\operatorname{box}(G_i)\lceil \log \alpha(G_i) \rceil$, where $\alpha(G_i)$ is the independence number of G_i which is at most $|V_i|$. (Recall the assumption that G is connected.)

It is easy to see that $\mathcal{C} = \bigcup_{1 \leq i \leq t} \mathcal{C}_i$ gives us a valid cube representation of G. We will prove that this cube representation gives the approximation ratio as required.

$$\begin{aligned} |\mathcal{C}| = \sum_{i=1}^{t} |\mathcal{C}_i| &\leq \sum_{i=1}^{t} |\mathcal{B}_i| \lceil \log \alpha(G_i) \rceil \leq \sum_{i=1}^{t} |\mathcal{B}_i| \lceil \log k \rceil \\ &\leq 2t \operatorname{box}(G) O(\log\log n) \leq O(t \log\log n) \operatorname{cub}(G) \end{aligned} \tag{4.1}$$

Substituting $t = \left\lceil \frac{n\sqrt{\log\log n}}{\sqrt{\log n}} \right\rceil$ in the inequality above gives the approximation ratio as required. □

Combining Lemma 4.7 and Lemma 4.8, we get the following theorem which gives $o(n)$ factor approximation algorithms for computing boxicity and cubicity.

Theorem 4.9. *Let $G(V, E)$ be a graph on n vertices. Then a box representation $\mathcal{B}$ of G, such that $|\mathcal{B}| \leq t \operatorname{box}(G)$, where t is $O\left(\frac{n\sqrt{\log\log n}}{\sqrt{\log n}}\right)$, can be computed in polynomial time. Further, a cube representation $\mathcal{C}$ of G, such that $|\mathcal{C}| \leq t' \operatorname{cub}(G)$, where t' is $O\left(\frac{n(\log\log n)^{\frac{3}{2}}}{\sqrt{\log n}}\right)$, can also be computed in polynomial time.*

Now, we use Theorem 4.9 and derive sublinear approximation algorithms for some other dimensional parameters closely related to boxicity.

4.4.1 Partial order dimension

A partially ordered set (poset) $\mathcal{P} = (X, P)$ consists of a nonempty set X and a binary relation P on X that is reflexive, antisymmetric and transitive. If every pair of distinct elements of X are comparable under the relation P, then (X, P)

is called a total order or a linear order. A linear extension of a partial order (X, P) is a linear order (X, P') such that $\forall x, y \in X$, $(x, y) \in P \Rightarrow (x, y) \in P'$. The dimension of a poset $\mathcal{P} = (X, P)$, denoted by $dim(\mathcal{P})$ is defined as the smallest integer k such that $\mathcal{P}$ can be expressed as the intersection of k linear extensions $(X, P_1), (X, P_2), \ldots, (X, P_k)$ of $\mathcal{P}$: i.e., if $\forall x, y \in X$, $(x, y) \in P \Leftrightarrow (x, y) \in P_i$, for each $1 \leq i \leq k$. This concept was introduced by Dushnik and Miller in 1941 [43].

A height-two poset is a poset (X, P) in which all elements of X are either minimal elements or maximal elements under the relation P. Even in the case of height-two posets, partial order dimension is hard to approximate within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$, unless NP = ZPP [25].

Corollary 4.10. *There is a polynomial time algorithm to approximate the partial order dimension of any poset $\mathcal{P} = (X, P)$ defined on a finite set X, within an $o(n)$ factor, where $n = |X|$.*

Proof. Assume that $\mathcal{P} = (X, P)$ defined on a finite set X.

We will first prove the statement for height-two posets. Adiga et al. [4] showed that if $\mathcal{P}$ is a height-two poset defined on a finite set X and G_P is the underlying comparability graph of $\mathcal{P}$ (i.e., X is the vertex set of G_P and two vertices are adjacent in G_P if and only if they are comparable under P), then $\mathrm{box}(G_P) \leq dim(\mathcal{P}) \leq 2\,\mathrm{box}(G_P)$. Since $\mathrm{box}(G_P)$ can be approximated in polynomial time within an $o(n)$ factor by Theorem 4.9, a polynomial time $o(n)$ factor approximation algorithm for computing the poset dimension of height-two posets follows.

By a construction given by R. Kimble [92], given a poset $\mathcal{P} = (X, P)$ of arbitrary height, we can construct a height-two poset $\mathcal{P}' = (S(X), P')$ from $\mathcal{P} = (X, P)$ in polynomial time so that $dim(\mathcal{P}) \leq dim(\mathcal{P}') \leq 1 + dim(\mathcal{P})$. Combined with this reduction, the polynomial time $o(n)$ factor approximation algorithm we obtained in the previous paragraph for height-two posets gets extended for posets of arbitrary height. □

4.4.2 Threshold dimension of split graphs

A graph $G(V, E)$ is called a threshold graph if there exists $s \in \mathbb{R}$ and a labeling of vertices $w : V \mapsto \mathbb{R}$ such that $\forall u, v \in V, (u, v) \in E \Leftrightarrow w(u) + w(v) \geq s$. The threshold dimension of G, denoted by $t(G)$ is the minimum integer k such that there exists threshold graphs $G_1, G_2, \ldots, G_k$ on the same vertex set as $V(G)$ with $E(G) = E(G_1) \cup E(G_2) \cup \cdots \cup E(G_k)$. The concept of threshold graphs and threshold dimension was introduced by Chvátal and Hammer [36] while studying some set-packing problems. Threshold dimension is also hard to approximate within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$, unless NP = ZPP [25]. The same hardness result holds for the restricted case of split graphs as well [3].

Corollary 4.11. *There is a polynomial time algorithm to approximate the threshold dimension of any split graph G within an $o(n)$ factor, where $n = |V(G)|$.*

Proof. Given any split graph G, Adiga et al. [3] gave a polynomial time method to construct another split graph H on the same vertex set such that $t(G) = \mathrm{box}(H)$. By Theorem 4.9, the result follows. □

4.5 Computing the boxicity of graphs with edit distances as the parameter

In this section we give parameterized approximation algorithms for the boxicity problem parameterized by various vertex (edge) edit distance parameters. A subset $S \subseteq V$ such that $|S| \leq k$ is called a **modulator** for an $\mathcal{F} + kv$ graph $G(V, E)$ if $G \setminus S \in \mathcal{F}$. Similarly, a set E_k of pairs of vertices such that $|E_k| \leq k$ is called a modulator for an $\mathcal{F} - ke$ graph $G(V, E)$ if $G'(V, E \cup E_k) \in \mathcal{F}$. Modulators for graphs in $\mathcal{F} + ke$ and $\mathcal{F} + k_1e - k_2e$ are defined in a similar manner. The following theorem is gives us a parameterized algorithm for computing the boxicity of $\mathcal{F} + kv$ graphs.

Theorem 4.12. *Let $\mathcal{F}$ be a family of graphs such that $\forall G' \in \mathcal{F}$, $\mathrm{box}(G') \leq b \leq n$. Let $T(n)$ denote the time required to compute a b-dimensional box representation of a graph belonging to $\mathcal{F}$ on n vertices. Let G be an $\mathcal{F} + kv$ graph on n vertices. Given a modulator of G, a box representation $\mathcal{B}$ of G, such that $|\mathcal{B}| \leq 2\,\mathrm{box}(G) + b$ can be computed in time $T(n-k) + n^2 2^{O(k^2 \log k)}$.*

Proof. Let $\mathcal{F}$ be the family of graphs of boxicity at most b. Let $G(V, E)$ be an $\mathcal{F} + kv$ graph on n vertices, with a modulator S_k on k vertices such that $G' = G \setminus S_k \in \mathcal{F}$. We define two supergraphs of G, namely $H_1(V, E_1)$ and $H_2(V, E_2)$ such that $E = E_1 \cap E_2$ with $\mathrm{box}(H_1) \leq 2\,\mathrm{box}(G)$, $\mathrm{box}(H_2) \leq b$ and their required valid box representations are computable within the time specified in the theorem. It is easy to see that the union of valid box representations of H_1 and H_2 will be a valid box representation $\mathcal{B}$ of G and hence $|\mathcal{B}| \leq \mathrm{box}(H_1) + \mathrm{box}(H_2) \leq 2\,\mathrm{box}(G) + b$. This will complete our proof of Theorem 4.12.

We define H_1 to be the graph obtained from G by making $V \setminus S_k$ a clique on $n - k$ vertices, without altering other adjacencies in G. Formally, $E_1 = E \cup \{(x, y) \mid x, y \in V \setminus S_k,\ x \neq y\}$. Using Theorem 4.5, we can get an optimal box representation $\mathcal{B}_1$ of H_1 in $n^2 2^{O(k^2 \log k)}$ time. By Lemma 4.3, $|\mathcal{B}_1| \leq 2\,\mathrm{box}(G)$.

We define H_2 to be the graph obtained from G by making each vertex in S_k adjacent to every other vertex in the graph and leaving other adjacencies in G unaltered. Formally, $E_2 = E \cup \{(x, y) \mid x \in S_k,\ y \in V,\ x \neq y\}$. Let $\mathcal{B}'$ be a box representation of G' of dimension at most b (computed in time $T(n-k)$).

Then, $\mathcal{B}'$ is a box representation of $H_2[V \setminus S_k]$ as well, because $H_2[V \setminus S_k] = G'$. By Lemma 4.2, $\text{box}(H_2) = \text{box}(H_2[V \setminus S_k])$ and a box representation $\mathcal{B}_2$ of H_2 of dimension at most $|\mathcal{B}'| \leq b$ can be produced in $O(n^2)$ time.

Since $G = H_1 \cap H_2$, $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ is a valid box representation of G, of dimension at most $2\,\text{box}(G) + b$. All computations were done in $T(n-k) + n^2 2^{O(k^2 \log k)}$ time. $\square$

Using a similar method, we also get a parameterized approximation algorithm for computing the boxicity of $\mathcal{F} + k_1e - k_2e$ graphs.

Theorem 4.13. *Let $\mathcal{F}$ be a family of graphs such that $\forall G' \in \mathcal{F}$, $\text{box}(G') \leq b \leq n$. Let $T(n)$ denote the time required to compute a b-dimensional box representation of a graph belonging to $\mathcal{F}$ on n vertices. Let G be an $\mathcal{F}+k_1e-k_2e$ graph on n vertices and let $k = k_1 + k_2$. Given a modulator of G, a box representation $\mathcal{B}$ of G, such that $|\mathcal{B}| \leq \text{box}(G) + 2b$, can be computed in time $T(n) + O(n^2) + 2^{O(k^2 \log k)}$.*

Proof. Let $\mathcal{F}$ be the family of graphs of boxicity at most b. Let $G(V, E)$ be an $\mathcal{F} + k_1e - k_2e$ graph on n vertices, where $k_1 + k_2 = k$. Let $E_{k_1} \cup E_{k_2}$ be a modulator of G such that $|E_{k_1}| = k_1$, $|E_{k_2}| = k_2$ and $G'(V, (E \cup E_{k_2}) \setminus E_{k_1}) \in \mathcal{F}$. Let $T \subseteq V(G)$ be the set of vertices incident with edges in $E_{k_1} \cup E_{k_2}$.

As in the proof of Theorem 4.12, we define two supergraphs of G, namely $H_1(V, E_1)$ and $H_2(V, E_2)$ such that $E = E_1 \cap E_2$ with $\text{box}(H_1) \leq 2b$, $\text{box}(H_2) \leq \text{box}(G)$ and their required valid box representations are computable within the time specified in the theorem. As earlier, the union of valid box representations of H_1 and H_2 will be a valid box representation of $\mathcal{B}$ of G and hence $|\mathcal{B}| \leq \text{box}(H_1) + \text{box}(H_2) \leq 2b + \text{box}(G)$. This will complete our proof of Theorem 4.13.

Let $H_1(V, E_1)$ be the graph obtained from G' by making T a clique, without altering other adjacencies in G'. Formally, $E_1 = E' \cup \{(x, y) | x, y \in T, x \neq y\}$. Let $\mathcal{B}'$ be a box representation of G' of dimension at most b computed in time $T(n)$. From the box representation $\mathcal{B}'$ of G', in $O(b \cdot n)$=$O(n^2)$ time we can construct (by Lemma 4.3) a box representation $\mathcal{B}_1$ of H_1 with dimension $2b$.

Let $H_2(V, E_2)$ be the graph obtained from G by making each vertex in $V \setminus T$ adjacent to every other vertex in the graph and leaving other adjacencies in G unaltered. Formally, $E_2 = E \cup \{(x, y) | x \in V \setminus T, y \in V, x \neq y\}$. Clearly, $|T| \leq 2k$ and therefore, using the construction in Proposition 4.1, an optimal box representation $\mathcal{B}_T$ of $H_2[T]$ can be computed in $2^{O(k^2 \log k)}$ time. By Lemma 4.2, $\text{box}(H_2) = \text{box}(H_2[T])$ and a box representation $\mathcal{B}_2$ of H_2 of dimension $\text{box}(H_2[T])$ can be computed from the box representation $\mathcal{B}_T$ of $H_2[T]$ in $O(n^2)$ time. Observe that $H_2[T] = G[T]$. Therefore, $|\mathcal{B}_2| = \text{box}(G[T]) \leq \text{box}(G)$, because $G[T]$ is an induced subgraph of G.

Since $G = H_1 \cap H_2$, $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ is a valid box representation of G, of dimension at most $\text{box}(G) + 2b$. All computations were done in $T(n) + O(n^2) +$

$2^{O(k^2 \log k)}$ time. □

Remark 4.2. *Though in Theorem 4.12 and Theorem 4.13 we assumed that a modulator of G for $\mathcal{F}$ is given, in several important special cases (as in the case of corollaries discussed below), the modulator for $\mathcal{F}$ can be computed from G in FPT time. Moreover, in those cases, $T(n)$ is a polynomial in n. Thus, the algorithms given by Theorem 4.12 and Theorem 4.13 turns out to be FPT approximation algorithms for boxicity.*

4.5.1 Corollaries of Theorem 4.12 and Theorem 4.13

FPT approximation algorithms for computing boxicity with various parameters of interest result as consequences of Theorem 4.12. It is easy to see that these parameters are special cases of the vertex edit distance parameter. The general procedure is:

(i) Use known FPT algorithms to compute the parameter of interest and obtain the modulator S_k for the corresponding family $\mathcal{F}$.

(ii) Compute a low dimensional box representation for the graph $G' = (G \setminus S_k) \in \mathcal{F}$, in polynomial time.

(iii) Use our algorithm of Theorem 4.12 to get the FPT approximation algorithm for computing boxicity with the parameter of interest.

Some corollaries of Theorem 4.12 are discussed below.

Corollary 4.14. *FVS as the parameter: The minimum number of vertices to be deleted from a graph G so that the resultant graph is acyclic is called the feedback vertex set size (FVS) of G. If $FVS(G) \leq k$, we get a $\left(2 + \frac{2}{\mathrm{box}(G)}\right)$ factor approximation for boxicity with FVS as the parameter k, which runs in time $2^{O(k^2 \log k)} n^{O(1)}$.*

Proof. If $FVS(G) \leq k$, using existing FPT algorithms [24], in $O(3.83^k k n^2)$ time we can compute a minimum feedback vertex set S of $G(V, E)$ such that $G' = G(V \setminus S)$ is a forest. Thus, with modulator S, $G \in \mathcal{F} + kv$, where $\mathcal{F}$ is the family of graphs which are forests. Since a box representation of dimension two can be computed in polynomial time for any forest [83], using our algorithm of Theorem 4.12, we get a $2 + \frac{2}{\mathrm{box}(G)}$ factor approximation for boxicity with FVS as the parameter k, which runs in time $2^{O(k^2 \log k)} n^{O(1)}$. □

Remark 4.3. *Note that, for the boxicity problem parameterized by FVS, the algorithm in Adiga et al. [7] gave the same approximation factor but with running time $2^{O(2^k k^2)} n^{O(1)}$. Our algorithm gives a better running time.*

Corollary 4.15. *Proper Interval Vertex Deletion number (PIVD) as the parameter: The minimum number of vertices to be deleted from the graph G, so that the resultant graph is a proper interval graph, is called $PIVD(G)$. If $PIVD(G) \leq k$, we get a $2 + \frac{1}{\mathrm{box}(G)}$ factor approximation for boxicity with PIVD as the parameter k, which runs in time $2^{O(k^2 \log k)} n^{O(1)}$.*

Proof. If $PIVD(G)$ is at most k, we can use the FPT algorithm running in $O(6^k kn^6)$ time for proper interval vertex deletion [96] to compute a $S \subseteq V$ with $|S| \leq k$ such that $G \setminus S$ is a proper interval graph. Thus, with modulator S, $G \in \mathcal{F} + kv$, where $\mathcal{F}$ is the family of all proper interval graphs. Since a box representation of dimension one can be computed in polynomial time for any proper interval graph [16], using our algorithm of Theorem 4.12, we get a $2 + \frac{1}{\mathrm{box}(G)}$ factor approximation for boxicity with PIVD as the parameter k, which runs in time $2^{O(k^2 \log k)} n^{O(1)}$. □

Remark 4.4. *It is easy to see that $PIVD(G) \leq MVC(G)$. Hence, $PIVD(G)$ is a better parameter than the parameter $MVC(G)$ discussed in Adiga et al. [7]. Our algorithm has the same running time as the additive one approximation algorithm with $MVC(G)$ as the parameter, discussed in Adiga et al. [7].*

Corollary 4.16. *Planar Vertex Deletion number (PVD) as the parameter: The minimum number of vertices to be deleted from G to make it a planar graph, is called the planar vertex deletion number of G. If $PVD(G) \leq k$, we get an FPT algorithm for boxicity, giving a $\left(2 + \frac{3}{\mathrm{box}(G)}\right)$ factor approximation for boxicity using planar vertex deletion number as the parameter.*

Proof. If $G \in$ Planar$+kv$, we can use the FPT algorithm running in $O(f(k)n^2)$ time for planar deletion [70] to compute a $S \subseteq V$ with $|S| \leq k$ such that $G \setminus S$ is planar. Thus, with modulator S, $G \in \mathcal{F} + kv$, where $\mathcal{F}$ is the family of planar graphs. Since planar graphs have 3 dimensional box representations computable in polynomial time [91], using our algorithm of Theorem 4.12, we get an FPT algorithm for boxicity, giving a $2 + \frac{3}{\mathrm{box}(G)}$ factor approximation for boxicity of graphs that can be made planar by deleting at most k vertices, using planar vertex deletion number as the parameter. □

Theorem 4.13 also gives us FPT approximation algorithms for computing boxicity with various parameters of interest.

Corollary 4.17. *Interval Completion number as the parameter: The minimum number of edges to be added to a graph G, so that the resultant graph is an interval graph, is called the interval completion number of G. If the interval completion number G is at most k, we get an FPT algorithm that achieves an additive 2 approximation for* $\mathrm{box}(G)$ *which runs in time $2^{O(k^2 \log k)} n^{O(1)}$.*

Proof. If the interval completion number of a graph $G(V, E)$ is at most k, we can use the FPT algorithm for interval completion [97] with running time $O(k^{2k}n^{O(1)}) = 2^{O(k \log k)}n^{O(1)}$ to compute E_k such that $|E_k| \leq k$ and $G'(V, E \cup E_k)$ is an interval graph. Thus, with modulator E_k, $G \in \mathcal{F} - ke$, where $\mathcal{F}$ is the class of interval graphs. Since a box representation of dimension one can be computed in polynomial time for any interval graph [16], combining with our algorithm of Theorem 4.13, we get an FPT algorithm that achieves an additive 2 factor approximation for $\text{box}(G)$, with interval completion number k as the parameter which runs in time $2^{O(k^2 \log k)}n^{O(1)}$. □

Corollary 4.18. *Proper Interval Edge Deletion number (PIED) as the parameter: The minimum number of edges to be deleted from the graph G, so that the resultant graph is a proper interval graph, is called $PIED(G)$. If $PIED(G)$ is at most k, we get an FPT algorithm that achieves an additive 2 approximation for* $\text{box}(G)$*, with $PIED(G)$ as the parameter k, which runs in time $2^{O(k^2 \log k)}n^{O(1)}$.*

Proof. If $PIED(G)$ is at most k, we can use the FPT algorithm running in $O(9^k n^{O(1)})$ time for proper interval edge deletion [96] to compute a $E_k \subseteq E$ with $|E_k| \leq k$ such that $G'(V, E \setminus E_k)$ is a proper interval graph. Thus, with modulator S, $G \in \mathcal{F} + ke$, where $\mathcal{F}$ is the family of all proper interval graphs. Since a box representation of dimension one can be computed in polynomial time for any interval graph, combining with our algorithm of Theorem 4.13, we get an FPT algorithm that achieves an additive 2 factor approximation for $\text{box}(G)$, with PIED as the parameter k, which runs in time $2^{O(k^2 \log k)}n^{O(1)}$. □

Corollary 4.19. *Planar Edge Deletion number (PED) as the parameter: The minimum number of edges to be deleted from G so that the resultant graph is planar is called $PED(G)$. If $PED(G) \leq k$, we get an FPT algorithm that gives an additive 6 approximation for* $\text{box}(G)$ *with $PED(G)$ as the parameter.*

Proof. If $PED(G) \leq k$, we can use the FPT algorithm for planar edge deletion [57] to compute $E_k \subseteq E$ such that $|E_k| \leq k$ and $G'(V, E \setminus E_k)$ is a planar graph. Thus, with modulator E_k, $G \in \mathcal{F} + ke$, where $\mathcal{F}$ is the class of planar graphs. Since planar graphs have 3 dimensional box representations computable in polynomial time [91], using our algorithm of Theorem 4.13, we get an FPT algorithm that gives an additive 6-factor approximation for $\text{box}(G)$ with $PED(G)$ as the parameter. □

Corollary 4.20. *Max Leaf number (ML) as the parameter: The number of the maximum possible leaves in any spanning tree of a graph G is called $ML(G)$. If $ML(G) \leq k$, we get an FPT algorithm that achieves an additive 2 approximation for* $\text{box}(G)$ *which runs in time $2^{O(k^3 \log k)}n^{O(1)}$.*

Proof. The underlying algorithm here is precisely that of Section 4.5. However, we adopt some of the ideas used in the proof given in Adiga et al. [7, Section 4] for our proof.

We assume that G is connected and the max leaf number of G is at most k. If the graph G is just a cycle on n vertices ($n \geq 3$), we know that $\text{box}(G) = 1$ if $n = 3$ and $\text{box}(G) = 2$ if $n > 3$. Thus, we can also assume that G is not a cycle. Moreover, the maximum degree of any vertex in G is at most k; otherwise we can start with a vertex of degree at least $k+1$ and grow it to a spanning tree with more than k leaves, which is a contradiction.

In Section 4.5, interval supergraphs H_1 and H_2 were obtained by modifying a certain graph G' whose edge edit distance to G is small. Here, we will define G' in a slightly different way and then define H_1 and H_2 in a similar way as we did in Section 4.5.

We start by defining a graph G_1, such that G is a subdivision of G_1. For this, we use the following result.

Property 4.1 (Fellows et al. [48]). *If the max leaf number of a graph G is at most k, then G is a subdivision of a graph $G_1(V', F)$ with $|V'| \leq 4k - 2$ and $V' \subseteq V$. (G_1 may contain multi edges and self loops.)*

Let $G_1(V', F)$ the graph given by Property 4.1. Since G is not a cycle, we can eliminate all degree two vertices from G_1 one by one, by edge contractions. Therefore, without loss of generality, we can assume that there are no degree two vertices in G_1 and V' is precisely the set of vertices of G whose degree is not equal to 2.

Claim 4.20.1. *There are at most $4k-2$ vertices in G, whose degree is not equal to 2.*

Proof. As explained above, we assume that there are no degree two vertices in G_1. Since G is a subdivision of G_1 and a subdivision only introduces degree 2 vertices, we can conclude that there are at most $4k - 2$ vertices in G, whose degree is not equal to 2. □

Let $E_k \subseteq E$ be the set of edges of G which have at least one of its incident vertices belonging to V'. Now, we will define G' as the graph with vertex set V and edge set $E \setminus E_k$.

Claim 4.20.2. *The graph $G'(V, E \setminus E_k)$ is an interval graph and it can be computed in polynomial time from G.*

Proof. Since G is a subdivision of $G_1(V', F)$, it is easy to see that, the graph $G'(V, E \setminus E_k)$ is just a collection of vertex disjoint paths and isolated vertices. It is straightforward to verify that G' is an interval graph. To compute G', we just need to compute E_k. Since E_k is defined from V' and G, we only need to compute the set V', which is precisely the set of vertices of G whose degree is not equal to 2. This can be done in polynomial time. □

Since G' is an interval graph, we have $\text{box}(G') \leq 1$ and an interval representation of G' can be constructed in linear time [16]. Let T be the set of vertices of G, which are incident to at least one edge in E_k. In other words, $T = V' \bigcup_{v \in V'} N_G(v)$. Since maximum degree of G is at most k (as explained at the beginning of this proof), we get $|T| \leq |V'| + k \cdot |V'| \leq (4k-2) + k \cdot (4k-2) = O(k^2)$. From the proof of Theorem 4.13 given in Section 4.5, we can notice that the proof goes through with this definition of T and the complexity of the algorithm depends only on $|T|$ and not on the number of edges being modified in G.

For clarity, we just repeat some important points of the algorithm of Section 4.5 here, with modifications occurring mainly in the running time analysis. Let $H_1(V, E_1)$ be the graph obtained from G' by making T a clique, without altering other adjacencies in G'. From the box representation of G' of dimension one, in $O(n)$ time we can construct (by Lemma 4.3) a box representation $\mathcal{B}_1$ of H_1 with dimension 2.

Let $H_2(V, E_2)$ be the graph obtained from G by making vertices in $V \setminus T$ adjacent to every other vertex in the graph and maintaining other adjacencies in G unaltered. As in Section 4.5, we have $H_2[T] = G[T]$. Hence, $\text{box}(H_2[T]) = \text{box}(G[T]) \leq \mathit{treewidth}(G[T]) + 2 \leq \mathit{treewidth}(G) + 2 \leq 2 \cdot ML(G) + 2 \leq 2k + 2$ [7, 29, 48]. We know that $|T| = O(k^2)$ and therefore, using the construction in Proposition 4.1, an optimal box representation $\mathcal{B}_T$ of $H_2[T]$ can be computed in $2^{O(k^3 \log k)}$ time and from $\mathcal{B}_T$, an optimal box representation of H_2 of dimension at most $\text{box}(G)$ is computed in polynomial time.

Union of box representations of H_1 and H_2 gives a $2 + \text{box}(G)$ dimensional box representation for G, obtained in $2^{O(k^3 \log k)} n^{O(1)}$ time. $\square$

Remark 4.5. *The FPT approximation algorithm for boxicity described above with ML as the parameter has the same running time and approximation ratio as the algorithm discussed in Adiga et al. [7].*

4.6 An FPT approximation algorithm for computing cubicity

Fellows et al. [49, Corollary 5] proved an existential result that for every fixed pair of integers k and b, there is an $f(k) \cdot n$ time algorithm which determines whether a given graph G on n vertices and $MVC(G) \leq k$ has cubicity at most b. In the theorem below, we derive a FPT approximation algorithm, for computing the cubicity of graphs, using their vertex cover number as the parameter. Our algorithm is constructive.

Theorem 4.21. *Let G be a graph on n vertices. A cube representation of G which is of dimension at most $2\,\text{cub}(G)$ can be computed in time $2^{O(2^k k^2)} n^{O(1)}$,* *where $k = MVC(G)$. By allowing a larger running time of $2^{O(g(k,\epsilon))}n^{O(1)}$, we can achieve a $(1+\epsilon)$ approximation factor, for any $\epsilon > 0$, where $g(k,\epsilon) = \frac{1}{\epsilon}k^3 2^{\frac{4k}{\epsilon}}$.*

Proof. Let $G(V,E)$ be a graph on n vertices. Without loss of generality, we can assume that G is connected. We can compute a minimum vertex cover of G in time $2^{O(k)}n^{O(1)}$ [74]. Let $S \subseteq V$ be a vertex cover of G of cardinality k. We define two supergraphs of G, namely $H_1(V,E_1)$ and $H_2(V,E_2)$ such that $E = E_1 \cap E_2$ with $\text{cub}(H_1) \leq \text{cub}(G)$ and $\text{cub}(H_2) \leq \text{cub}(G)$.

Let $S \subseteq V$ be a vertex cover of G of cardinality k. First we define an equivalence relation on the vertices of the independent set $V \setminus S$ such that vertices u and v are in the same equivalence class if and only if $N_G(u) = N_G(v)$. Let $A_1, A_2, \ldots, A_t$ be the equivalence classes. We define H_1 to be the graph obtained from G by making each A_i into a clique and maintaining other adjacencies as it is in G. Formally, $E_1 = E \cup \{(u,v) \mid u \neq v$ and u,v belong to the same A_i, for some $1 \leq i \leq t\}$.

For each A_i, let us consider the mapping $n_{A_i} : A_i \mapsto \{1, 2, \cdots, |A_i|\}$, where $n_{A_i}(v)$ is the unique number representing $v \in A_i$. (Note that if $u \in A_i$ and $v \in A_j$, where $i \neq j$, then, $n_{A_i}(u)$ and $n_{A_j}(v)$ could potentially be the same.) Let $s = \max_{1 \leq i \leq t} |A_i|$. We define one more partitioning of the independent set $V \setminus S$ into equivalence classes $B_1, B_2, \ldots, B_s$ such that for $1 \leq i \leq s$, $B_i = \{v \mid n_{A_j}(v) = i$, for some $1 \leq j \leq t\}$. We define H_2 to be the graph obtained from G by making each B_i into a clique, and making each vertex in S adjacent to every other vertex in V. Formally, $E_2 = \{(u,v) \mid u \neq v$ and $u \in S, v \in V\} \cup \{(u,v) \mid u \neq v$ and u,v belong to the same B_i, for some $1 \leq i \leq s\}$.

If u,v are two adjacent vertices of a graph G such that $N_G(u) \cup \{u\} = N_G(v) \cup \{v\}$, we call them as twin vertices . G' is called a reduced graph of G if G' is obtained from G by repeatedly contracting the edges among pairs of twin vertices.

Claim 4.21.1. *If G' is a reduced graph of G, then,* $\text{cub}(G) = \text{cub}(G')$ *and from an optimal cube representation $\mathcal{C}'$ of G', in polynomial time, we can obtain an optimal cube representation $\mathcal{C}$ of G.*

Proof. Let $\mathcal{C}' = \{I_1', I_2', \cdots, I_p'\}$ be an optimal cube representation of G'. For each $1 \leq i \leq p$, define the interval graph I_i as follows : If $u \in V(G')$, then the interval corresponding to u in I_i is same as it is in I_i'. If $u \in V(G) \setminus V(G')$, then $\exists v \in V(G')$ such that u,v are twins in G. In this case, define the interval corresponding to u in I_i is same as the interval of its twin v in I_i'. It can be verified that $\mathcal{C} = \{I_1, I_2, \cdots, I_p\}$ is a valid cube representation of G. Thus, $\text{cub}(G) \leq p$. Since G' is an induced subgraph of G, we also have $\text{cub}(G) \geq \text{cub}(G') = p$. □

Observe that in graph H_1, vertices in each A_i, $1 \leq i \leq t$ are twins of each other. We can construct a reduced graph H_1' of H_1 by contracting all vertices in A_i to a single vertex, for each $1 \leq i \leq t$. Now, H_1' has only $t + |S|$ vertices, which is at most $2^k - 1 + k$. It is known that $\text{cub}(H_1') \leq MVC(H_1') + \lceil \log(|V(H_1')| - MVC(H_1')) \rceil - 1$ [27]. Since $MVC(H_1') = k$, we get $\text{cub}(H_1') \leq 2k - 1$. Using the construction in Lemma 4.1, we can compute an optimal cube representation $\mathcal{C}_1'$ of H_1' in time $2^{O(2^k k^2)}$. By the claim above, from $\mathcal{C}_1'$ we can get an optimal cube representation $\mathcal{C}_1$ of H_1 in polynomial time, with $|\mathcal{C}_1| = \text{cub}(H_1')$. Observe that H_1' is an induced subgraph of G, which implies $|\mathcal{C}_1| \leq \text{cub}(G)$.

Similarly, observe that, in graph H_2, vertices in each B_i, $1 \leq i \leq s$ are twins of each other. We can construct a reduced graph H_2' of H_2 by contracting all vertices in B_i to a single vertex, for each $1 \leq i \leq s$ and contracting S to a single vertex. Now, H_2' is a graph on $s + 1$ vertices. We can also observe that H_2' is in fact a star graph with s leaves. In polynomial time, we can construct an optimal cube representation $\mathcal{C}_2'$ of H_2' which is of dimension $\lceil \log s \rceil$ [83]. As earlier, from $\mathcal{C}_2'$ we can get an optimal cube representation $\mathcal{C}_2$ of H_2 in polynomial time, with $|\mathcal{C}_2| = \text{cub}(H_2') = \lceil \log s \rceil$. Observe that H_2' is an induced subgraph of G, which implies $|\mathcal{C}_2| \leq \text{cub}(G)$.

It can be easily verified that $E = E_1 \cap E_2$ and hence $\mathcal{C}_1 \cup \mathcal{C}_2$ is a valid cube representation of G of dimension $|\mathcal{C}_1| + |\mathcal{C}_2| \leq 2\,\text{cub}(G)$, constructible in $2^{O(2^k k^2)} n^{O(1)}$ time.

We can also achieve a $(1+\epsilon)$ approximation factor, for any $\epsilon > 0$ by allowing a larger running time as explained below. Define $f(k_\epsilon) = k\left(1 + 2^{\frac{2k-1}{\epsilon}}\right)$, where $k = MVC(G)$. If $|V(G)| = n \leq f(k_\epsilon)$, then, by Lemma 4.1, we can get an optimal cube representation of G in time $2^{O(f^2(k_\epsilon)\log f(k_\epsilon))}$. Otherwise, we have $\frac{2k-1}{\lceil \log \lceil \frac{n-k}{k} \rceil \rceil} \leq \epsilon$. In this case, we use the construction described above, to get a cube representation of G of dimension $|\mathcal{C}_1| + |\mathcal{C}_2|$. We prove that in this case, $|\mathcal{C}_1| + |\mathcal{C}_2| \leq \text{cub}(G)(1+\epsilon)$.

It is known that $\text{cub}(G) \geq \lceil \log \psi(G) \rceil$, where $\psi(G)$ is the number of leaf nodes in the largest induced star in G [5]. By the pigeon hole principle, $\max_{v \in S} |N_G(v) \cap (V \setminus S)| \geq \left\lceil \frac{n-k}{k} \right\rceil$. Therefore, $\text{cub}(G) \geq \lceil \log \psi(G) \rceil \geq \left\lceil \log \left\lceil \frac{n-k}{k} \right\rceil \right\rceil$. Recall that $|\mathcal{C}_1| \leq 2k-1$. Therefore, $|\mathcal{C}_1| + |\mathcal{C}_2| \leq 2k - 1 + \text{cub}(G)$ $\leq \text{cub}(G)\left(\frac{2k-1}{\text{cub}(G)} + 1\right) \leq \text{cub}(G)\left(\frac{2k-1}{\lceil \log \lceil \frac{n-k}{k} \rceil \rceil} + 1\right) \leq \text{cub}(G)(1+\epsilon)$.

The total running time of this algorithm is $2^{O\left(\frac{1}{\epsilon} k^3 2^{\frac{4k}{\epsilon}}\right)} n^{O(1)}$. □

4.7 Conclusion and open problems

Among the several parameters giving FPT approximations for boxicity, we know the existence of exact FPT algorithms with parameter $MVC(G)$ only. The FPT status of the problem with other parameters is still open. Our

FPT approximation algorithms for boxicity are dependent on the fact that intervals can be of different lengths. Hence, we do not know of a direct way of producing similar FPT approximation algorithms for cubicity. It will be interesting to investigate the possibility of FPT algorithms or approximations for cubicity, with parameters smaller than $MVC(G)$. We have presented $o(n)$ factor approximation algorithms for computing the boxicity and cubicity of graphs. Using these algorithms, we also derived a $o(n)$ factor approximation algorithm for computing the partial order dimension of finite posets and a $o(n)$ factor approximation algorithm for computing the threshold dimension of split graphs. To our knowledge, for none of these problems polynomial time sublinear factor approximation algorithms were known previously. Since polynomial time approximations within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$ is considered unlikely for any of these problems, no significant improvement in the approximation factor can be expected.

Chapter 5

Planar grid-drawings of outerplanar graphs

Given a connected outerplanar graph G of pathwidth p, we[1] give an algorithm to add edges to G to get a supergraph of G, which is 2-vertex-connected, outerplanar and of pathwidth $O(p)$. This settles an open problem raised by Biedl [14], in the context of computing minimum height planar straight line drawings of outerplanar graphs, with their vertices placed on a two dimensional grid. In conjunction with the result of this chapter, the constant factor approximation algorithm for this problem obtained by Biedl [14] for 2-vertex-connected outerplanar graphs will work for all outerplanar graphs.

5.1 Introduction

A graph $G(V, E)$ is outerplanar, if it has a planar embedding with all its vertices lying on the outer face. Computing planar straight line drawings of planar graphs, with their vertices placed on a two dimensional grid, is a well known problem in graph drawing. Any planar graph on n vertices can be drawn on an $(n-1) \times (n-1)$ sized grid [82]. The height of a grid is defined as the smaller of the two dimensions of the grid. If G has a planar straight line drawing, with its vertices placed on a two dimensional grid of height h, then we call it a planar drawing of G of height h. The optimization problem of minimizing the height of the planar drawing is well studied in literature.

Pathwidth is a structural parameter of graphs, which is widely used in graph drawing and layout problems [14, 42, 88]. We use $\mathrm{pw}(G)$ to denote the pathwidth of a graph G. The study of pathwidth, in the context of graph

[1] Joint work with Manu Basavaraju, L. Sunil Chandran, Deepak Rajendraprasad and Naveen Sivadasan. This work was presented in COCOON 2013.

drawings, was initiated by Dujmovic et al. [42]. It is known that any planar graph that has a planar drawing of height h has pathwidth at most h [88]. However, there exist planar graphs of constant pathwidth but requiring $\Omega(n)$ height in any planar drawing [13]. In the special case of trees, Suderman [88] showed that any tree T has a planar drawing of height at most $3\,\mathrm{pw}(T) - 1$. Biedl [14] considered the same problem for the bigger class of outerplanar graphs. For any 2-vertex-connected outerplanar graph G, Biedl [14] obtained an algorithm to compute a planar drawing of G of height at most $4\,\mathrm{pw}(G) - 3$. Since it is known that pathwidth is a lower bound for the height of the drawing [88], the algorithm given by Biedl [14] is a 4-factor approximation algorithm for the problem, for any 2-vertex-connected outerplanar graph. The method in Biedl [14] is to add edges to the 2-vertex-connected outerplanar graph G to make it a maximal outerplanar graph H and then draw H on a grid of height $4\,\mathrm{pw}(G) - 3$. The same method would give a constant factor approximation algorithm for arbitrary outerplanar graphs, if it is possible to add edges to an arbitrary connected outerplanar graph G to obtain a 2-vertex-connected outerplanar graph G' such that $\mathrm{pw}(G') = O(\mathrm{pw}(G))$. This was an open problem in Biedl [14].

In this chapter, we settle this problem by giving an algorithm to augment a connected outerplanar graph G of pathwidth p by adding edges so that the resultant graph is a 2-vertex-connected outerplanar graph of pathwidth $O(p)$. Notice that, the non-triviality lies in the fact that G' has to be maintained outerplanar. (If we relax this condition, the problem becomes very easy. It is easy to verify that the supergraph G' of G, obtained by making two arbitrarily chosen vertices of G adjacent to each other and to every other vertex in the graph, is 2-vertex-connected and has pathwidth at most $\mathrm{pw}(G) + 2$.) Similar problems of augmenting outerplanar graphs to make them 2-vertex-connected, while maintaining the outerplanarity and optimizing some other properties, like number of edges added [53, 61], have also been investigated previously.

5.2 Background

A *tree decomposition* of a graph $G(V, E)$ [79] is a pair $(T, \mathscr{X})$, where T is a tree and $\mathscr{X} = (X_t : t \in V(T))$ is a family of subsets of $V(G)$, such that:

1. $\bigcup(X_t : t \in V(T)) = V(G)$.

2. For every edge e of G there exists $t \in V(T)$ such that e has both its end points in X_t.

3. For every vertex $v \in V$, the induced subgraph of T on the vertex set $\{t \in V(T) : v \in X_t\}$ is connected.

The width of the tree decomposition is $\max_{t\in V(T)}(|X_t|-1)$. Each $X_t \in \mathscr{X}$ is referred to as a bag in the tree decomposition. A graph G has *treewidth* w if w is the minimum integer such that G has a tree decomposition of width w.

A *path decomposition* $(P, \mathscr{X})$ of a graph G is a tree decomposition of G with the additional property that the tree P is a path. The width of the path decomposition is $\max_{t\in V(P)}(|X_t|-1)$. A graph G has *pathwidth* w if w is the minimum integer such that G has a path decomposition of width w.

Without loss of generality we can assume that, in any path decomposition $(\mathcal{P}, \mathscr{X})$ of G, the vertices of the path $\mathcal{P}$ are labeled as $1, 2, \ldots$, in the order in which they appear in $\mathcal{P}$. Accordingly, the bags in $\mathscr{X}$ also get indexed as $1, 2, \ldots$. For each vertex $v \in V(G)$, define $FirstIndex_{\mathscr{X}}(v) = \min\{i \mid X_i \in \mathscr{X} \text{ contains } v\}$, $LastIndex_{\mathscr{X}}(v) = \max\{i \mid X_i \in \mathscr{X} \text{ contains } v\}$ and $Range_{\mathscr{X}}(v) = [FirstIndex_{\mathscr{X}}(v), LastIndex_{\mathscr{X}}(v)]$. By the definition of a path decomposition, if $t \in Range_{\mathscr{X}}(v)$, then $v \in X_t$. If v_1 and v_2 are two distinct vertices, define $Gap_{\mathscr{X}}(v_1, v_2)$ as follows:

- If $Range_{\mathscr{X}}(v_1) \cap Range_{\mathscr{X}}(v_2) \neq \emptyset$, then $Gap_{\mathscr{X}}(v_1, v_2) = \emptyset$.
- If $LastIndex_{\mathscr{X}}(v_1) < FirstIndex_{\mathscr{X}}(v_2)$, then $Gap_{\mathscr{X}}(v_1, v_2) = [LastIndex_{\mathscr{X}}(v_1) + 1, FirstIndex_{\mathscr{X}}(v_2)]$.
- If $LastIndex_{\mathscr{X}}(v_2) < FirstIndex_{\mathscr{X}}(v_1)$, then $Gap_{\mathscr{X}}(v_1, v_2) = [LastIndex_{\mathscr{X}}(v_2) + 1, FirstIndex_{\mathscr{X}}(v_1)]$.

The motivation for this definition is the following. Suppose $(\mathcal{P}, \mathscr{X})$ is a path decomposition of a graph G and v_1 and v_2 are two non-adjacent vertices of G. If we add a new edge between v_1 and v_2, a natural way to modify the path decomposition to reflect this edge addition is the following. If $Gap_{\mathscr{X}}(v_1, v_2) = \emptyset$, there is already an $X_t \in \mathscr{X}$, which contains v_1 and v_2 together and hence, we need not modify the path decomposition. If $LastIndex_{\mathscr{X}}(v_1) < FirstIndex_{\mathscr{X}}(v_2)$, we insert v_1 into all $X_t \in \mathscr{X}$, such that $t \in Gap_{\mathscr{X}}(v_1, v_2)$. On the other hand, if $LastIndex_{\mathscr{X}}(v_2) < FirstIndex_{\mathscr{X}}(v_1)$, we insert v_2 to all $X_t \in \mathscr{X}$, such that $t \in Gap_{\mathscr{X}}(v_1, v_2)$. It is clear from the definition of $Gap_{\mathscr{X}}(v_1, v_2)$ that this procedure gives a path decomposition of the new graph. Whenever we add an edge (v_1, v_2), we stick to this procedure to update the path decomposition.

A *block* of a connected graph G is a maximal connected subgraph of G without a cut vertex. Every block of a connected graph G is thus either a single edge which is a bridge in G, or a maximal 2-vertex-connected subgraph of G. If a block of G is not a single edge, we call it a non-trivial block of G.

Given a connected outerplanar graph G, we define a rooted tree T (hereafter referred to as the *rooted block tree* of G) as follows: The blocks of G and the cut-vertices of G form the vertex set of T. A vertex of T corresponding to a cut-vertex x of G is made adjacent to a vertex of T corresponding to a block B of G if and only if x is a vertex belonging to block B in G. The root of T

is defined to be an arbitrary block of G which contains at least one non-cut vertex (it is easy to see that such a block always exists). It is easy to see that T, as defined above, is a tree [40]. In our discussions, we restrict ourselves to a fixed rooted block tree of G and all the definitions hereafter will be with respect to this chosen tree. For any two distinct blocks B_i and B_j of G sharing a cut vertex x in G, if the vertex in the rooted block tree T of G corresponding to B_j is on the path between the root of T and the vertex in T corresponding to B_i, we say that B_i is a child block of B_j at x.

It is known that every 2-vertex-connected outerplanar graph has a unique Hamiltonian cycle [90]. Though the Hamiltonian cycle of a 2-vertex-connected block of G can be traversed either clockwise or anticlockwise, let us fix one of these orderings, so that the **successor** and **predecessor** of each vertex in the Hamiltonian cycle in a block is fixed. We call this order the clockwise order. Consider a non-root block B_i of G such that B_i is a child block of its parent, at the cut vertex x. If B_i is a non-trivial block and y_i and y_i' respectively are the predecessor and successor of x in the Hamiltonian cycle of B_i, then we call y_i the last vertex of B_i and y_i' the first vertex of B_i. If B_i is a trivial block, the sole neighbor of x in B_i is regarded as both the first vertex and the last vertex of B_i. By the term **path**, we always mean a simple path, i.e., a path in which no vertex repeats.

5.3 An overview of our method

Given a connected outerplanar graph $G(V, E)$ of pathwidth p, our algorithm will produce a 2-vertex-connected outerplanar graph $G''(V, E'')$ of pathwidth $O(p)$, where $E \subseteq E''$. Our algorithm proceeds in three stages.

(1) We use a modified version of the algorithm proposed by Govindan et al. [56] to obtain a *nice tree decomposition* (defined in Section 5.4) of G. Using this nice tree decomposition of G, we construct a special path decomposition of G of width at most $4p + 3$.

(2) For each cut vertex x of G, we define an ordering among the child blocks attached through x to their parent block. To define this ordering, we use the special path decomposition constructed in the first stage. This ordering helps us in constructing an outerplanar supergraph $G'(V, E')$ of G, whose pathwidth is at most $8p+7$, such that for every cut vertex x in G', $G' \setminus x$ has exactly two components. The properties of the special path decomposition of G obtained in the first stage is crucially used in our argument to bound the width of the path decomposition of G', produced in the second stage.

(3) We 2-vertex-connect G' to construct $G''(V, E'')$, using a straightforward algorithm. As a by-product, this algorithm also gives us a surjective mapping from the cut vertices of G' to the edges in $E'' \setminus E'$. We give a counting argument based on this mapping and some basic properties of path decompositions to

show that the width of the path decomposition of G'' produced in the third stage is at most $16p + 15$.

5.4 Stage 1: Construct a nice path decomposition of G

In this section, we construct a *nice tree decomposition* of the connected outerplanar graph G and then use it to construct a *nice path decomposition* of G. We begin by giving the definition of a nice tree decomposition.

Given an outerplanar graph G, Govindan et al. [56, Section 2] gave a linear time algorithm to construct a width 2 tree decomposition $(T, \mathscr{Y})$ of G where $\mathscr{Y} = (Y_t : t \in V(T))$, with the following special properties:

1. There is a bijective mapping b from $V(G)$ to $V(T)$ such that, for each $v \in V(G)$, v is present in the bag $Y_{b(v)}$.

2. If B_i is a child block of B_j at a cut vertex x, the vertex set $\{b(v) \mid v \in V(B_i \setminus x)\}$ induces a subtree T' of T such that, if $(b(u), b(v))$ is an edge in T', then $(u, v) \in E(G)$ - this means that the subgraph $B_i \setminus x$ of G has a spanning tree, which is a copy of T' on the corresponding vertices. Moreover, $(T', \mathscr{Y}')$, with $\mathscr{Y}' = (Y_t : t \in V(T'))$ gives a tree decomposition of B_i.

3. G has a spanning tree, which is a copy of T on the corresponding vertices; i.e. if $(b(u), b(v))$ is an edge in T, then $(u, v) \in E(G)$.

Definition 5.1 (Nice tree decomposition of an outerplanar graph G)**.** A tree decomposition $(T, \mathscr{Y})$ of G, where $\mathscr{Y} = (Y_t : t \in V(T))$ having properties 1, 2 and 3 above, together with the following additional property, is called a nice tree decomposition of G.

4. If y_i and y'_i are respectively the last and first vertices of a non-root, non-trivial block B_i, then the bag $Y_{b(y_i)} \in \mathscr{Y}$ contains both y_i and y'_i.

In the discussion that follows, we will show that any outerplanar graph G has a nice tree decomposition $(T, \mathscr{Y})$ of width at most 3. Initialize $(T, \mathscr{Y})$ to be the tree decomposition of G, constructed using the method proposed by Govindan et al. [56], satisfying properties 1, 2 and 3 of nice tree decompositions. We need to modify $(T, \mathscr{Y})$ in such a way that, it satisfies property 4 as well.

For every non-root, non-trivial block B_i of G, do the following. If y_i and y'_i are respectively the last and first vertices of B_i, then, for each $t \in \{b(v) \mid v \in V(B_i \setminus x)\}$, we insert y'_i to Y_t, if it is not already present in Y_t and we call y'_i as a *propagated* vertex. Note that, after this modification $Y_{b(y_i)}$ contains both y_i and y'_i.

Claim 5.0.2. *After the modification, $(T, \mathscr{Y})$ remains a tree decomposition of G.*

Proof. Clearly, we only need to verify that the third property in the definition of a tree decomposition holds, for all the propagated vertices. Let y_i' be a propagated vertex, which got inserted to the bags corresponding to vertices of $B_i \setminus x$, during the modification. Let $V_{y_i'} = \{t \mid y_i' \in Y_t$, before the modification$\}$ and let $V_{y_i'}' = \{t \mid y_i' \in Y_t$, after the modification$\}$. Then, clearly, $V_{y_i'}' = V_{y_i'} \cup \{b(v) \mid v \in V(B_i \setminus x)\}$.

Clearly, the induced subgraph of T on the vertex set $V_{y_i'}$ is connected, since we had a tree decomposition of G before the modification. By property 2 of nice decompositions, the induced subgraph of T on the vertex set $\{b(v) \mid v \in V(B_i \setminus x)\}$ is also connected. Moreover, by property 1 of nice decompositions, $b(y_i') \in V_{y_i'}$ and hence, $b(y_i') \in \{b(v) \mid v \in V(B_i \setminus x)\} \cap V_{y_i'}$. This implies that the induced subgraph of T on the vertex set $V_{y_i'}'$ is connected. □

Claim 5.0.3. *After the modification, $(T, \mathscr{Y})$ becomes a nice tree decomposition of G of width at most* 3.

Proof. It is easy to verify that all the four properties required by nice decompositions are satisfied, after the modification. Moreover, for any block B_i, attached to its parent at the cut vertex x, at most one (propagated) vertex is getting newly inserted into the bags corresponding to vertices of $B_i \setminus x$. Since the size of any bag in $\mathscr{Y}$ was at most three initially and it got increased by at most one, in the new decomposition the size of any bag is at most four. Therefore, the new decomposition has width at most three. □

From the claims above, we can conclude the following.

Lemma 5.1. *Every outerplanar graph G has a nice tree decomposition $(T, \mathscr{Y})$ of width* 3, *constructible in polynomial time.*

Definition 5.2 (Nice path decomposition of an outerplanar graph)**.** Let $(\mathcal{P}, \mathscr{X})$ be a path decomposition of an outerplanar graph G. If, for every non-root non-trivial block B_i, there is a bag $X_t \in \mathscr{X}$ containing both the first and last vertices of B_i together, then $(\mathcal{P}, \mathscr{X})$ is called a nice path decomposition of G.

Lemma 5.2. *Let G be an outerplanar graph with $\mathrm{pw}(G) = p$. A nice path decomposition $(\mathcal{P}, \mathscr{X})$ of G, of width at most $4p + 3$, is constructible in polynomial time.*

Proof. Let $(T, \mathscr{Y})$, with $\mathscr{Y} = (Y_{v_T} : v_T \in V(T))$ be a nice tree decomposition of G of width 3, obtained using Lemma 5.1. Obtain an optimal path decomposition $(\mathcal{P}_T, \mathscr{X}_T)$ of the tree T in polynomial time, using a standard algorithm (For example, the algorithm from [85]). Since T is a spanning tree of G, the pathwidth of T is at most that of G. Therefore, the width of the path decomposition $(\mathcal{P}_T, \mathscr{X}_T)$ is at most p; i.e. there are at most $p+1$ vertices of T in each bag $X_{T_i} \in \mathscr{X}_T$.

Let $\mathcal{P} = \mathcal{P}_T$ and for each $X_{T_i} \in \mathscr{X}_T$, we define $X_i = \bigcup_{v_T \in X_{T_i}} Y_{v_T}$. It is not difficult to show that $(\mathcal{P}, \mathscr{X})$, with $\mathscr{X} = (X_1, \ldots, X_{|V(\mathcal{P}_T)|})$, is a path decomposition of G (See [56]). The width of this path decomposition is at most $4(p+1)-1 = 4p+3$, since $|Y_{v_T}| \leq 4$, for each bag $Y_{v_T} \in \mathscr{Y}$ and $|X_{T_i}| \leq p+1$, for each bag $X_{T_i} \in \mathscr{X}_T$.

Let B_i be a non-root, non-trivial block in G and y_i and y'_i respectively be the first and last vertices of B_i. Since $b(y_i)$ is a vertex of the tree T, there is some bag $X_{T_j} \in \mathscr{X}_T$, containing $b(y_i)$. The bag $Y_{b(y_i)} \in \mathscr{Y}$ contains both y_i and y'_i, since $(T, \mathscr{Y})$ is a nice tree decomposition of G. It follows from the definition of X_j that $X_j \in \mathscr{X}$ contains both y_i and y'_i. Therefore, $(\mathcal{P}, \mathscr{X})$ is a nice path decomposition of G. $\square$

5.5 Edge addition without spoiling the outerplanarity

In this section, we prove some technical lemmas which will be later used to prove that the intermediate graph G' obtained in Stage 2 and the 2-vertex-connected graph G'' obtained in Stage 3 are outerplanar.

The following is a simple observation about 2-vertex-connected outerplanar graphs.

Observation 5.1. *Let H be a 2-vertex-connected outerplanar graph. Then, the number of internally vertex disjoint paths in H between any two consecutive vertices in the Hamiltonian cycle of H is exactly two.*

Proof. Since H is a 2-vertex-connected outerplanar graph, it can be embedded in the plane, so that its exterior cycle C is the unique Hamiltonian cycle of H [30]. Consider such an embedding of H and let $C = (v_1, v_2, \ldots, v_n, v_1)$, where the vertices of the cycle C are given in the clockwise order of the cycle. Consider any pair of of consecutive vertices in C. Without loss of generality, let (v_1, v_2) be this pair. The paths $P_1 = (v_1, v_2)$ and $P_2 = (v_1, v_n, v_{n-1}, \ldots, v_3, v_2)$ are obviously two internally vertex disjoint paths in H, between v_1 and v_2.

Since the path $P_1 = (v_1, v_2)$ is internally vertex disjoint from any other path in H between v_1 and v_2, it is enough to show that, there cannot be two internally vertex disjoint paths P_i and P_j between v_1 and v_2 without using the edge (v_1, v_2). For contradiction, assume that P_i and P_j are two internally vertex disjoint paths between v_1 and v_2 without using the edge (v_1, v_2). Let (v_1, v_i) be the first edge of P_i and (v_1, v_j) be the first edge of P_j. Without loss of generality, assume that $i < j$. This implies that the edge (v_1, v_i) is not an edge of the exterior cycle C and hence, the (curve corresponding to the) edge

(v_1, v_i) splits the region bounded by C into two parts. Let the closed region bounded by the path $v_i, v_{i+1}, \ldots, v_n, v_1$ and the edge (v_1, v_i) be denoted by C_l and the closed region bounded by by the path $v_i, v_{i-1}, \ldots, v_1$ and the edge (v_1, v_i) be denoted by C_r.

Let the subpath of P_j from v_j to v_2 be denoted by P'_j. Since v_j is in $C_l \setminus C_r$ and v_2 is in $C_r \setminus C_l$, the path P'_j has to cross from C_l to C_r at least once. Since P_j is vertex disjoint from P_i, the path P'_j cannot cross from C_l to C_r at v_i. Since the path P_j is simple, P'_j cannot cross from C_l to C_r at v_1 also. This implies that there is an edge (u, v) in P'_j with u belonging to $C_l \setminus C_r$ and v belonging to $C_r \setminus C_l$. This would mean that the curve corresponding to the edge (u, v) will cross the curve corresponding to the edge (v_1, v_i), which is a contradiction, because by our assumption, our embedding is an outerplanar embedding. Therefore, there cannot be two internally vertex disjoint paths P_i and P_j between v_1 and v_2 without using the edge (v_1, v_2).

Thus, the number of internally vertex disjoint paths in H between any two consecutive vertices in the Hamiltonian cycle of H is exactly two. □

The following lemma describes some conditions to ensure that the outerplanarity of a graph is not spoiled on the addition of a new edge. To get an intuitive understanding of this lemma, the reader may refer to Figure 5.1. Recall that, when we use the term *path*, it always refers to a simple path.

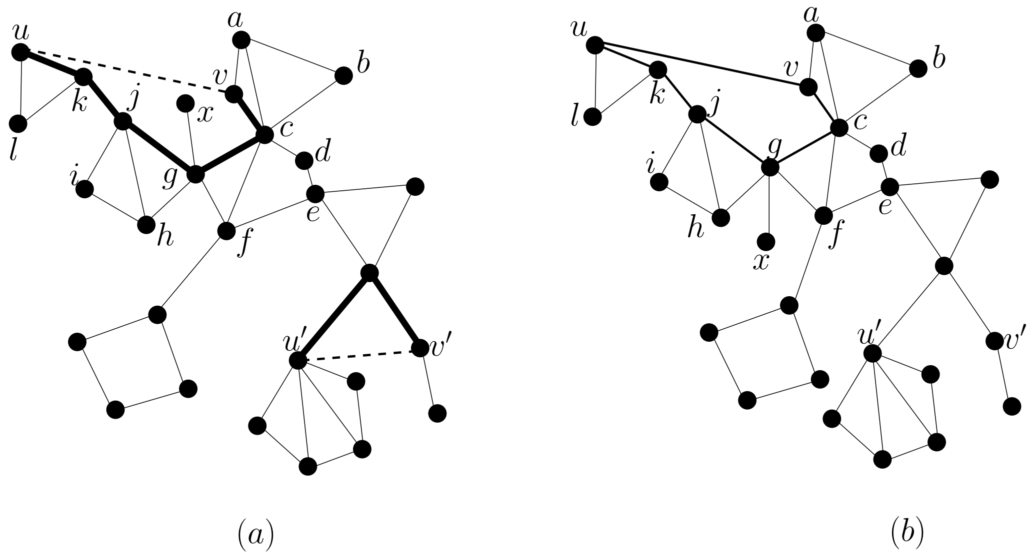


Figure 5.1: (a) The path between u and v and the path between u' and v' (shown in thick edges) satisfy the conditions stated in Lemma 5.3. According to Lemma 5.3, on adding any one of the dotted edges (u, v) or (u', v'), the resultant graph is outerplanar. (b) An outerplanar drawing of the resultant graph, after adding the edge (u, v). In this graph, $u, v, a, b, c, d, e, f, g, h, i, j, k, l, u$ is the Hamiltonian cycle of the new block formed.

Lemma 5.3. *Let $G(V, E)$ be a connected outerplanar graph. Let u and v be two distinct non-adjacent vertices in G and let $P = (u = x_0, x_1, x_2, \ldots, x_k, x_{k+1} = v)$ where $k \geq 1$ be a path in G such that:*

(i) *P shares at most one edge with any block of G.*

(ii) *For $0 \leq i \leq k$, if the block containing the edge (x_i, x_{i+1}) is non-trivial, then x_{i+1} is the successor of x_i in the Hamiltonian cycle of that block.*

Then the graph $G'(V, E')$, where $E' = E \cup \{(u, v)\}$, is outerplanar.

Proof. It is well known that a graph G is outerplanar if and only if it contains no subgraph that is a subdivision of K_4 or $K_{2,3}$ [30]. Consider a path P between u and v as stated in the lemma.

Property 5.1. *In every path in G from u to v, vertices $x_1, \ldots, x_k$ should appear and for $0 \leq i \leq k$, x_i should appear before x_{i+1} in any such path.*

Proof. For any $1 \leq i \leq k$, the two consecutive edges $e_i = (x_{i-1}, x_i)$ and $e_{i+1} = (x_i, x_{i+1})$ of the path P belong to two different blocks of G, by assumption. Therefore, each internal vertex x_i, $1 \leq i \leq k$, is a cut vertex in G. As a result, in every path in G between u and v, vertices $x_1, \ldots, x_k$ should appear and for $0 \leq i \leq k$, x_i should appear before x_{i+1} in any such path. □

Property 5.2. *For any $0 \leq i \leq k$, there are at most two internally vertex disjoint paths in G between x_i and x_{i+1}.*

Proof. Any path from x_i to x_{i+1} lies fully inside the block B_i that contains the edge (x_i, x_{i+1}). If B_i is trivial, the only path from x_i to x_{i+1} is the direct edge between them.

If this is not the case, B_i is 2-vertex-connected. Since this means that B_i is non-trivial, by the assumption of Lemma 5.3 x_{i+1} is the successor of x_i in the Hamiltonian cycle of B_i. Therefore, by Observation 5.1 the property follows. □

We will show that if G' is not outerplanar, then G also was not outerplanar, which is a contradiction. Assume that G' is not outerplanar. This implies that there is a subgraph H' of G' that is a subdivision of K_4 or $K_{2,3}$. Since G does not have a subgraph that is a subdivision of K_4 or $K_{2,3}$, H' cannot be a subgraph of G. Hence, the new edge (u, v) should be an edge in H' and all other edges of H' are edges of G.

Case 1. H' is a subdivision of K_4.

Let k_1, k_2, k_3 and k_4 denote the four vertices of H' that correspond to the vertices of K_4. We call them as branch vertices of H'. For $i, j \in \{1, 2, 3, 4\}$, $i \neq j$, let $P_{i,j}$ denote the path in H' from the branch vertex k_i to the branch vertex k_j, such that each intermediate vertex of the path is a degree two vertex in H'. Without loss of generality, assume that the edge (u, v) is part of the path $P_{1,2}$ of H'.

Claim 5.3.1. *All of the vertices $x_1, \ldots, x_k$ appear in $P_{1,2}$. The order $<$ in which the vertices u, v, $x_1, \ldots, x_k$ appear in $P_{1,2}$ should be one of the following three: (without loss of generality, assuming $u < v$): (1) $u < v < x_k < x_{k-1} < \cdots < x_1$ (2) $x_k < x_{k-1} < \cdots < x_1 < u < v$ (3) $x_j < x_{j-1} < \cdots < x_1 < u < v < x_k \cdots < x_{j+1}$ for some $j \in \{1, 2, \ldots, k-1\}$.*

Proof. Suppose x_i, $1 \leq i \leq k$, does not belong to the path $P_{1,2}$. Then, there is a path in $H' \setminus (u, v)$ between vertices u and v, avoiding the vertex x_i, since H' is a subdivision of K_4. Since $H' \setminus (u, v)$ is a subgraph of G, this implies that there is a path in G, between u and v that avoids x_i. This is a contradiction to Property 5.1. Therefore, $x_i \in P_{1,2}$. Notice that there is a path in $H' \setminus (u, v)$, and hence in G, between u and v that goes through the vertex k_3. To satisfy Property 5.1, x_i should appear before x_{i+1}, for $0 \leq i \leq k$, in this path. Hence, one of the orderings mentioned in the claim should happen in $P_{1,2}$. □

Let us denote the first vertex in the ordering $<$ by z_1 and the last vertex in the ordering $<$ by z_2. (In the first case, $z_1 = u$ and $z_2 = x_1$. In the second case, $z_1 = x_k$ and $z_2 = v$. In the third case, $z_1 = x_j$ and $z_2 = x_{j+1}$.) In all the three cases of the ordering, there is a direct edge in G, between z_1 and z_2. Notice that in any of these three possible orderings, we do not have $z_1 = u$ and $z_2 = v$ simultaneously. Since $(z_1, z_2) \neq (u, v)$, by deleting the intermediate vertices between z_1 and z_2 from the path $P_{1,2}$ and including the direct edge between z_1 and z_2, we get a path $P'_{1,2}$ between k_1 and k_2 in G. All vertices in $P'_{1,2}$ are from the vertex set of $P_{1,2}$. Therefore, by replacing the path $P_{1,2}$ in H' by $P'_{1,2}$, we get a subgraph H of G that is a subdivision of K_4. This means that G is not outerplanar, which is a contradiction. Therefore, H' cannot be a subdivision of K_4.

Case 2. H' is a subdivision of $K_{2,3}$.

As earlier, let k_1, k_2, k_3, k_4 and k_5 denote the branch vertices of H' that correspond to the vertices of $K_{2,3}$. Let k_1, k_3, k_5 be the degree 2 branch vertices in H' and k_2, k_4 be the degree 3 branch vertices of H'. For $i \in \{1, 3, 5\}$ and $j \in \{2, 4\}$, let $P_{i,j}$ denote the path in H' from vertex k_i to vertex k_j, such that each intermediate vertex of the path is a degree two vertex in H'. Also, for $i \in \{1, 3, 5\}$ and $j \in \{2, 4\}$ let $P_{j,i}$ denote the path from j to i in which the vertices in $P_{j,i}$ appear in the reverse order compared to $P_{i,j}$. Without loss of generality, assume that the edge (u, v) is part of the path $P_{1,2}$ of H'. Let $P_{4,1,2}$ denote the path in H' between vertices k_4 and k_2, obtained by concatenating the paths $P_{4,1}$ and $P_{1,2}$.

Claim 5.3.2. *All of the vertices $x_1, \ldots, x_k$ appear in $P_{4,1,2}$. The order $<$ in which the vertices u, v, $x_1, \ldots, x_k$ appear in $P_{4,1,2}$ should be one of the following three (without loss of generality, assuming $u < v$): (1) $u < v < x_k < x_{k-1} < \cdots < x_1$ (2) $x_k < x_{k-1} < \cdots < x_1 < u < v$ (3) $x_j < x_{j-1} < \cdots < x_1 < u < v < x_k \cdots < x_{j+1}$ for some $j \in \{1, 2, \ldots, k-1\}$.*

This can be proved in a similar way as in Case 1. The remaining part of the proof is also similar. Let us denote the first vertex in the ordering $<$ by z_1 and the last vertex in the ordering $<$ by z_2. Repeating similar arguments as in Case 1, we can prove that by deleting the intermediate vertices between z_1 and z_2 from the path $P_{4,1,2}$ and including the direct edge between z_1 and z_2, we get a path $P'_{4,1,2}$ between k_4 and k_2 in G. All vertices in $P'_{4,1,2}$ are from the vertex set of $P_{4,1,2}$. Therefore, by replacing the path $P_{4,1,2}$ in H' by $P'_{4,1,2}$, we get a subgraph H of G. If $P'_{4,1,2}$ has at least one intermediate vertex, the subgraph H of G, obtained by replacing the path $P_{4,1,2}$ in H' by $P'_{4,1,2}$, is a subdivision of $K_{2,3}$, where an intermediate vertex of $P'_{4,1,2}$ takes the role of the branch vertex k_1. This contradicts the assumption that G is outerplanar.

Therefore, assume that $P'_{4,1,2}$ has no intermediate vertices, i. e., $z_1 = k_4$ and $z_2 = k_2$. In the first case of ordering $<$ mentioned in the claim above, we have $k_4 = z_1 = u = x_0$ and $k_2 = z_2 = x_1$. In the second case, $k_4 = z_1 = x_k$ and $k_2 = z_2 = v = x_{k+1}$. In the third case, $k_4 = z_1 = x_j$ and $k_2 = z_2 = x_{j+1}$. In each of these cases, by Property 5.2, there can be at most two vertex disjoint paths in G between z_1 and z_2. But, in all these cases, there is a direct edge between $k_4 = z_1$ and $k_2 = z_2$ in G. Since H' is a subdivision of $K_{2,3}$, other than this direct edge, in $H' \setminus (u, v)$ there are two other paths from $k_4 = z_1$ to $k_2 = z_2$ that are internally vertex disjoint and containing at least one intermediate vertex. This will mean that there are at least three internally vertex disjoint paths from $z_1 = k_4$ to $z_2 = k_2$ in G, which is a contradiction. Therefore, H' cannot be a subdivision of $K_{2,3}$.

Since, G' does not contain a subgraph H' that is a subdivision of K_4 or $K_{2,3}$, G' is outerplanar. □

The following lemma explains the effect of the addition of an edge (u, v) as mentioned in Lemma 5.3, to the block structure and the Hamiltonian cycle of each block. Assume that for $0 \le i \le k$, the edge (x_i, x_{i+1}) belongs to the block B_i.

Lemma 5.4.

1. *Other than the blocks B_0 to B_k of G merging together to form a new block B' of G', blocks in G and G' are the same.*

2. *Vertices in blocks B_0 to B_k, except x_i, $0 \le i \le k+1$, retains their successor and predecessor in the Hamiltonian cycle of B' same as it was in its respective block's Hamiltonian cycle in G.*

3. *Each x_i, $0 \leq i \leq k$, retains its Hamiltonian cycle predecessor in B' same as it was in the block B_i of G and each x_i, $1 \leq i \leq k+1$, retains its Hamiltonian cycle successor in B' same as in the block B_{i-1} of G.*

Proof. When the edge (u, v) is added, it creates a cycle containing the vertices $u = x_0, x_1, \ldots, x_{k+1} = v$. Hence, the blocks B_0 to B_k of G merge together to form a single block B' in G'. It is obvious that other blocks are unaffected by this edge addition.

For simplicity, if B_i is a trivial block containing the edge (x_i, x_{i+1}), we say that x_i and x_{i+1} are neighbors of each other in the Hamiltonian cycle of B_i. For each B_i, $0 \leq i \leq k$, let $x_i, x_{i+1}, z_{i1}, z_{i2}, \ldots, z_{it_i}, x_i$ be the Hamiltonian cycle of B_i in G. For $0 \leq i \leq k$, let us denote the path $x_{i+1}, z_{i1}, z_{i2}, \ldots, z_{it_i}$ by P_i. Then, the Hamiltonian cycle of B' is $u = x_0 \circ P_k \circ P_{k-1} \circ \ldots P_0 \circ u$, where $\circ$ denotes the concatenation of the paths. (For example, in Figure 5.1, $u, v, a, b, c, d, e, f, g, h, i, j, k, l, u$ is the Hamiltonian cycle of the new block formed, when the edge (u, v) is added.) From this, we can conclude that the second and third parts of the lemma holds. □

5.6 Stage 2: Construction of G' and its path decomposition

The organization of this sections is as follows: For each cut vertex x of G, we define an ordering among the child blocks attached through x to their parent block, based on the nice path decomposition $(\mathcal{P}, \mathscr{X})$ of G obtained using Lemma 5.2. This ordering is then used in defining a supergraph $G'(V, E')$ of G such that for every cut vertex x in G', $G' \backslash x$ has exactly two components. Using repeated applications of Lemma 5.3, we then show that G' is outerplanar. We extend the path decomposition $(\mathcal{P}, \mathscr{X})$ of G to a path decomposition $(\mathcal{P}', \mathscr{X}')$ of G', as described in Section 5.2. By a counting argument using the properties of the nice path decomposition $(\mathcal{P}, \mathscr{X})$, we show that the width of the path decomposition $(\mathcal{P}', \mathscr{X}')$ is at most $2p' + 1$, where p' is the width of $(\mathcal{P}, \mathscr{X})$.

5.6.1 Defining an ordering of child blocks

If $(\mathcal{P}, \mathscr{X})$ is a nice path decomposition of G, then, for each non-root block B of G, at least one bag in $\mathscr{X}$ contains both the first and last vertices of B together.

Definition 5.3 (Sequence number of a non-root block)**.** Let $(\mathcal{P}, \mathscr{X})$ be the nice path decomposition of G obtained using Lemma 5.2. For each non-root block B of G, we define the sequence number of B as $\min\{i \mid X_i \in \mathscr{X}$ simultaneously contains both the first and last vertices of $B\}$.

For each cut vertex x, there is a unique block B^x such that B^x and its child blocks are intersecting at x. For each cut vertex x, we define an ordering among the child blocks attached at x, as follows. If $B_1, \ldots, B_k$ are the child blocks attached at x, we order them in the increasing order of their sequence numbers in $(\mathcal{P}, \mathscr{X})$. If B_i and B_j are two child blocks with the same sequence number, their relative ordering is arbitrary.

From the ordering defined, we can make some observations about the appearance of the first and last vertices of a block B_i in the path decomposition. These observations are crucially used for bounding the width of the path decomposition $(\mathcal{P}', \mathscr{X}')$ of G'. Let $B_1, \ldots, B_k$ be the child blocks attached at a cut vertex x, occurring in that order according to the ordering we defined above. For $1 \leq i \leq k$, let y_i and y'_i respectively be the last and first vertices of B_i.

Property 5.3. *For any $1 \leq i \leq k-1$, if $Gap_{\mathscr{X}}(y'_i, y_{i+1}) \neq \emptyset$, then $Gap_{\mathscr{X}}(y'_i, y_{i+1}) = [LastIndex_{\mathscr{X}}(y'_i) + 1, FirstIndex_{\mathscr{X}}(y_{i+1})]$ and for all $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$, $x \in X_t$.*

Proof. If $Gap_{\mathscr{X}}(y'_i, y_{i+1}) \neq \emptyset$, either $LastIndex_{\mathscr{X}}(y'_i) < FirstIndex_{\mathscr{X}}(y_{i+1})$ or $LastIndex_{\mathscr{X}}(y_{i+1}) < FirstIndex_{\mathscr{X}}(y'_i)$. The latter case will imply that, sequence number of B_{i+1} < sequence number of B_i, which is a contradiction. Therefore, $LastIndex_{\mathscr{X}}(y'_i) < FirstIndex_{\mathscr{X}}(y_{i+1})$ and hence $Gap_{\mathscr{X}}(y'_i, y_{i+1}) = [LastIndex_{\mathscr{X}}(y'_i) + 1, FirstIndex_{\mathscr{X}}(y_{i+1})]$.

Since x is adjacent to y'_i and y_{i+1}, we get $FirstIndex_{\mathscr{X}}(x) \leq LastIndex_{\mathscr{X}}(y'_i)$ and $LastIndex_{\mathscr{X}}(x) \geq FirstIndex_{\mathscr{X}}(y_{i+1})$. We can conclude that $Range_{\mathscr{X}}(x) \supseteq [LastIndex_{\mathscr{X}}(y'_i), FirstIndex_{\mathscr{X}}(y_{i+1})]$ and the property follows. □

Property 5.4. *For any $1 \leq i < j \leq k-1$, $Gap_{\mathscr{X}}(y'_i, y_{i+1}) \cap Gap_{\mathscr{X}}(y'_j, y_{j+1}) = \emptyset$.*

Proof. We can assume that $Gap_{\mathscr{X}}(y'_i, y_{i+1}) \neq \emptyset$ and $Gap_{\mathscr{X}}(y'_j, y_{j+1}) \neq \emptyset$, since the property holds trivially otherwise. By Property 5.3, we get, $Gap_{\mathscr{X}}(y'_i, y_{i+1}) = [LastIndex_{\mathscr{X}}(y'_i) + 1, FirstIndex_{\mathscr{X}}(y_{i+1})]$ and $Gap_{\mathscr{X}}(y'_j, y_{j+1}) = [LastIndex_{\mathscr{X}}(y'_j) + 1, FirstIndex_{\mathscr{X}}(y_{j+1})]$. Since $i + 1 \leq j$, by the property of the ordering of blocks, we know that sequence number of $B_{i+1} \leq$ sequence number of B_j. From the definitions, we have, $FirstIndex_{\mathscr{X}}(y_{i+1}) \leq$ sequence number of $B_{i+1} \leq$sequence number of $B_j \leq LastIndex_{\mathscr{X}}(y'_j)$ and the property follows. □

5.6.2 Algorithm for constructing G' and its path decomposition

We use Algorithm 3 to construct $G'(V, E')$ and a path decomposition $(\mathcal{P}', \mathscr{X}')$ of G'. The processing of each cut vertex is done in lines 2 to 7 of Algorithm 3. While processing a cut vertex x, the algorithm adds the edges

$(y'_1, y_2), (y'_2, y_3), \ldots, (y'_{k_x-1}, y_{k_x})$ (as defined in the algorithm) and modifies the path decomposition, to reflect each edge addition.

Algorithm 3: Computing the intermediate supergraph G' and its path decomposition

---

Input: A connected outerplanar graph $G(V, E)$ and a nice path decomposition $(\mathcal{P}, \mathscr{X})$ of G, the rooted block tree of G, the Hamiltonian cycle of each non-trivial block of G and the first and last vertices of each non-root block of G

Output: An outerplanar supergraph $G'(V, E')$ of G such that, for every cut vertex x of G', $G' \setminus x$ has exactly two connected components, a path decomposition $(\mathcal{P}', \mathscr{X}')$ of G'

1 $E' = E$, $(\mathcal{P}', \mathscr{X}') = (\mathcal{P}, \mathscr{X})$
2 **for** *each cut vertex* $x \in V(G)$ **do**
3   Let $B_1, \ldots, B_{k_x}$, in that order, be the child blocks attached at x, according to the ordering defined in Section 5.6.1
4   **for** $i = 1$ *to* $k_x - 1$ **do**
5   Let y'_i be the first vertex of B_i and y_{i+1} be the last vertex of B_{i+1}
6         $E' = E' \cup \{(y'_i, y_{i+1})\}$
7   **if** $Gap_{\mathscr{X}}(y'_i, y_{i+1}) \neq \emptyset$ **then for** $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$ **do**
            $X'_t = X'_t \cup \{y'_i\}$

Lemma 5.5. *G' is outerplanar and for each cut vertex x of G', $G' \setminus x$ has exactly two components.*

Proof. We know that G is outerplanar to begin with. At a certain stage, let x be the cut vertex taken up by the algorithm for processing (in Line 2). Assume that the graph at this stage, denoted by G_0, is outerplanar and each cut vertex x' whose processing is completed, satisfies the condition that all the child blocks attached at x' have merged together to form a single child block attached at x'.

It is clear that the child blocks attached at a vertex x remain unchanged until x is picked up by the algorithm for processing. Let $B_1, \ldots, B_{k_x}$, in that order, be the child blocks attached at x, according to the ordering defined in Section 5.6.1. Let B^x be the parent block of $B_1, \ldots, B_{k_x}$, in the current graph G_0. For each $1 \leq i \leq k_x$, let y'_i and y_i respectively be the first and last vertices of B_i. For $1 \leq i \leq k_x - 1$, let G_i be the graph obtained, when the algorithm has added the edges up to (y'_i, y_{i+1}).

We will prove that the algorithm maintains the following invariants, while processing the cut vertex x, for each $0 \leq i \leq k_x - 1$:

> *The graph G_i is outerplanar. In G_i, the blocks $B_1, \ldots, B_{i+1}$ of G_{i-1} have merged together and formed a child block B' of B^x. The vertex*

y'_{i+1} is the first vertex of B'. If $i \leq k_x - 2$, blocks $B_{i+2}, \ldots, B_{k_x}$ remain the same in G_i, as in G.

By our assumption, the invariants hold for G_0. We need to show that if the invariants hold for G_{i-1}, they hold for G_i as well. Assume that the invariants hold for G_{i-1} and let B' be the child block of B^x in G_{i-1} that is formed by merging together the blocks $B_1, \ldots, B_i$ of G_{i-2}, as stated in the invariant. Since the invariants hold for G_{i-1} by our assumption, y'_i is the first vertex of B' and y_{i+1} is the last vertex in B_{i+1}. In other words, y'_i is the successor of x in B' and y_{i+1} is the predecessor of x in B_{i+1} and the edges (y_{i+1}, x) and (x, y'_i) of the path $P_i = (y_{i+1}, x, y'_i)$ belong to two different blocks of G_{i-1}. Hence, by Lemma 5.3, after adding the edge (y'_i, y_{i+1}), the resultant intermediate graph G_i is outerplanar. By Lemma 5.4, the blocks B' and B_{i+1} merges together to form a child block B'' of B^x in G_i. Further, the vertex y'_{i+1} will be the successor of x in the Hamiltonian cycle of B'' i.e, the first of the block B''. Remaining blocks of G_i are the same as in G_{i-1}. Thus, all the invariants hold for G_i. It follows that the graph G_{k_x-1} is outerplanar and the blocks $B_1, \ldots, B_{k_x}$ have merged together in G_{k_x-1} to form a single child block of B^x at x.

When this processing is repeated at all cut vertices, it is clear that G' is outerplanar and for each cut vertex x of G', $G' \setminus x$ has exactly two components.

□

Lemma 5.6. *$(\mathcal{P}', \mathscr{X}')$ is a path decomposition of G' of width at most $8p + 7$.*

Proof. Algorithm 3 initialized $(\mathcal{P}', \mathscr{X}')$ to $(\mathcal{P}, \mathscr{X})$ and modified it in Line 7, following each edge addition. By Property 5.3, we have $Gap_{\mathscr{X}}(y'_i, y_{i+1}) = [LastIndex_{\mathscr{X}}(y'_i) + 1, FirstIndex_{\mathscr{X}}(y_{i+1})]$. Hence, by the modification done in Line 7 while adding a new edge (y'_i, y_{i+1}), $(\mathcal{P}', \mathscr{X}')$ becomes a path decomposition of the graph containing the edge (y'_i, y_{i+1}), as explained in Section 5.2. It follows that, when the algorithm terminates $(\mathcal{P}', \mathscr{X}')$ is a path decomposition of G'.

Consider any $X'_t \in \mathscr{X}'$. While processing the cut vertex x, if Algorithm 3 inserts a new vertex y'_i to X'_t, to reflect the addition of a new edge (y'_i, y_{i+1}) then, $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$. Suppose (y'_i, y_{i+1}) and (y'_j, y_{j+1}) are two new edges added while processing the cut vertex x, where, $1 \leq i < j \leq k_x - 1$. By Property 5.4, we know that if $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$, then, $t \notin Gap_{\mathscr{X}}(y'_j, y_{j+1})$. Therefore, when the algorithm processes a cut vertex x in lines 2 to 7, at most one vertex is newly inserted to the bag X'_t. Moreover, if $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$ then, the cut vertex $x \in X_t$, by Property 5.3.

That means, a vertex not present in the bag X_t can be added to X'_t only when a cut vertex x that is already present in the bag X_t is being processed. Moreover, when a cut vertex x that is present in X_t is processed, at most one new vertex can be added to X'_t. It follows that $|X'_t| \leq |X_t|$+number of cut vertices present in $X_t \leq 2|X_t| \leq 2(4p+4)$. Therefore, the width of the path decomposition $(\mathcal{P}', \mathscr{X}')$ is at most $8p+7$. $\square$

5.7 Construction of G'' and its path decomposition

In this section, we give an algorithm to add some more edges to $G'(V, E')$ so that the resultant graph $G''(V, E'')$ is 2-vertex-connected. The algorithm also extend the path decomposition $(\mathcal{P}', \mathscr{X}')$ of G' to a path decomposition $(\mathcal{P}'', \mathscr{X}'')$ of G''. The analysis of the algorithm shows the existence of a surjective mapping from the cut vertices of G' to the edges in $E'' \setminus E'$. A counting argument based on this surjective mapping shows that the width of the path decomposition $(\mathcal{P}'', \mathscr{X}'')$ is at most $16p+15$. For making our presentation simpler, if a block B_i is just an edge (u, v), we abuse the definition of a Hamiltonian cycle and say that u and v are clockwise neighbors of each other in the Hamiltonian cycle of B_i.

Recall that for every cut vertex x of G', $G' \setminus x$ has exactly two components. Since any cut vertex belongs to exactly two blocks of G, based on the rooted block tree structure of G, we call them the parent block containing x and the child block containing x. We use $child_x(B)$ to denote the unique child block of the block B at the cut vertex x and $parent(B)$ to denote the parent block of the block B. For a block B, $next_B(v)$ denotes the successor of the vertex v in the Hamiltonian cycle of B. We say that a vertex u is *encountered by the algorithm*, when u gets assigned to the variable v', during the execution of the algorithm. The block referred to by the variable B represents the *current block* being traversed.

To get a high level picture of our algorithm, the reader may consider it as a traversal of vertices of G', starting from a non-cut vertex in the root block of G' and proceeding to the successor of v on reaching a non-cut vertex v. On reaching a cut vertex x, the algorithm bypasses x and recursively traverses the child block containing x and its descendant blocks, starting from the successor of x in child block containing x. After this, the algorithm comes back to x to visit it, and continues the traversal of the remaining graph, by moving to the successor of x in the parent block containing x. Before starting the recursive traversal of the child block containing x and its descendant blocks, the algorithm sets $bypass(x) = TRUE$. (Note that, since there is only one child block attached to any cut vertex, each cut vertex is bypassed only once.) In this way, when a sequence of one or more cut vertices is bypassed, an edge is added from the vertex visited just before bypassing the first cut vertex in the sequence to the vertex visited just after bypassing the last cut vertex in the sequence. The path decomposition is also modified, to reflect this edge addition. The detailed algorithm to 2-vertex-connect G' is given in Algorithm 4.

If G' has only a single vertex, then it is easy to see that the algorithm does not modify the graph. For the rest of this section, we assume that this is not the case. The following recursive definition is made in order to make the description of the algorithm easier.

Definition 5.4. Let G be a connected outerplanar graph with at least two vertices such that $G \setminus x$ has exactly two connected components for every cut vertex x and v be a non-cut vertex in the root block of G. For any cut vertex x of G, let G_x denote the subgraph of G induced on the vertices belonging to the unique child block attached at x and all its descendant blocks. If the root-bock of G is a non-trivial block, let $v, v_1, \ldots, v_t, v$ be the Hamiltonian cycle of the root-block of G, starting at v. Then,

$$Order(G, v) = \begin{cases} v, v_1 & \text{if } G \text{ is a single edge } (v, v_1) \\ v, v_1, \ldots, v_t & \text{if } G \text{ is 2-vertex-connected} \\ v, S_1, \ldots, S_t & \text{otherwise} \end{cases}$$

where, for each $1 \leq i \leq t$,

$$S_i = \begin{cases} v_i & \text{if } v_i \text{ is not a cut vertex in } G \\ Order(G_{v_i}, v_i), v_i & \text{otherwise.} \end{cases}$$

The following lemma gives a precise description of the order in which the algorithm encounters vertices of G'.

Lemma 5.7. *Let G' be a connected outerplanar graph with at least two vertices, such that for every cut vertex x of G', $G' \setminus x$ has exactly two connected components. If G' is given as the input graph to Algorithm 4 and v_0 is the non-cut vertex in the root block of G' from which the algorithm starts the traversal, then $Order(G', v_0)$ is the order in which Algorithm 4 encounters the vertices of G'.*

Since the proof of this lemma is easy but is lengthy and detailed, in order to make the reading easier we defer the proof to Section 5.9. Now, using Lemma 5.7 we derive some properties maintained by Algorithm 4.

Property 5.5.

1. *Every non-cut vertex of G' is encountered exactly once. Every cut vertex of G' is encountered exactly twice.*

2. *A non-cut vertex is completed when it is encountered for the first time. A cut vertex is bypassed when it is encountered for the first time and is completed when it is encountered for the second time. Each cut vertex is bypassed exactly once.*

Algorithm 4: Computing a 2-vertex-connected outerplanar supergraph

Input: A connected outerplanar graph $G'(V, E')$ such that $G' \setminus x$ has exactly two connected components for every cut vertex x of G'. A path decomposition $(\mathcal{P}', \mathscr{X}')$ of G'. The rooted block tree of G', the Hamiltonian cycle of each non-trivial block of G' and the first and last vertices of each non-root block of G'

Output: A 2-vertex-connected outerplanar supergraph $G''(V, E'')$ of G', a path decomposition $(\mathcal{P}'', \mathscr{X}'')$ of G''

$E'' = E'$, $(\mathcal{P}'', \mathscr{X}'') = (\mathcal{P}', \mathscr{X}')$
for *each vertex* $v \in V(G')$ **do**
    completed(v) = FALSE
    if v *is a cut vertex* **then** bypass(v) = FALSE
B = rootBlock
Choose v' to be some non-cut vertex of the rootBlock
$completed(v')$ =TRUE, $completedCount = 1$
$v = v'$
while $completedCount < |V(G')|$ **do**
    $v' = next_B(v)$
    $bypassLoopTaken$ =FALSE, $sequence$ = EmptyString
    while v' *is a cut vertex and* $bypass(v')$ *is FALSE* **do**
        $bypassLoopTaken$ =TRUE
        $bypass(v')$ =TRUE, $sequence$ =Concatenate($sequence, v'$)
        $B = child_{v'}(B)$, $v' = next_B(v')$
    if $bypassLoopTaken$ *is TRUE* **then**
        $e = (v, v')$, $bypassSeq(e) = sequence$
        $E'' = E'' \cup \{e\}$
        if $Gap_{\mathscr{X}'}(v, v') \neq \emptyset$ **then**
            if $LastIndex_{\mathscr{X}'}(v) < FirstIndex_{\mathscr{X}'}(v')$ **then for** $t \in Gap_{\mathscr{X}'}(v, v')$ **do** $X''_t = X''_t \cup \{v\}$
            else if $LastIndex_{\mathscr{X}'}(v') < FirstIndex_{\mathscr{X}'}(v)$ **then for** $t \in Gap_{\mathscr{X}'}(v, v')$ **do** $X''_t = X''_t \cup \{v'\}$
    if v' *is a cut vertex and* $bypass(v')$ *is TRUE* **then**
        $B = parent(B)$
    $completed(v')$= TRUE, $completedCount = completedCount + 1$
    $v = v'$

3. *Every vertex is completed exactly once and a vertex that is declared completed is never encountered again.*

Proof. The first part of the property follows directly from Lemma 5.7.

To prove the second part, observe that a vertex u is encountered only in Lines 6, 10 or 15. If $v' = u$ is a non-cut vertex, the inner while-loop will not be entered. The vertex u is completed in Line 7 or Line 24 before another vertex is encountered by executing Line 10 again.

For any cut vertex x, $bypass(x)$ = FALSE initially and it is changed only after x is encountered and the algorithm enters the inner while-loop with $v' = x$. When x is first encountered, $bypass(x)$ is FALSE and the algorithm gets into the inner while-loop and inside this loop $bypass(x)$ is set to TRUE. After this, $bypass(x)$ is never set to FALSE. Therefore, when x is encountered for the second time, the inner while-loop is not entered and before v' gets reassigned, x is completed in Line 24.

The third part of the property follows from the first two parts and Lemma 5.7. □

Lemma 5.8. *Each cut vertex of G' is bypassed exactly once by the algorithm and is associated with a unique edge in $E'' \setminus E'$. Every edge $e \in E'' \setminus E'$ has a non-empty sequence of bypassed cut vertices associated with it, given by $bypassSeq(e)$. Hence, the function f : cut vertices of $G' \mapsto E'' \setminus E'$, defined as*

$$f(x) = e \text{ such that } x \text{ is present in } bypassSeq(e)$$

is a surjective map.

Proof. From Property 5.5, each cut vertex x of G' is bypassed exactly once by the algorithm. Note that when x is bypassed in Line 14, x is appended to the string *sequence* and the variable $bypassLoopTaken$ was set to TRUE in the previous line. On exiting the inner while-loop, since $bypassLoopTaken$=TRUE, an edge is added in Line 18. Before adding the new edge e, in the previous line the algorithm set $bypassSeq(e)$ to be the sequence of cut vertices accumulated in the variable *sequence*. As we have seen, x is present in the string *sequence* and it is clear from the algorithm that *sequence* was not reset to emptystring before assigning it to $bypassSeq(e)$. Therefore, x is present in $bypassSeq(e)$. In the next iteration of the outer while-loop, *sequence* is reset to emptystring. Since x is bypassed only once, it will not be added to the string *sequence* again nor it will be part of $bypassSeq(e')$ for any other edge e'.

An edge e gets added in Line 18 only if $bypassLoopTaken$ is TRUE, while executing Line 16. However, the variable $bypassLoopTaken$ is set to FALSE in the outer while-loop every time just before entering the inner while-loop and is set to TRUE only inside the inner while-loop, where at least one cut vertex is bypassed in Line 14 and is added to *sequence*. Until the algorithm exits from the inner while-loop, and the next edge e is added, $bypassLoopTaken$ is maintained to be TRUE. This ensures that $bypassSeq(e)$ is a non-empty string for each edge $e \in E'' \setminus E'$. □

Property 5.6. *Let $e = (u_i, v_i)$ be an edge such that, at the time of adding the edge e in Line 18 of Algorithm 4, the variable v contained the value u_i and the variable v' contained the value v_i.*

- *The vertex $v = u_i$ is already completed at this time and the vertex v_i is completed subsequently.*

- *Cut vertices that belong to $bypassSeq(e)$ are precisely the vertices bypassed during the period from the execution of Line 10 just before adding the edge e in Line 18 to the time when e is added in Line 18. These vertices were bypassed in the order in which they appear in $bypassSeq(e)$.*

- *Each cut vertex bypassed before bypassing the first cut vertex that belong to $bypassSeq(e)$ belongs to the bypass sequence of one of the edges in $E'' \setminus E'$ which was added before e.*

- *If $bypassSeq(e) = x_1, x_2, \ldots, x_k$, then $u_i = x_0, x_1, x_2, \ldots, x_k, x_{k+1} = v_i$ is a path in G' such that*

  - *for each $1 \leq i \leq k$, x_i is the successor of x_{i-1} in the Hamiltonian cycle of the parent block in G', at the cut vertex x_i.*
  - *$v_i = x_{k+1}$ is the successor of x_k in the Hamiltonian cycle of the child block in G', at the cut vertex x_k.*

  Since each bypassed vertex is a cut vertex in G', it is easy to see that $e = (u_i, v_i)$ was not already an edge in G'.

Proof. Suppose that at the time of adding the edge e in Line 18 of Algorithm 4, the variable v contained the value u_i and the variable v' contained the value v_i. Observe that the variable v always gets its value from the variable v' in lines 8 and 25 and just before this, in Lines 7 and 24 v' was declared completed. Therefore the vertex assigned to v is always a completed vertex. Therefore, at the time of adding the edge e in Line 18, the vertex $v = u_i$ is already completed. After the edge is added in Line 18, in Line 24 the vertex assigned to $v' = v_i$ is completed. Since by Property 5.5 a vertex is completed only once, this is the only time at which v_i is completed.

Since each cut vertex is bypassed only once, and is added to the bypass sequence of the next edge added (see the proof of Lemma 5.8), it is clear that if a cut vertex is bypassed before x_1, it should belong to the bypass sequence of an edge in $E'' \setminus E'$ added before e. The remaining parts of the property are easy to deduce from lines 12 - 15 and Line 17 of the algorithm. □

Lemma 5.9. *G'' is 2-vertex-connected.*

Proof. We show that G'' does not have any cut vertices. Since G'' is a supergraph of G', if a vertex x is not a cut vertex in G', it will not be a cut vertex in G''. We need to show that the cut vertices in G' become non-cut vertices in G''. Consider a newly added edge (u, v) of G''. Without loss of generality, assume that u was completed before v in the traversal and for $e = (u, v)$, $bypassSeq(e) = (x_1, x_2, \ldots, x_k)$. By Property 5.6, $u, x_1, x_2, \ldots, x_k, v$ is a path in G'. When our algorithm adds the edge (u, v), it creates the cycle $u, x_1, x_2, \ldots, x_k, v, u$ in the resultant graph. Recall that, for each $1 \leq i \leq k$, $G' \setminus x_i$ had exactly two components; one containing x_{i-1} and the other containing x_{i+1}. After the addition of the edge (u, v), vertices x_{i-1}, x_i and x_{i+1} lie on a common cycle. Hence, after the edge (u, v) is added, for $1 \leq i \leq k$, x_i is no longer a cut vertex. Since by Lemma 5.8 every cut vertex in G' was part of the bypass sequence associated with some edge in $E'' \setminus E'$, all of them become non-cut vertices in G''. □

To prove that G'' is outerplanar, we can imagine the edges in $E'' \setminus E'$ being added to G' one at a time. Our method is to repeatedly use Lemma 5.3 and show that after each edge addition, the resultant graph remains outerplanar. We will first note down some properties maintained by Algorithm 4.

Let $\{e_i = (u_i, v_i) \mid 1 \leq i \leq m = |E'' \setminus E'|\}$ be the set of edges added by Algorithm 4. Assume that, for each $1 \leq i < m$, (u_i, v_i) was added before (u_{i+1}, v_{i+1}) and at the time of adding the edge e_i in Line 18 of Algorithm 4, the variable v contained the value u_i and the variable v' contained the value v_i. By Property 5.6, u_i is completed before v_i. Let $bypassSeq((u_i, v_i)) = x^i_1, x^i_2, \ldots, x^i_{k_i}$, where $k_i \geq 1$, and $P^i = (u_i = x^i_0, x^i_1, x^i_2, ..., x^i_{k_i}, x^i_{k_i+1} = v_i)$ be the associated path in G' (Property 5.6). Let B^i_j denote the block containing the edge (x^i_j, x^i_{j+1}) in G'. Clearly, B^1_0 is the root block of G'. The following statement is an immediate corollary of Property 5.6, with the definitions above.

Property 5.7. *For each $0 \leq j \leq k_i$, the vertex x^i_{j+1} is the successor of the vertex x^i_j in the Hamiltonian cycle of the block B^i_j. The path P^i shares only one edge with any block of G'.*

Property 5.8. *If $1 \leq i < j \leq m = |E'' \setminus E'|$, then $u_i \neq u_j$.*

Proof. By our assumption, at the time of adding the edge (u_i, v_i), we had $v = u_i$ and $v' = v_i$. By Property 5.6, the vertex $v = u_i$ was already completed at this time. After adding the edge $(v, v') = (u_i, v_i)$, the algorithm reassigns $v = v' = v_i$ in Line 25. By Property 5.5, the algorithm will never encounter the completed vertex u_i again, and this means that v' is never set to u_i in future. This also implies that v is never set to u_i in future, since v gets reassigned later only in Line 25, where it gets its value from the variable v'. Since $v = u_j$ when the edge (u_j, v_j) is added, we have $u_i \neq u_j$. □

At a stage of Algorithm 4, we say that a non-root block B is **touched**, if at that stage Algorithm 4 has already bypassed the cut vertex y such that B is the child block containing y. At any stage of Algorithm 4, we consider the root block of G' to be touched.

Property 5.9. *When Algorithm 4 has just finished adding the edge* (u_i, v_i)*, the touched blocks are precisely* $B_0^1 \cup \bigcup_{1\le j\le i}\{B_1^j, \ldots, B_{k_j}^j\}$. *This implies that if* $i < m$*, the blocks* $\{B_1^{i+1}, B_2^{i+1}, \ldots, B_{k_i}^{i+1}\}$ *remain untouched when the algorithm has just finished adding the edge* (u_i, v_i).

Proof. The root block B_0^1 is always a touched block by definition and the other touched blocks when Algorithm 4 has just finished adding the edge (u_i, v_i) are the child blocks attached to cut vertices bypassed so far. However, by Property 5.6, when Algorithm 4 has just finished adding the edge (u_i, v_i), a cut vertex x is already bypassed if and only if x belongs to *bypassSeq*(e_j) for some $j \le i$. For $j \le i$, we had $\mathit{bypassSeq}((u_j, v_j)) = x_1^j, x_2^j, \ldots, x_{k_j}^j$ and for $1 \le l \le k_j$, B_l^j is the child block attached at x_l^j by the last part of Property 5.6. Hence, the initial part of the property holds. From this, the latter part of the property follows, by Lemma 5.8. □

Property 5.10. *For each* $2 \le i \le m$*, when the algorithm has just finished adding the edge* (u_{i-1}, v_{i-1})*, the block* B_0^i *is a touched block.*

Proof. If B_0^i is the root block, the property is trivially true. Assume that this is not the case.

Consider the situation when the algorithm has just finished adding the edge (u_{i-1}, v_{i-1}) in Line 18. By Property 5.6, the next cut vertex to be bypassed is x_1^i, which is the first vertex appearing in *bypassSeq*(e_i), and B_0^i is the parent block attached at x_1^i. Let y be the cut vertex such that B_0^i is the child block at y. If y has been encountered by now, y would have been bypassed (Property 5.5), making B_0^i a touched block. Since a cut vertex is bypassed when it is encountered for the first time (Property 5.5) and y is the first vertex the algorithm encounters among the vertices in the block B_0^i (Lemma 5.7), if y is not yet encountered, it will contradict the fact that x_1^i is the next cut vertex to be bypassed, because x_1^i is a vertex in B_0^i. Therefore y should have been encountered earlier and therefore, B_0^i is a touched block. □

Lemma 5.10. *G'' is outerplanar.*

Proof. Let $G'_0 = G'$ and for each $1 \le i \le m$, let $G'_i(V, E'_i)$ be the graph obtained by assigning $E'_i = E' \cup \{(u_j, v_j) \mid 1 \le j \le i\}$. Let M^0 denote the root block of G'. We will prove that Algorithm 4 maintains the following invariants for each $0 \le i \le m$:

- The graph G'_i is outerplanar.

- When the algorithm has just finished adding the edge (u_i, v_i), the set of touched blocks, $\bigcup_{1\leq j\leq i}\{B_0^j, B_1^j, \ldots, B_{k_j}^j\}$, have merged together and formed a single block, which we call as the **merged block** M^i in G_i'. M^i will be taken as the root block of G_i'. The other blocks of G' remain the same in G_i'.

- If $i < m$, x_1^{i+1} is the successor of u_{i+1} in the Hamiltonian cycle of the block M^i.

By Lemma 5.5, $G_0' = G'$ is outerplanar and it is clear that the above invariants hold for G_0'. Assume that the invariants hold for each i, where $1 \leq i < h \leq m$. Consider the case when $i = h$. Since the invariants hold for $h-1$, x_1^h is the successor of u_h in the Hamiltonian cycle of the block M^{h-1}. By Property 5.9, the blocks $\{B_1^h, B_2^h, \ldots, B_{k_h}^h\}$ are untouched when the algorithm has just added the edge (u_{h-1}, v_{h-1}). Since the invariants hold for $h-1$, these blocks remain the same in G_{h-1}' as in G'. Therefore, the path $P^h = (u_h, x_1^h, x_2^h, \ldots, x_{k_h}^h, v_h)$ continues to satisfy the pre-conditions of Lemma 5.3 in G_{h-1}' (Property 5.7). On addition of the edge (u_h, v_h) to G_{h-1}', the resultant graph G_h' is outerplanar, by Lemma 5.3.

By Property 5.9, the blocks $\{B_1^h, B_2^h, \ldots, B_{k_h}^h\}$ are precisely the blocks that were not touched at the time when e_{h-1} was just added but became touched by the time when e_h is just added. However, by Lemma 5.4, the blocks $\{B_1^h, B_2^h, \ldots, B_{k_h}^h\}$ merges with M^{h-1} and forms the block M^h of G_h' and other blocks of G_h' are same as those of G_{h-1}' (and hence of G') when the edge e_h is added. Thus, all touched blocks have merged together to form the block M^h in G_h and the other blocks of G' remain the same in G_h.

Finally, we have to prove that the successor of u_{h+1} in the Hamiltonian cycle of the block M^h is x_1^{h+1}, which is the same as the successor u_{h+1} in the Hamiltonian cycle of the block B_0^{h+1} in G'. To see this, note that by Lemma 5.4, if v' is the successor of v in the block containing the edge (v, v') before an edge (u_j, v_j) is added, it remains so after adding this edge, unless $v = u_j$. By Property 5.8, $u_{h+1} \neq u_j$ for any $j < h+1$ and hence it follows that x_1^{h+1} remains the successor u_{h+1} in the block containing the edge (u_{h+1}, x_1^{h+1}) in G_h'. Hence, in order to prove that x_1^{h+1} is the successor of u_{h+1} in the Hamiltonian cycle of the block M^h, it suffices to prove that the edge (u_{h+1}, x_1^{h+1}) belongs to the block M^h in G_h'. In the previous paragraph we saw that, at the time of adding the edge (u_h, v_h), all the touched blocks so far have merged together to form the the block M^h of G_h'. Since the edge (u_{h+1}, x_1^{h+1}) is in the block B_0^{h+1} in G', which is a touched block by Property 5.10 when the algorithm has just finished adding the edge (u_h, v_h), the block B_0^{h+1} has also been merged into M^h and hence, the edge (u_{h+1}, x_1^{h+1}) is in the block M^h in G_h'.

Thus, all the invariants hold for $i = h$ and hence for each $1 \leq i \leq m$. Since $G'' = G_m'$ by definition, G'' is outerplanar. □

Lemma 5.11. *$(\mathcal{P}'', \mathscr{X}'')$ is a path decomposition of G'' of width at most $16p+15$.*

Proof. It is clear that $(\mathcal{P}'', \mathscr{X}'')$ is a path decomposition of G'', since we constructed it using the method explained in Section 5.2.

For each $e_i = (u_i, v_i) \in E'' \setminus E'$, let $bypassSeq(e_i) = x_1^i, x_2^i, \ldots, x_{k_i}^i$ and let S_i denote the set of cut vertices that belong to $bypassSeq(e_i)$. By Property 5.6, $u_i, x_1^i, \ldots, x_{k_i}^i, v_i$ is a path in G'.

We will show that, if $t \in Gap_{\mathscr{X}'}(u_i, v_i)$, then, $X'_t \cap S_i \neq \emptyset$. Without loss of generality, assume that $LastIndex_{\mathscr{X}'}(u_i) < FirstIndex_{\mathscr{X}'}(v_i)$. Since u_i is adjacent to x_1^i, both of them are together present in some bag $X'_t \in \mathscr{X}'$, with $t \leq LastIndex_{\mathscr{X}'}(u_i)$. Similarly, since v_i is adjacent to $x_{k_i}^i$, they both are together present in some bag $X'_t \in \mathscr{X}'$, with $t \geq FirstIndex_{\mathscr{X}'}(v_i)$. Suppose some bag $X'_t \in \mathscr{X}'$ with $t \in Gap_{\mathscr{X}'}(u_i, v_i)$ does not contain any element of S_i. Let $U_i = \{x_j^i \in S_i \mid x_j^i$ belongs to $X'_{t'} \in \mathscr{X}'$ for some $t' < t\}$ and $V_i = \{x_j^i \in S_i \mid x_j^i$ belongs to $X'_{t'} \in \mathscr{X}'$ for some $t' > t\}$. From the definitions, $x_1^i \in U_i$ and $x_{k_i}^i \in V_i$. If $U_i \cap V_i \neq \emptyset$, the vertices belonging to $U_i \cap V_i$ will be present in X'_t as well, which is a contradiction. Therefore, (U_i, V_i) is a partitioning of S_i. Let q be the maximum such that $x_q^i \in U_i$. Clearly, $q < k_i$. Since (x_q^i, x_{q+1}^i) is an edge in G', both x_q^i and x_{q+1}^i should be simultaneously present in some bag in $\mathscr{X}'$. But this cannot happen because $x_q^i \in U_i$ and $x_{q+1}^i \in V_i$. This is a contradiction and therefore, if $t \in Gap_{\mathscr{X}'}(u_i, v_i)$, then, $X'_t \cap S_i \neq \emptyset$.

By the modification done to the path decomposition to reflect the addition of an edge e_i, a vertex was inserted into $X''_t \in \mathscr{X}''$ only if $t \in Gap_{\mathscr{X}'}(u_i, v_i)$ and for each $X''_t \in \mathscr{X}''$ such that $t \in Gap_{\mathscr{X}'}(u_i, v_i)$, exactly one vertex ($u_i$ or v_i) was inserted into X''_t while adding e_i. Moreover, when this happens, $X'_t \cap S_i \neq \emptyset$. Therefore, for any t in the index set, $|X''_t| \leq |X'_t| + |\{i \mid 1 \leq i \leq m, S_i \cap X'_t \neq \emptyset\}|$. But, $|\{i \mid 1 \leq i \leq m, S_i \cap X'_t \neq \emptyset\}| \leq |X'_t|$, because $S_i \cap S_j = \emptyset$, for $1 \leq i < j \leq m$, by Lemma 5.8. Therefore, for any t, $|X''_t| \leq 2|X'_t| \leq 2(8p+8)$. Therefore, width of the path decomposition $(\mathcal{P}'', \mathscr{X}'')$ is at most $16p+15$. □

5.8 Time Complexity

For our preprocessing, we need to compute a rooted block tree of the given outerplanar graph G and compute the Hamiltonian cycles of each non-trivial block. These can be done in linear time [30, 60, 90]. The special tree decomposition construction in Govindan et al.[56] is also doable in linear time. Using the Hamiltonian cycle of each non-trivial block, we do only a linear time modification in Section 5.4, to produce the nice tree decomposition $(T, \mathscr{Y})$ of G of width 3. An optimal path decomposition of the tree T, can be computed in $O(n \log n)$ time [85]. For computing the nice path decomposition $(\mathcal{P}, \mathscr{X})$ of

G in Section 5.4, the time spent is linear in the size of the path decomposition obtained for T, which is $O(n \log n)$ [85], and the size of $(\mathcal{P}, \mathscr{X})$ is $O(n \log n)$. Computing the FirstIndex, LastIndex and Range of vertices and the sequence number of blocks can be done in time linear in the size of the path decomposition. Since the resultant graph is outerplanar, Algorithm 3 and Algorithm 4 adds only a linear number of new edges. Since the size of each bag in the path decompositions $(\mathcal{P}', \mathscr{X}')$ of G' and $(\mathcal{P}'', \mathscr{X}'')$ of G'' are only a constant times the size of the corresponding bag in $(\mathcal{P}, \mathscr{X})$, the time taken for modifying $(\mathcal{P}, \mathscr{X})$ to obtain $(\mathcal{P}', \mathscr{X}')$ and later modifying it to $(\mathcal{P}'', \mathscr{X}'')$ takes only time linear in size of $(\mathcal{P}, \mathscr{X})$; i.e., $O(n \log n)$ time. Hence, the time spent in constructing G'' and its path decomposition of width $O(\mathrm{pw}(G))$ is $O(n \log n)$.

5.9 Proof of Lemma 5.7

For any cut vertex y of G', let $s(y)$ denote the Hamiltonian cycle successor of y in the (unique) child block at y and let G'_y denote the subgraph of G' induced on the vertices belonging to the child block attached at y and its descendant blocks. We call the variables $v', v, B, completed[\ ], bypass[\ ], completedCount$ the variables relevant for the traversal. First we prove two basic lemmas which makes the proof of Lemma 5.7 easier.

Lemma 5.12. *At any point of execution, if a non-cut vertex u is encountered, i.e., v' is set to u, until a cut vertex is encountered in Line 10 or $completedCount = |V(G)|$, from each vertex the algorithm proceed to encounter its Hamiltonian successor in the current block, completing it and incrementing the $completedCount$ by one each time.*

Proof. If u is encountered in Line 6, the next vertex is encountered in Line 10, inside the outer while-loop. When v' is a non-cut vertex encountered in Line 10 or Line 15, the inner while-loop condition and the condition in Line 22 will be evaluated to false until a cut vertex is encountered in Line 10. Therefore, the variables relevant for the traversal can get updated in lines 24-25 and lines 10-11 only, until v' gets assigned to refer to a cut vertex. From this, the property follows. □

Lemma 5.13. *Suppose at a certain time T_1 of execution, Algorithm 4 has just executed Line 12 and the variable v' is referring to a cut vertex x in G' and the following conditions are also true:*

- C_1. *The current block being traversed, i.e. the block referred to by the variable B, is the parent block at x.*
- C_2. *$bypass(x)$ =FALSE and for each cut vertex y of G'_x, $bypass(y)$ =FALSE.*

C_3. *For each vertex y of G'_x, completed(y) = FALSE and completedCount = c, where $c \leq |V(G')| - |V(G'_x)|$*

Then, the algorithm will again come to Line 12 with the variable v' referring to the same cut vertex x. Let T_2 be the next time after T_1 when this happens. At time T_2, the following conditions will be true:

E_1. *The current block being traversed is the child block at x and during the time between T_1 and T_2 the algorithm never sets the variable B to a block other than the child block at x or its descendant blocks.*

E_2. *bypass(x) =TRUE and for each cut vertex y of G'_x, bypass(y) =TRUE.*

E_3. *completed(x) =FALSE and for each vertex y of G'_x other than x, completed(y) =TRUE and completedCount $= c+|V(G'_x)|-1 < |V(G')|$.*

E_4. *The order in which the algorithm encounters vertices during the period from the time x was encountered just before T_1 and till the time T_2 is $Order(G'_x, x), x$.*

Proof. We give a detailed proof of this lemma below, which is in principle just a description of the execution of the algorithm. Instead of going through the proof, the reader may verify the correctness of the lemma directly from the algorithm.

We prove this lemma using an induction on $n(x)$, the number of blocks in G'_x. For the base case, assume that $n_x = 1$; i.e., G'_x is a leaf block of G'. Suppose the assumptions in the statement of the lemma hold at time T_1. By this assumption, the condition of the inner while-loop in Line 12 has been evaluated to true at time T_1 and after executing Line 14 *bypass*(x) will be set to TRUE. Similarly, it follows from the assumptions that after executing Line 15 the current block is set as the child block at x and the algorithm sets $v' = s(x)$, the successor of x in the child block at x. Since the child block at x is a leaf block, $v' = s(x)$ is not a cut vertex. By Lemma 5.12, until v' gets assigned to refer to a cut vertex, from each vertex the algorithm proceed to encounter its Hamiltonian successor in the current block, completing it and incrementing the *completedCount* by one each time. Note that *completedCount* $< |V(G')|$ all this time, because at time T_1, we had $c \leq |V(G')| - |V(G'_x)|$ and the number of times *completedCount* was incremented since time T_1 is less than $|V(G'_x)|$. Since the only cut vertex in the current block is x itself, this goes on until x is encountered in Line 10 and then it reaches Line 12 at time T_2. From this, it follows that the order in which vertices were encountered during the period from the time x was encountered just before T_1 and till the time T_2 is the Hamiltonian cycle order of the child block at x, starting and ending at x. This is precisely $Order(G'_x, x), x$. It is evident that conditions E_1 - E_3 are also true at time T_2.

Now, we will assume that the lemma holds for all cut vertices y such that $n(y) < n(x)$. In order to prove the lemma, it is enough to prove that the lemma holds for x as well. Let $x, v_1, v_2, \ldots, v_t, x$ be the Hamiltonian cycle of the child block at x. Suppose the assumptions in the statement of the lemma hold at time T_1 when the Algorithm 4 has just executed Line 12 and $v' = x$. As in the base case, $bypass(x)$ will be set to TRUE and in Line 15 the current block is set as the child block at x and the algorithm sets $v' = s(x) = v_1$.

If the set $\{v_1, v_2, \ldots, v_t\}$ does not contain any cut vertices, we are in the base case and we are done. Otherwise, let l be the minimum index in $\{1, 2, \ldots, t\}$ such that v_l is a cut vertex. By Lemma 5.12, from each vertex the algorithm proceed to encounter its Hamiltonian successor in the current block, completing it and incrementing the *completedCount* each time until $v' = v_l$. Notice that $completedCount = c+l-1 \leq |V(G')|-|V(G'_x)|+l-1 \leq |V(G')|-|V(G'_{v_l})|$, when v_l is encountered in Line 10. After this, the algorithm reaches Line 12 and executes it with $v' = v_l$ at time T'_l. At this time, the block being traversed is the parent block at v_l. Since v_l or any other vertex in G'_{v_l} were not encountered till now after T_1, and by the assumptions of the lemma about the state of the traversal related variables at time T_1, we know that at time T'_l, $bypass(v_l)$ =FALSE and for each cut vertex z of G'_{v_l}, $bypass(z)$ =FALSE. Similarly, for each vertex y of G'_{v_l}, *completed*(y) = FALSE. Thus, the preconditions of the lemma are satisfied for the vertex v_l and G'_l at time T'_l.

The order in which the vertices are encountered during the period from the time x was encountered just before T_1 and till the time T'_l is $x, v_1, v_2, \ldots, v_l$. Since v_l is a cut vertex in the child block at x, $n(v_l) < n(x)$. Therefore, by induction hypothesis, the algorithm will again come to Line 12 with the variable $v' = v_l$ and if T''_l is the next time this happens after T_l, the conditions E_1 - E_4 will be satisfied with v_l replacing x, T'_l and T''_l replacing T_1 and T_2 respectively and $c + l - 1$ replacing c.

At time T''_l, when the algorithm is back at Line 12 and executes the line with $v' = v_l$ and $bypass(v_l)$ =TRUE, the while-loop condition will evaluate to FALSE and so, the loop will not be entered. When the algorithm reaches Line 22, the condition will evaluate to true and therefore, in Line 23, the variable B will be updated to its parent block. Since B is the child block at v_l before this, B will be updated to the parent block at v_l, which is the same as the child block at x. In Line 24, *completed*(v_l) is set to TRUE and *completedCount* becomes $c + l - 1 + |V(G'_{v_l})| < |V(G')|$ and therefore, in Line 9, the outer while-loop condition evaluates to TRUE. Since $v' = v_l$ now, the algorithm executes Line 10, and v' will be updated to v_{l+1}, the successor of v_l in B. From the time x was encountered just before T_1, the order in which the algorithm has encountered vertices is $x, v_1, v_2, \ldots, v_{l-1}, Order(G'_{v_l}, v_l), v_l, v_{l+1}$.

By repeating similar arguments as above, we can reach the following conclusion. If i is the maximum index in $\{1, 2, \ldots, t\}$ such that v_i is a cut vertex, the algorithm will come to Line 12 with the variable $v' = v_i$ at time T''_i such that the conditions below will be true at time T''_i.

- The current block being traversed is the child block at v_i. During the time between T_1 and T''_i the algorithm never sets the variable B to a block other than the child block at x or its descendant blocks.

- For each cut vertex y of G'_x, $bypass(y)$ =TRUE.

- For $v_k \in \{v_i, v_{i+1}, \ldots, v_t, x\}$, $completed(v_k)$ =FALSE and for each vertex y of G'_x outside this set, $completed(y)$ =TRUE. Moreover, $completedCount = c + i - 1 + \sum_{1 \leq j \leq i, v_j \text{ is a cut vertex}} (|V(G'_{v_j})| - 1)$.

- The order in which the algorithm encounters vertices during the period from the time x was encountered just before T_1 and till the time T''_i is given by $x, S_1, \ldots, S_{v_i}$, where for $1 \leq j \leq i$, $S_j = v_j$ if v_j is not a cut vertex in G' and $S_j = Order(G'_{v_j}, v_j), v_j$ otherwise.

By similar arguments as at time T''_l, we can show that, at the time T''_i, the inner while-loop condition is false, because $bypass(v_i)$ =TRUE and in Line 23 the variable B will be updated to the child block at x. In Line 24, $completed(v_i)$ is set to TRUE and $completedCount$ becomes $c+i+\sum_{1 \leq j \leq i, v_j \text{ is a cut vertex}} (|V(G'_{v_j})| - 1)$. Since this value is less than $|V(G')|$, in Line 9 the outer while-loop condition evaluates to TRUE. Since $v' = v_i$ now, the algorithm executes Line 10, and v' will be updated to the successor of v_i in B.

If $v_i = v_t$, then at this stage, $v' = x$ and when the algorithm reaches Line 12, that is the time T_2 mentioned in the lemma and the order in which the algorithm has encountered vertices is $x, S_1, \ldots, S_{v_t}, x$, where for $1 \leq j \leq t$, $S_j = v_j$ if v_j is not a cut vertex in G' and $S_j = Order(G'_{v_j}, v_j), v_j$ otherwise. If $v_i \neq v_t$, by the maximality of i and using Lemma 5.12, until v' gets assigned the value x in Line 10, from each vertex the algorithm proceed to encounter its Hamiltonian successor in the current block, completing it and incrementing the $completedCount$ each time. We have $completedCount < |V(G')|$ all this time, because at time T_1 we had $completedCount = c$ and afterwards $completedCount$ was incremented once for each vertex y in G'_x for which $completed(y) = TRUE$ but $completed(x)$ =FALSE still. When x is encountered in Line 10 and then the algorithm reaches Line 12, that is the time T_2 mentioned in the lemma. From the time when x was encountered just before T_1, the order in which the algorithm has encountered vertices is $x, S_1, \ldots, S_{v_t}, x$, where for $1 \leq j \leq i$, $S_j = v_j$ if v_j is not a cut vertex in G' and $S_j = Order(G'_{v_j}, v_j), v_j$ otherwise. Thus, in both cases, at time T_2 the order in which the algorithm has encountered vertices from the time when x was encountered just before T_1 is given by $x, S_1, \ldots, S_{v_t}, x = Order(G'_x, x), x$. Notice also that the conditions E_1-E_3 also hold at time T_2 and hence the lemma is proved. $\square$

Lemma 5.7. If G' is given as the input graph to Algorithm 4, where G' is a connected outerplanar graph with at least two vertices, such that for every cut vertex x of G', $G' \setminus x$ has exactly two connected components, and if v_0 is the non-cut vertex in the root block of G' from which the algorithm starts the traversal, then $Order(G', v_0)$ is the order in which Algorithm 4 encounters the vertices of G'.

Proof. Suppose $v_0, v_1, \ldots, v_t, v_0$ is the Hamiltonian cycle of the root block of G'. Our method is to use Lemma 5.13 at each cut vertex in the root block of G' and Lemma 5.12 at each non-cut vertex in the root block of G', until $completedCount = |V(G')|$. We will see that this will go on until v_t is completed.

After the initializations done in lines 2 - 7, for every cut vertex y in G', *bypass*(y) =FALSE, current block B is the root block of G', $v' = v_0$, $completedCount = 1$, *completed*(v_0)=TRUE and for every other vertex y in G', *completed*(y) =FALSE. Since the outer while-loop condition is TRUE, the algorithm enters the loop and in Line 10, v_1 is encountered. Let us call this instant as time T_1.

If the set $\{v_1, v_2, \ldots, v_t\}$ contains a cut vertex, let l be the minimum index in $\{1, 2, \ldots, t\}$ such that v_l is a cut vertex. Otherwise, let $v_l = v_t$. By Lemma 5.12, until v' gets assigned the value v_l in Line 10, from each vertex the algorithm proceed to encounter its Hamiltonian successor in the current block (i.e the root block), completing it and incrementing the $completedCount$ each time, making $completedCount = l < |V(G')|$ when v_l is encountered in Line 10. The algorithm then reaches Line 12 and executes it with $v' = v_l$. Let us call this instant as time T_l'. The order in which the vertices are encountered from the beginning of execution of the algorithm is $v_0, v_1, v_2, \ldots, v_l$.

If $v_l = v_t$ and v_l is not a cut vertex, that means the graph G' does not have a cut vertex and $|V(G')| = t + 1$. In this case, the inner while-loop condition evaluates to FALSE. Similarly, the condition in Line 22 also evaluates to FALSE. In Line 24, the algorithm sets *completed*(v_t) =TRUE and increments $completedCount$, making $completedCount = t + 1 = |V(G')|$. When the algorithm executes Line 9, the outer while-loop condition evaluates to FALSE and the algorithm terminates. The order in which the vertices were encountered from the beginning of execution of the algorithm is $v_0, v_1, \ldots, v_t = Order(G', v_0)$.

The other case is when v_l is a cut vertex at time T_l'. At this time, $completedCount = l \leq |V(G')| - |V(G_l')|$ and by similar arguments as in the proof of Lemma 5.13, the pre-conditions of the lemma are satisfied for the vertex v_l and G_l' at time T_l'. Therefore, by Lemma 5.13 applied to the vertex

v_l and G'_l, the algorithm will again come to Line 12 with the variable $v' = v_l$. If T''_l is the next time this happens after T'_l, we can see that the state of the traversal related variables at time T''_l is similar to those we obtained in the proof of Lemma 5.13, except that $completedCount = l + |V(G'_{v_l})| - 1$.

Following similar arguments as in the proof of Lemma 5.13, we can show that when the algorithm executes Line 23 the next time after T''_l, the variable B will be updated to the parent block at v_l, which is the same as the root block. In Line 24, $completed(v_l)$ is set to TRUE and $completedCount$ becomes $l + |V(G'_{v_l})|$.

By repeating similar arguments as in the proof of Lemma 5.13, we can reach the following conclusion. If i is the maximum index in $\{1, 2, \ldots, t\}$ such that v_i is a cut vertex, the algorithm will come to Line 12 with the variable $v' = v_i$ at time T''_i such that the conditions below will be true at time T''_i.

- The current block being traversed is the child block at v_i.
- For each cut vertex y of G', $bypass(y)$ =TRUE.
- For $v_k \in \{v_i, v_{i+1}, \ldots, v_t\}$, $completed(v_k)$ =FALSE and for each vertex y of G' outside this set, $completed(y)$ =TRUE. Moreover, $completedCount$ $= i + \sum_{1 \leq j \leq i, v_j \text{ is a cut vertex}} (|V(G'_{v_j})| - 1)$.
- The order in which the algorithm encounters vertices from the beginning of execution of the algorithm till the time T''_i is given by $v_0, S_1, \ldots, S_{v_i}$, where for $1 \leq j \leq i$, $S_j = v_j$ if v_j is not a cut vertex in G' and $S_j = Order(G'_{v_j}, v_j), v_j$ otherwise.

By similar arguments as earlier, when the algorithm executes Line 23 the next time after T''_i, the variable B will be updated to the parent block at v_i, which is the same as the root block. In Line 24, $completed(v_i)$ is set to TRUE and $completedCount$ becomes $i + 1 + \sum_{1 \leq j \leq i, v_j \text{ is a cut vertex}} (|V(G'_{v_j})| - 1)$.

If the above sum is equal to $|V(G')|$, that means $v_i = v_t$. When the algorithm executes Line 9, the outer while-loop condition evaluates to FALSE and the algorithm terminates. The order in which the vertices were encountered from the beginning of execution of the algorithm is $v_0, S_1, \ldots, S_{v_t}$, where for $1 \leq j \leq t$, $S_j = v_j$ if v_j is not a cut vertex in G' and $S_j = Order(G'_{v_j}, v_j), v_j$ otherwise. This is the same as $Order(G', v_0)$.

Instead, if the sum is less than $|V(G')|$, then $v_i \neq v_t$. In Line 9 the outer while-loop condition evaluates to TRUE. Since $v' = v_i$ and B is the root block, when the algorithm executes Line 10, v' will be updated to v_{i+1}, the successor of v_i in the root block. By the maximality of i and using similar arguments as earlier, we can show that until v' gets assigned the value v_t in Line 10, from each vertex the algorithm proceed to encounter its Hamiltonian successor in the current block. When v_t is encountered in Line 10 and then reach Line 12, the condition of the inner while-loop will evaluate to FALSE because v_t is not a cut vertex. Similarly, the condition in Line 22 will also evaluate to FALSE. In Line 24, $completed(v_t)$ is set to TRUE and $completedCount$ becomes $t + 1 + \sum_{1 \leq j \leq i, v_j \text{ is a cut vertex}} (|V(G'_{v_j})| - 1)$, which is equal to $|V(G')|$. When the algorithm executes Line 9, the outer while-loop condition evaluates to FALSE and the algorithm terminates. From the beginning of execution, the order in which the algorithm has encountered vertices is $v_0, S_1, \ldots, S_{v_t}$, where for $1 \leq j \leq i$, $S_j = v_j$ if v_j is not a cut vertex in G' and $S_j = Order(G'_{v_j}, v_j), v_j$ otherwise. This order is the same as $Order(G', v_0)$.

In all cases, from the beginning of execution of Algorithm 4 till it terminates, the order in which the algorithm encounters the vertices of G' is given by $Order(G', v_0)$. □

5.10 Conclusion

In this chapter, we have described a $O(n \log n)$ time algorithm to add edges to a given connected outerplanar graph G of pathwidth p to get a 2-vertex-connected outerplanar graph G'' of pathwidth at most $16p + 15$. We also get the corresponding path decomposition of G'' in $O(n \log n)$ time. Our technique is to produce a nice path decomposition of G and make use of the properties of this decomposition, while adding the new edges. Biedl [14] obtained an algorithm for computing planar straight line drawings of a 2-vertex-connected outerplanar graph G on a grid of height $O(p)$. In conjunction with our algorithm, Biedl's algorithm will work for any outer planar graph G. As explained by Biedl [14], this gives a constant factor approximation algorithm to get a planar drawing of G of minimum height.

Chapter 6

Matchings in TD-Delaunay graphs - Equilateral triangle matchings

Given a point set P and a class $\mathcal{C}$ of geometric objects, $G_{\mathcal{C}}(P)$ is a geometric graph with vertex set P such that any two vertices p and q are adjacent if and only if there is some $C \in \mathcal{C}$ containing both p and q but no other points from P. In this chapter[1] we study $G_{\triangledown}(P)$ graphs where $\triangledown$ is the class of downward equilateral triangles (i.e. equilateral triangles with one of their sides parallel to the x-axis and the corner opposite to this side below the side parallel to the x-axis). For point sets in general position, these graphs have been shown to be equivalent to half-Θ_6 graphs and TD-Delaunay graphs.

The main result in this chapter is that for point sets P in general position, $G_{\triangledown}(P)$ always contains a matching of size at least $\left\lceil \frac{|P|-1}{3} \right\rceil$ and this bound is tight. We also give some structural properties of $G_{\text{\ding{86}}}(P)$ graphs, where ✡ is the class which contains both upward and downward equilateral triangles. We show that for point sets in general position, the block cut point graph of $G_{\text{\ding{86}}}(P)$ is simply a path. Through the equivalence of $G_{\text{\ding{86}}}(P)$ graphs with Θ_6 graphs, we also derive that any Θ_6 graph can have at most $5n - 11$ edges, for point sets in general position.

[1] Joint work with Ahmad Biniaz, Anil Maheshwari and Michiel Smid. This work has been accepted for publication in Theoretical Computer Science.

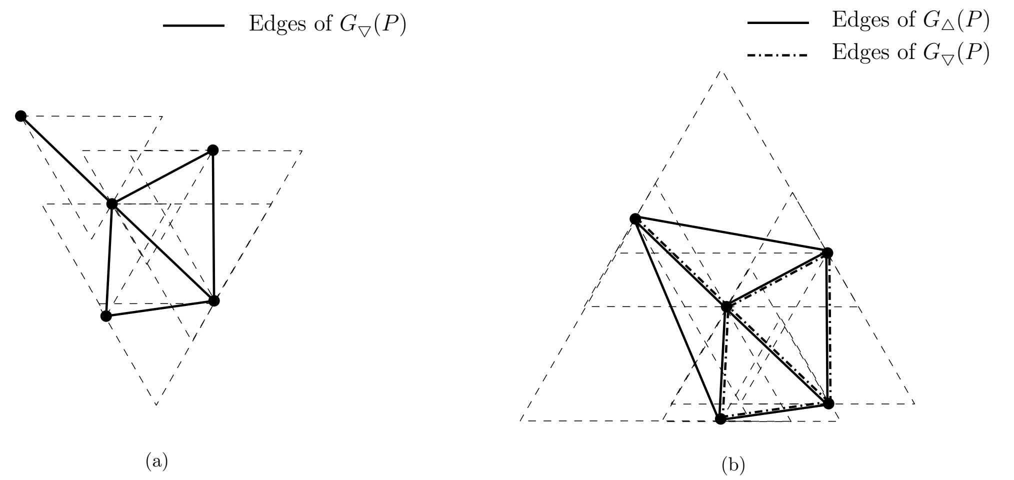


Figure 6.1: A point set P and its (a) $G_\bigtriangledown(P)$ and (b) $G_{\text{✡}}(P)$.

6.1 Introduction

In this work, we study the structural properties of some special geometric graphs defined on a set P of n points on the plane. A point set P is said to be in general position, if the line passing through any two points from P does not make angles 0°, 60° or 120° with the horizontal [15, 76]. We consider only point sets that are in general position and our results in this chapter assume this pre-condition.

First we revisit some of the definitions we made in Section 1.3. A down (resp. up)-triangle is an equilateral triangle with one side parallel to the x-axis and the corner opposite to this side below (resp. above) the side parallel to the x-axis, as in $\bigtriangledown$ (resp. $\bigtriangleup$). Given a point set P, $G_\bigtriangledown(P)$ (resp. $G_\bigtriangleup(P)$) is defined as the graph whose vertex set is P and that has an edge between any two vertices p and q if and only if there is a down-(resp. up-)triangle containing both points p and q but no other points from P. We also define another graph $G_{\text{✡}}(P)$ as the graph whose vertex set is P and that has an edge between any two vertices p and q if and only if there is a down-triangle or an up-triangle containing both points p and q but no other points from P (See Figure 6.1). In Section 6.3 we will see that, for any point set P in general position, its $G_\bigtriangledown(P)$ graph is the same as the well known Triangle Distance Delaunay (TD-Delaunay) graph of P and the half-Θ_6 graph of P on so-called negative cones. Moreover, $G_{\text{✡}}(P)$ is the same as the Θ_6 graph of P [15, 35].

Given a point set P and a class $\mathcal{C}$ of geometric objects, the maximum $\mathcal{C}$-matching problem is to compute a subclass $\mathcal{C}'$ of $\mathcal{C}$ of maximum cardinality such that no point from P belongs to more than one element of $\mathcal{C}'$ and for each $C \in \mathcal{C}'$, there are exactly two points from P which lie inside C. Dillencourt [41] proved that every point set admits a perfect circle-matching. Ábrego et al. [1] studied the isothetic square matching problem. Bereg et al. concentrated on matching points using axis-aligned squares and rectangles [11].

A matching in a graph G is a subset M of the edge set of G such that no two edges in M share a common end-point. A matching is called a maximum matching if its cardinality is the maximum among all possible matchings in G. If all vertices of G appear as end-points of some edge in the matching, then it is called a perfect matching. It is not difficult to see that for a class $\mathcal{C}$ of geometric objects, computing the maximum $\mathcal{C}$-matching of a point set P is equivalent to computing the maximum matching in the graph $G_{\mathcal{C}}(P)$.

The maximum $\triangle$-matching problem, which is the same as the maximum matching problem on $G_{\triangle}(P)$, was previously studied by Panahi et al. [76]. It was claimed that, for any point set P of n points in general position, any maximum matching of $G_{\triangle}(P)$ (and $G_{\bigtriangledown}(P)$) will match at least $\left\lfloor \frac{2n}{3} \right\rfloor$ vertices. But we found that their proof of Lemma 7, which is very crucial for their result, has gaps. By a completely different approach, we show that for any point set P in general position, $G_{\bigtriangledown}(P)$ (and by symmetric arguments, $G_{\triangle}(P)$) will have a maximum matching of size at least $\left\lceil \frac{n-1}{3} \right\rceil$; i.e, at least $2\left(\left\lceil \frac{n-1}{3} \right\rceil\right)$ vertices are matched. We also give examples of point sets, where our bound is tight.

We also prove some structural and geometric properties of the graphs $G_{\bigtriangledown}(P)$ (and by symmetric arguments, $G_{\triangle}(P)$) and $G_{\text{✡}}(P)$. It will follow that for point sets in general position, Θ_6 graphs can have at most $5n - 11$ edges and their block cut point graph is a simple path.

6.2 Notations used in this chapter

Our notations are similar to those used in [15], with some minor modifications adopted for convenience. A *cone* is the region in the plane between two rays that emanate from the same point, its apex. Consider the rays obtained by a counter-clockwise rotation of the positive x-axis by angles of $\frac{i\pi}{3}$ with $i = 1, \ldots, 6$ around a point p. (See Figure 6.2). Each pair of successive rays, $\frac{(i-1)\pi}{3}$ and $\frac{i\pi}{3}$, defines a cone, denoted by $A_i(p)$, whose apex is p. For $i \in \{1, \ldots, 6\}$, when i is odd, we denote $A_i(p)$ using $C_{\frac{i+1}{2}}(p)$ and the cone opposite to $C_i(p)$ using $\overline{C_i}(p)$. We call $C_i(p)$ a positive cone around p and $\overline{C_i}(p)$ a negative cone around p. For each cone $\overline{C_i}(p)$ (resp. $C_i(p)$), let $\ell_{\overline{C_i}(p)}$ (resp. $\ell_{C_i(p)}$) be its bisector. If $p' \in \overline{C_i}(p)$, then let $\overline{c_i}(p, p')$ denote the distance between p and the orthogonal projection of p' onto $\ell_{\overline{C_i}(p)}$. Similarly, if $p' \in C_i(p)$, then let $c_i(p, p')$ denote the distance between p and the orthogonal projection of p' onto $\ell_{C_i(p)}$. For $1 \leq i \leq 3$, let $V_i(p) = \{p' \in P \mid p' \in C_i(p), p' \neq p\}$ and $\overline{V_i}(p) = \{p' \in P \mid p' \in \overline{C_i}(p), p' \neq p\}$. For any two points p and q, the smallest down-triangle containing p and q is denoted by $\bigtriangledown pq$ and the smallest up-triangle containing p and q is denoted by $\triangle pq$. If G_1 and G_2 are graphs on the same vertex set, $G_1 \cap G_2$ (resp. $G_1 \cup G_2$) denotes the graph on the same vertex set whose edge set is the intersection (resp. union) of the edge sets of

G_1 and G_2.

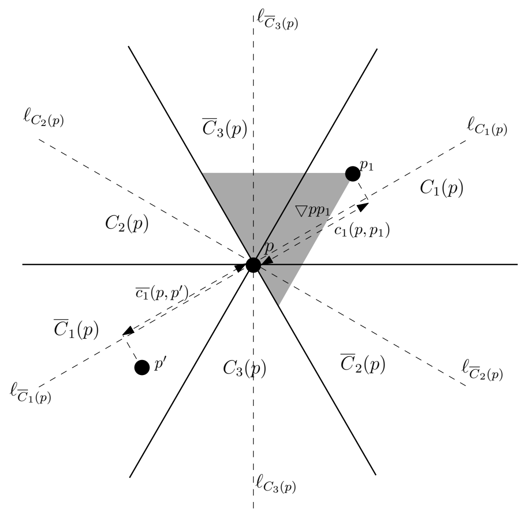


Figure 6.2: Six angles around a point p.

6.3 Preliminaries

In this section, we describe some basic properties of the geometric graphs described earlier and their equivalence with other geometric graphs which are well known in the literature.

The class of down-triangles (and up-triangles) admits a shrinkability property [1]: each triangle object in this class that contains two points p and q, can be shrunk such that p and q lie on its boundary. It is also clear that we can continue the shrinking process—from the edge that does not contain neither p or q—until at least one of the points, p or q, becomes a triangle vertex and the other point lies on the edge opposite to this vertex. After this, if we shrink the triangle further, it cannot contain p and q together. Therefore, for any pair of points p and q, $\triangledown pq$ ($\triangle pq$) has one of the points p or q at a vertex of $\triangledown pq$ ($\triangle pq$) and the other point lies on the edge opposite to this vertex. In Figure 6.1, triangles are shown after shrinking.

By the shrinkability property, for the $\triangledown$-matching problem, it is enough to consider the smallest down-triangle for every pair of points (p, q) from P. Thus, $G_{\triangledown}(P)$ is equivalent to the graph whose vertex set is P and that has an edge between any two vertices p and q if and only if $\triangledown pq$ contains no other points from P. Notice that if $\triangledown pq$ has p as one of its vertices, then $q \in \overline{C_1}(p) \cup \overline{C_2}(p) \cup \overline{C_3}(p)$. The following two properties are simple, but useful.

Property 6.1. *Let p and p' be two points in the plane. Let $i \in \{1, 2, 3\}$. The point p is in the cone $C_i(p')$ if and only if the point p' is in the cone $\overline{C}_i(p)$. Moreover, if p is in the cone $C_i(p')$, then $c_i(p', p) = \overline{c_i}(p, p')$.*

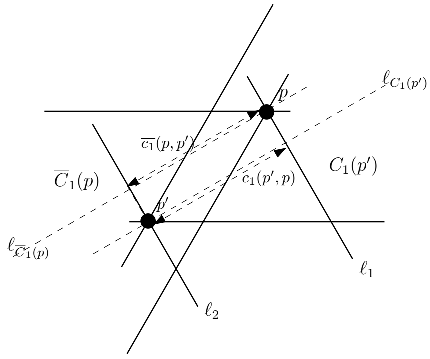


Figure 6.3: Proof of Property 6.1.

Proof. The first part of the claim is obvious. Now, without loss of generality, assume that $i = 1$ and $p \in C_1(p')$. (See Figure 6.3). Since $\ell_{\overline{C_1}(p)}$ is the bisector of $\overline{C_1}(p)$ and $\ell_{C_1(p')}$ is the bisector of $C_1(p')$, $\ell_{\overline{C_1}(p)}$ and $\ell_{C_1(p')}$ are parallel lines. Hence, $\overline{c_1}(p, p')$ is the perpendicular distance of p' to the line ℓ_1, which makes an angle 120° with the horizontal and passes though p. Similarly, $c_1(p', p)$ is the perpendicular distance of p to the line ℓ_2, which makes an angle 120° with the horizontal and passes though p'. Hence both $\overline{c_1}(p, p')$ and $c_1(p', p)$ are equal to the perpendicular distance between the lines ℓ_1 and ℓ_2. □

Property 6.2. *Let P be a point set, $p \in P$ and $i \in \{1, 2, 3\}$. If $\overline{V}_i(p)$ is non-empty, then, in $G_\bigtriangledown(P)$, the vertex p' corresponding to the point in $\overline{V}_i(p)$ with the minimum value of $\overline{c_i}(p, p')$ is the unique neighbor of vertex p in $\overline{V}_i(p)$.*

Proof. Assume $\overline{V}_i(p) \neq \emptyset$. For any point p' in $\overline{V}_i(p)$, it is easy to see that $\bigtriangledown pp'$ contains no points outside the cone $\overline{C_i}(p)$. Let p' be the point with the minimum value of $\overline{c_i}(p, p')$. The minimality ensures that $\bigtriangledown pp'$ does not contain any other point other than p and p' from P. Therefore, p and p' are neighbors in $G_\bigtriangledown(P)$.

In order to prove uniqueness, consider any point q in $P \cap \overline{V}_i(p)$ other than p and p'. It can be seen that $\bigtriangledown pq$ contains the point p' and therefore, p and q are not adjacent in $G_\bigtriangledown(P)$. Thus p' is the only neighbor of p in $\overline{V}_i(p)$. □

Consider a point set P and let $p, q \in P$ be two distinct points. By Property 6.1, $\exists i \in \{1, 2, 3\}$ such that $p \in \overline{C_i}(q)$ or $q \in \overline{C_i}(p)$; by the general position assumption, both conditions cannot hold simultaneously. Since $\bigtriangledown pq$ has either p or q as a vertex, Property 6.2 implies that we can construct $G_\bigtriangledown(P)$ as follows. For every point $p \in P$, and for each of the three cones, $\overline{C_i}$, for $i \in \{1, 2, 3\}$, add an edge from p to the point p' in $\overline{V_i}(p)$ with the minimum value of $\overline{c_i}(p, p')$, if $\overline{V_i}(p) \neq \emptyset$. This definition of $G_\bigtriangledown(P)$ is the same as the definition of the half-Θ_6-graph on negative cones ($\overline{C_i}$), given by Bonichon et al. [15]. We can similarly define the graph $G_\bigtriangledown(P)$ using the cones C_i instead of $\overline{C_i}$, for $i \in \{1, 2, 3\}$, and show that it is equivalent to the half-Θ_6 graph on positive cones (C_i), given by Bonichon et al. [15]. In Bonichon et al. [15], it was shown that for point sets in general position, the half-Θ_6-graph, the *triangular distance-Delaunay graph* (TD-Del) [35], which are 2-spanners, and the *geodesic embedding* of P, are all equivalent.

The Θ_k-graphs discovered by Clarkson [37] and Keil [62] in the late 80's, are also used as spanners [73]. In these graphs, adjacency is defined as follows: the space around each point p is decomposed into $k \geqslant 2$ regular cones, each with apex p, and a point q of a given cone C is linked to p if, from p, the orthogonal projection of q onto C's bisector [2] is the nearest point in C. In Bonichon et al. [15], it was shown that every Θ_6-graph is the union of two half-Θ_6-graphs, defined by C_i and $\overline{C}_i$ cones. In our notation this is same as the graph $G_\triangledown(P) \cup G_\vartriangle(P)$, which by definition, is equivalent to $G_{\davidsstar}(P)$. Thus, for a point set in general position, $\Theta_6(P) = G_{\davidsstar}(P)$.

6.4 Some properties of $G_\triangledown(P)$

6.4.1 Planarity

Chew defined [35] TD-Delaunay graph to be a planar graph and its equivalence with $G_\triangledown(P)$ graph implies that $G_\triangledown(P)$ is planar. This also follows from the general result that Delaunay graph of any convex distance function is a planar graph [17]. For the sake of completeness, we include a direct proof here.

Lemma 6.1. *For a point set P, its $G_\triangledown(P)$ is a plane graph, where its edges are straight line segments between the corresponding end-points.*

Proof. Whenever there is an edge between p and q in $G_\triangledown(P)$, we draw it as a straight line segment from p to q. Notice that this segment always lies within $\triangledown pq$. We will show that this gives a planar embedding of $G_\triangledown(P)$. Consider

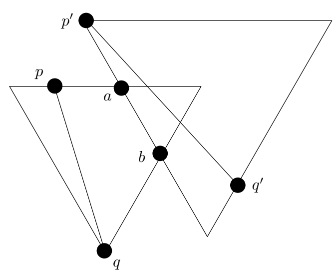


Figure 6.4: Intersection of $\triangledown pq$ and $\triangledown p'q'$ does not lead to crossing of edges pq and $p'q'$.

two edges pq and $p'q'$ of $G_\triangledown(P)$. If the interiors of $\triangledown pq$ and $\triangledown p'q'$ have no

[2]Sometimes the definition of Θ_k-graphs allows the orthogonal projection to be made to any ray in the cone C. But in our definition, we stick to the convention that the orthogonal projection is made to the bisector of C.

point in common, the line segments pq and $p'q'$ can not cross each other. Suppose the interiors of $\bigtriangledown pq$ and $\bigtriangledown p'q'$ share some common area. The case that $\bigtriangledown pq \subseteq \bigtriangledown p'q'$ (or vice versa) is not possible, because in this case $\bigtriangledown p'q'$ contains p and q (or $\bigtriangledown pq$ contains p' and q'), which contradicts its emptiness. Since $\bigtriangledown pq$ and $\bigtriangledown p'q'$ have parallel sides, this implies that one corner of $\bigtriangledown pq$ infiltrates into $\bigtriangledown p'q'$ or vice versa (see Figure 6.4). Thus their boundaries cross at two distinct points, a and b. Since $P \cap \bigtriangledown p'q' \cap \bigtriangledown p'q' = \emptyset$, the points p and q must be on that portion of the boundary of $\bigtriangledown pq$ that does not lie inside $\bigtriangledown p'q'$. So the line through ab separates pq from $p'q'$. $\square$

Throughout this chapter, we use $G_\bigtriangledown(P)$ to represent both the abstract graph and its planar embedding described in Lemma 6.1. The meaning will be clear from the context.

6.4.2 Connectivity

In this section, we prove that for a point set P, its $G_\bigtriangledown(P)$ is connected. As stated in the following lemma, between every pair of vertices, there exist a path with a special structure.

Lemma 6.2. *Let P be a point set with $p, q \in P$. Then, in $G_\bigtriangledown(P)$, there is a path between p and q which lies fully in $\bigtriangledown pq$ and hence $G_\bigtriangledown(P)$ is connected.*

Proof. We will prove this using induction on the rank of the area of $\bigtriangledown pq$. For any pair of distinct points $p, q \in P$, if the interior of $\bigtriangledown pq$ does not contain any point from P, by definition, there is an edge from p to q in $G_\bigtriangledown(P)$. By induction, assume that for pairs of points $x, y \in P$ such that the area of $\bigtriangledown xy$ is less than the area of $\bigtriangledown pq$, in the graph in $G_\bigtriangledown(P)$, there is a path which lies fully in $\bigtriangledown xy$ between x and y.

If the interior of $\bigtriangledown pq$ does not contain any point from P, there is an edge from p to q in $G_\bigtriangledown(P)$. Otherwise, there is a point $x \in P$ which is in the interior of $\bigtriangledown pq$. This implies $\bigtriangledown px \subset \bigtriangledown pq$ and $\bigtriangledown xq \subset \bigtriangledown pq$. Since the area of $\bigtriangledown px$ and the area of $\bigtriangledown xq$ are both less than the area of $\bigtriangledown pq$, by the induction hypothesis, there is a path that lies in $\bigtriangledown px$ between p and x and there is a path that lies in $\bigtriangledown xq$ between x and q. By concatenating these two paths, we get a path which lies in $\bigtriangledown pq$ between p and q. $\square$

6.4.3 Number of degree-one vertices

In this section, we prove for a point set P, its $G_\bigtriangledown(P)$ has at most three vertices of degree one. This fact is important for our proof of the lower bound of the cardinality of a maximum matching in $G_\bigtriangledown(P)$.

Definition 6.1. Let x be a degree-one vertex in $G_\bigtriangledown(P)$ and let p be the unique neighbor of x. We say that x uses the horizontal line, if x is below the horizontal line passing through p and points in $P \setminus \{p, x\}$ are all above the horizontal line passing through p. We say that x uses the $120°$ line, if x lies to the right of the $120°$ line passing through p and all points in $P \setminus \{p, x\}$ lie to the left of this line. We say that x uses the $60°$ line, if x lies to the left of the $60°$ line passing through p and all points in $P \setminus \{p, x\}$ lie to the right of this line.

Property 6.3. *Let x be a degree-one vertex in $G_{\bigtriangledown}(P)$ and let p be the unique neighbor of x such that $x \in V_i(p)$ for $i \in \{1, 2, 3\}$.*

- *If $x \in V_1(p)$, then x uses the* $120°$ *line.*
- *If $x \in V_2(p)$, then x uses the* $60°$ *line.*
- *If $x \in V_3(p)$, then x uses the horizontal line.*

Proof. To get a pictorial understanding of the property, the reader may refer to Figure 6.5. Let us consider the case when $x \in V_1(p)$. It is clear that x lies to the right of the $120°$ line passing through p. Consider a point $y \in P \setminus \{p, x\}$. By the general position assumption, y cannot lie on the $120°$ line passing through p. If y lies to the right of the $120°$ line passing through p, since x is already to the right side of the $120°$ line passing through p, the triangle $\bigtriangledown xy$ will be lying completely to the right side of the $120°$ line passing through p and therefore $p \notin \bigtriangledown xy$. Hence, by Lemma 6.2, in $G_{\bigtriangledown}(P)$ there is a path between x and y, which does not pass through p. This contradicts our assumption that p was the unique neighbor of x. Therefore, any point $y \in P \setminus \{p, x\}$ should lie to the left of the $120°$ line passing through p. Hence, x uses the $120°$ line.

When $x \in V_2(p)$ or $x \in V_3(p)$, the proofs are similar. $\square$

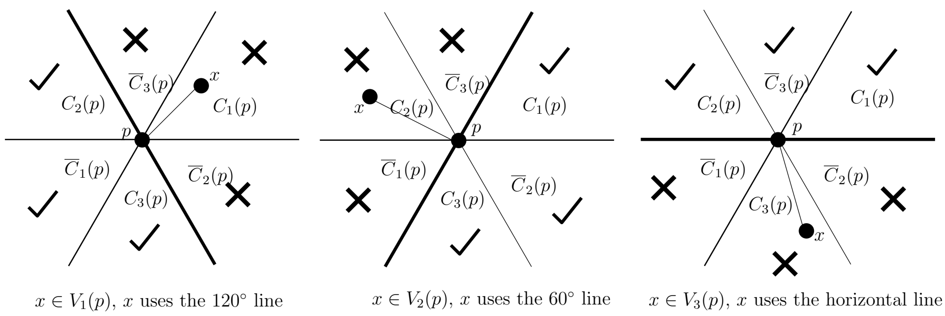
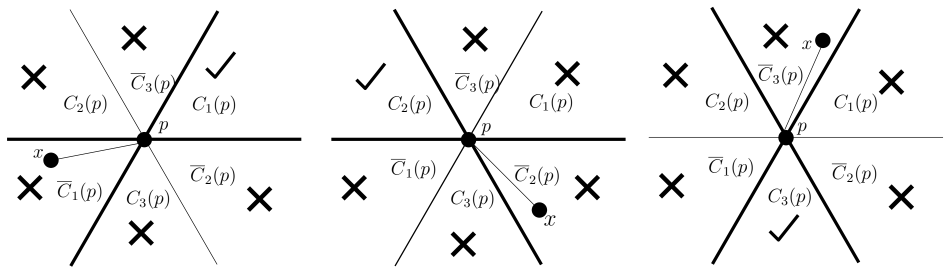


Figure 6.5: Illustration of Property 6.3. The cones around p which are allowed to have points from $P \setminus \{p, x\}$ are marked with ✓and the other cones around p are marked with $\times$.

Property 6.4. *Let x be a degree-one vertex in $G_{\bigtriangledown}(P)$ and let p be the unique neighbor of x such that $x \in \overline{V}_i(p)$ for $i \in \{1, 2, 3\}$.*



$x \in \overline{V}_1(p)$. x uses the horizontal line and the 60° line.

$x \in \overline{V}_2(p)$. x uses the horizontal line and the 120° line.

$x \in \overline{V}_3(p)$. x uses the 120° line and the 60° line.

Figure 6.6: Illustration of Property 6.4. The cones around p which are allowed to have points from $P \setminus \{p, x\}$ are marked with ✓and the other cones around p are marked with $\times$.

- *If $x \in \overline{V}_1(p)$, then x uses the horizontal line and the* 60° *line.*
- *If $x \in \overline{V}_2(p)$, then x uses the horizontal line and the* 120° *line.*
- *If $x \in \overline{V}_3(p)$, then x uses the* 60° *line and the* 120° *line.*

Proof. To get a pictorial understanding of this property, the reader may refer to Figure 6.6. This property can be proved using similar arguments as in the proof of Property 6.3. We omit the proof here, to avoid redundancy. □

Property 6.5. *Let x be a degree-one vertex in $G_\bigtriangledown(P)$ and p be the unique neighbor of x. Let $x' \in P \setminus \{x\}$ be another degree-one vertex in $G_\bigtriangledown(P)$.*

- *If x uses the horizontal line, then, x' cannot use the horizontal line.*
- *If x uses the* 60° *line, then, x' cannot use the* 60° *line.*
- *If x uses the* 120° *line, then, x' cannot use the* 120° *line.*

Proof. We prove only the first part. Proofs of the other parts are similar.

Suppose x uses the horizontal line. By definition, x lies below the horizontal line passing through p and $x' \in P \setminus \{x\}$ lies on or above above this line. This implies that x lies below the horizontal line through x'. If x' also uses the horizontal line, since $x \in P \setminus \{x'\}$, by a symmetric argument, we can show that x' lies below the horizontal line through x. Since these two conditions are not simultaneously possible, we can conclude that if x uses the horizontal line, then x' cannot use the horizontal line. □

Lemma 6.3. *For a point set P, its $G_\bigtriangledown(P)$ has at most three vertices of degree one.*

Proof. For contradiction, assume that there are four degree-one vertices x_1, x_2, x_3 and x_4 in $G_\bigtriangledown(P)$. From Property 6.3 and Property 6.4, we can see that each x_i uses at least one of the three types of reference lines: either the horizontal line, or the 60° line or the 120° line. By pigeonhole principle, at least two among these four degree-one vertices use the same type of reference line.

Without loss of generality, assume that x_1 and x_2 uses the same type of reference line. If x_1 and x_2 are adjacent to each other, these two degree-one vertices will form a connected component in $G_{\triangledown}(P)$, which will contradict the fact that $G_{\triangledown}(P)$ is connected. Therefore, x_1 and x_2 are non-adjacent. Hence, by Property 6.5, x_1 and x_2 cannot use the same type of reference line.

Therefore, we can conclude that $G_{\triangledown}(P)$ has at most three vertices of degree one. □

6.4.4 Internal triangulation

If all the internal faces of a plane graph are triangles, we call it an internally triangulated plane graph. In this section, we will prove that for a point set P, the plane graph $G_{\triangledown}(P)$ is internally triangulated. This property will be used in Section 6.5 to derive the lower bound for the cardinality of maximum matchings in $G_{\triangledown}(P)$.

Lemma 6.4. *For a point set P, all the internal faces of $G_{\triangledown}(P)$ are triangles.*

Proof. Consider an internal face f of $G_{\triangledown}(P)$. We need to show that f is a triangle. Let p be the vertex with the highest y-coordinate among the vertices on the boundary of f. Since f is an internal face, p has at least two neighbors on the boundary of f. Let q and r be the neighbors of p on the boundary of f such that r is to the right of the line passing through q and making an angle of 120° with the horizontal and any other neighbor of p on the boundary of f is to the right of the line passing through r and making an angle 120° with the horizontal. Because of the general position assumption, q and r can be uniquely determined.

We will prove that qr is also an edge on the boundary of f and there is no point from P in the interior of the triangle whose vertices are p, q and r. This will imply that the face f is the triangle whose vertices are p, q and r.

We know that $q, r \in \overline{C_1}(p) \cup \overline{C_2}(p) \cup C_3(p)$. By Property 6.2, it cannot happen that both $q, r \in \overline{C_i}(p)$, for any $i \in \{1, 2\}$. Other possibilities are shown in Figure 6.7, where q is assumed to be above r. An analogous argument can be made when r is above q as well. Since pq and pr are edges in $G_{\triangledown}(P)$, we know that $\triangledown pq \cap (P \setminus \{p, q\}) = \emptyset$ and $\triangledown pr \cap (P \setminus \{p, r\}) = \emptyset$.

Notice that, the area bounded by the lines (1) the horizontal line passing through p, (2) the line passing through q and making an angle of 120° with the horizontal, and (3) the line passing through r and making an angle of 60° with the horizontal, will define an equilateral down triangle with p, q and r on its boundary. Let us denote this triangle by $\triangledown pqr$.

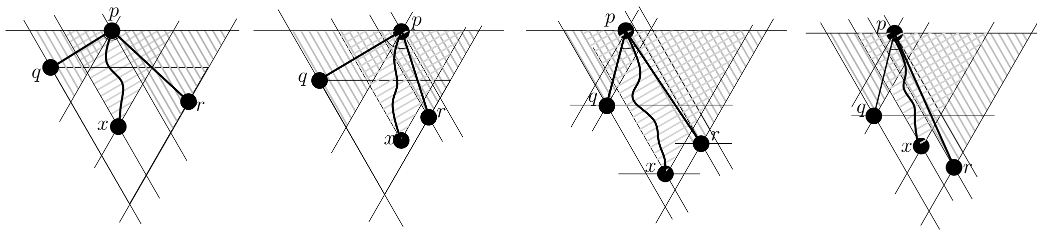


Figure 6.7: Case 1. $q \in \overline{C_1}(p)$ and $r \in \overline{C_2}(p)$, Case 2. $q \in \overline{C_1}(p)$ and $r \in C_3(p)$, Case 3. $r \in \overline{C_2}(p)$ and $q \in C_3(p)$, Case 4. $q, r \in C_3(p)$.

Claim 6.4.1. $\bigtriangledown pqr \cap (P \setminus \{p, q, r\}) = \emptyset$.

Proof. For contradiction, let us assume that there exists a point $x \in \bigtriangledown pqr \cap (P \setminus \{p, q, r\})$. Because of the general position assumption, x cannot be on the boundary of $\bigtriangledown pqr$. Therefore, $\bigtriangledown px$ does not contain q and r. By Lemma 6.2, in $G_{\bigtriangledown}(P)$, there exists a path between p and x which lies inside $\bigtriangledown px$. Let this path be $X = v_1 v_2, \ldots, v_k = x$. Since $\bigtriangledown pq \cap P \setminus \{p, q\} = \emptyset$, $\bigtriangledown pr \cap P \setminus \{p, r\} = \emptyset$ and $q, r \notin \bigtriangledown px$, we know that all vertices in the path $X = v_1 v_2, \ldots, v_k = x$ lie inside the region $R = (\bigtriangledown px \setminus (\bigtriangledown pq \cup \bigtriangledown pr)) \cup \{p\}$.

Let C be the cone with apex p bounded by the rays pq and pr. Observe that for any point $v \in R$, the line segment pv lies inside the cone C. Since $v_2 \in R$ and pv_2 is an edge (in the path from p to x), the line segment corresponding to the edge pv_2 lies inside C in $G_{\bigtriangledown}(P)$.

If the point v_2 is outside the face f, edge pv_2 will cross the boundary of f, which is contradicting the planarity of $G_{\bigtriangledown}(P)$. Since v_2 cannot be outside the face f, the edge pv_2 belongs to the boundary of f. Since v_2 lies inside the cone C and $v_2 \in R$, this means that v_2 is a neighbor of p on the boundary of f such that v_2 is to the left of the the line passing through r and making an angle of $120°$ with the horizontal. This is a contradiction to our assumption that q is the only neighbor of p on the boundary of f, lying to the left of the the line passing through r and making an angle of $120°$ with the horizontal. ☐

Let us continue with the proof of Lemma 6.4. Since the triangle with vertices p, q and r is inside the triangle $\bigtriangledown pqr$, from the above claim, it is clear that there is no point from P, other than the points p, q and r, inside the triangle whose vertices are p, q and r. Since the edges pq and pr belong to the boundary of f, to show that f is a triangle, it is now enough to prove that qr is also an edge in $G_{\bigtriangledown}(P)$. This fact also follows from the above claim as explained below.

Since $\bigtriangledown qr \subseteq \bigtriangledown pqr$, by the claim above, $\bigtriangledown qr$ cannot contain any point from P other than p, q and r. Moreover, since p lies above q and r, we know that $p \notin \bigtriangledown qr$. Therefore, $\bigtriangledown qr \cap (P \setminus \{q, r\}) = \emptyset$. Therefore, qr is an edge in $G_{\bigtriangledown}(P)$.

Thus, f has to be a triangle bounded by the edges pq, qr and pr. ☐

Corollary 6.5. *For a point set P, all the cut vertices of $G_{\bigtriangledown}(P)$ lie on its outer face.*

Proof. Consider any vertex v of $G_{\bigtriangledown}(P)$ which is not on its outer face. Since $G_{\bigtriangledown}(P)$ is internally triangulated, each neighbor of v in $G_{\bigtriangledown}(P)$ lies on a cycle in the graph $G_{\bigtriangledown}(P) \setminus v$. Since $G_{\bigtriangledown}(P)$ is connected, $G_{\bigtriangledown}(P) \setminus v$ remains connected. Thus, v cannot be a cut vertex. □

Combining Lemma 6.1, Lemma 6.2, Lemma 6.3 and Lemma 6.4, we get:

Theorem 6.6. *For a point set P, $G_{\bigtriangledown}(P)$ is a connected and internally triangulated plane graph, having at most three degree-one vertices.*

6.5 Maximum matching in $G_{\bigtriangledown}(P)$

In this section, we show that for any point set P of n points, $G_{\bigtriangledown}(P)$ contains a matching of size $\left\lceil \frac{n-1}{3} \right\rceil$; i.e, at least $2\left(\left\lceil \frac{n-1}{3} \right\rceil\right)$ vertices are matched. In order to do this, we will prove the following general statement:

Lemma 6.7. *Let G be a connected and internally triangulated plane graph, having at most three vertices of degree one. Then, G contains a matching of size at least $\left\lceil \frac{|V(G)|-1}{3} \right\rceil$.*

An overview of the proof. Let G be a graph on n vertices, satisfying the assumptions of Lemma 6.7. Since G is a connected graph, the lemma holds trivially when $n \leq 4$. Therefore, we assume that $n \geq 5$. We construct an auxiliary graph G' such that it is a 2-connected planar graph of minimum degree at least 3, and then make use of the following theorem of Nishizeki [75] to get a lower bound on the size of a maximum matching of G'.

Theorem 6.8 ([75]). *Let G' be a connected planar graph with n' vertices having minimum degree at least 3 and let M' be a maximum matching in G'. Then,*

$$|M'| \geq \begin{cases} \left\lceil \frac{n'+2}{3} \right\rceil & \text{when } n' \geq 10 \text{ and } G' \text{ is not 2-connected} \\ \left\lceil \frac{n'+4}{3} \right\rceil & \text{when } n' \geq 14 \text{ and } G' \text{ is 2-connected} \\ \left\lfloor \frac{n'}{2} \right\rfloor & \text{otherwise} \end{cases}$$

Using the above result, we will derive a lower bound on the size of a maximum matching of G.

Before getting into the proof of Lemma 6.7, it is worth mentioning that getting a weaker lower bound of $\frac{n}{3} - O(1)$ for the size of maximum matching in G is quite easy. Here we give a quick outline of the proof of this weaker bound, without getting into its details: Add a new vertex on the outer face of G and make it adjacent to all vertices which were on the outer face of G. It can be shown that, since all degree-one vertices and cut vertices of G were on its outer face, by this transformation, the resultant graph is a 2-connected planar graph of minimum degree at least two with at most three degree-two vertices. Until there are no degree-two vertices left, we do the following: we select a face of the current graph with a degree-two vertex on its boundary and place a new vertex on this face, making it adjacent to all other vertices those were on that face. It is easy to show that, after each step of this transformation, the new graph is also a 2-connected planar graph of minimum degree at least two and its number of degree-two vertices strictly lesser than that was before. Finally, we get a 2-connected planar graph G' of minimum degree at least three, on at least $n' = n + 1$ vertices, which has a maximum matching M' of size given by Theorem 6.8. To get a maximum matching M of G, we just need to delete edges in M' incident at any of the the newly added vertices of G'. Since we need to delete at most four (since the number of newly added vertices is at most four) edges from M' to get M, it is easy to show that $|M| \geq \frac{n}{3} - O(1)$.

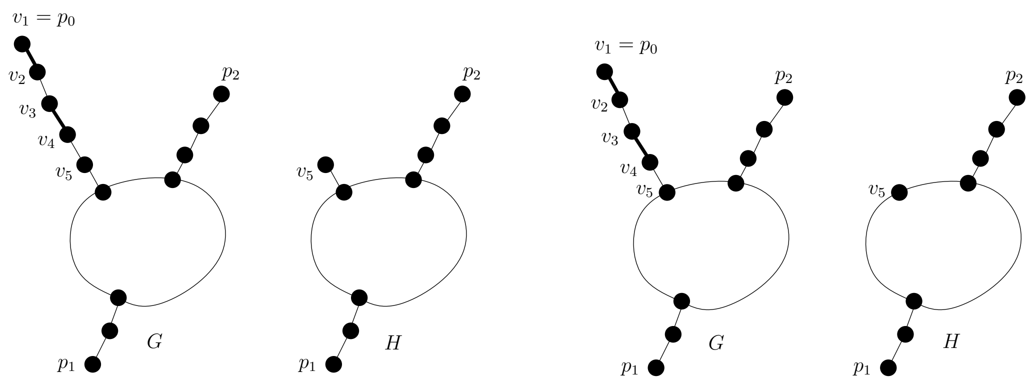
Now, our effort is to make the lower bound of $|M|$ as close to $\frac{n}{3}$ as possible. For this, we follow a slightly different method, which is described below.

Pre-processing. Let the degree-one vertices of G be denoted by p_0, p_1, $\ldots$, p_{k-1}. By our assumption, $k \leq 3$. If $k = 3$, and for each $0 \leq i \leq 2$ the unique neighbor of p_i is a degree two vertex in G, we do some pre-processing to convert it into a graph in which this condition does not hold. To understand this pre-processing easily, the reader may refer to Figure 6.8. Let $\mathcal{P}$ be the path $(p_0 = v_1, v_2, \ldots, v_{2t})$ of maximum length in G such that $\mathcal{P}$ contains an even number of vertices and $v_2, \ldots, v_{2t}$ are of degree two in G. We have $t \geq 1$. Let v_{2t+1} be the neighbor of v_{2t}, other than v_{2t-1} in G. Let H be the plane graph obtained from the plane graph G, by deleting the vertices $v_1, v_2, \ldots, v_{2t}$, along with their incident edges. It is clear that $\mathcal{P}$ has a unique maximum matching of size t and a maximum matching of G can be obtained by taking the union of a maximum matching in H and the maximum matching in $\mathcal{P}$.

Since $k = 3$ and G is connected, it is easy to see that the vertex v_{2t+1} is not a degree-one vertex in G. Since the degree of v_{2t+1} in H is one less than its degree in G, the degree of v_{2t+1} is at least one in H. By the maximality of $\mathcal{P}$, we can conclude that one of the following is true. If v_{2t+1} is a degree-one vertex in H, then, the unique neighbor of v_{2t+1} has degree at least 3 in H (as in Figure 6.8(a)). If v_{2t+1} has degree greater than one in H, then, H has at most two degree-one vertices, p_1 and p_2 (as in Figure 6.8(b)).

The properties of the path $\mathcal{P}$ ensures that H is connected. Since all the removed vertices $v_1, \ldots, v_{2t}$ were of degree less than three, they were all on the outer face of the internally triangulated graph G. Therefore, H remains internally triangulated as well.

When at least one of the degree-one vertices of G has a neighbor of degree greater than two or when $k \leq 2$ we initialize $H = G$.



(a) (b)

Figure 6.8: Pre-processing step constructing H from G. In both the cases above, the path $\mathcal{P}$=$(v_1, v_2, \ldots, v_4)$. The union of a maximum matching in H and the matching $\{(v_1, v_2), v_3, v_4)\}$ in $\mathcal{P}$ gives a maximum matching of G. (a) In G, the vertex v_5 is of degree two. It becomes a degree-one vertex in H and its neighbor has degree at least three in H. (b) In G, the vertex v_5 has degree greater than two. H has only two vertices of degree one.

From the construction of H, we can make the following observation.

Property 6.6. *H is a connected and internally triangulated plane graph. H has at most three degree-one vertices. If H has three degree-one vertices, then, one of the degree-one vertices has a neighbor of degree at least three. If M_H is a maximum matching in H, then, G has a matching of size $|M_H| + t$, where t is an integer given by $\frac{|V(G)|-|V(H)|}{2}$.*

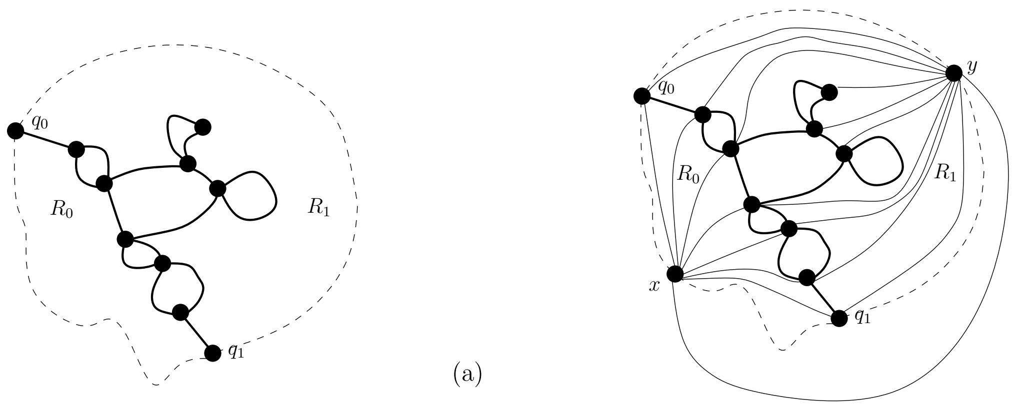
Construction of the auxiliary graph G'. Now we describe the construction of a supergraph G' of H such that G' will satisfy the assumptions of Theorem 6.8; i.e. we want G' to be a bi-connected planar graph of minimum degree at least 3. Our construction will also ensure that there exist either a single vertex v or two vertices u and v in G', such that every edge in $E(G') \setminus E(H)$ has one of its end points at u or v. Since a matching M' of G' can have at most one edge incident at each of u and v, this implies that H has a matching of size at least $M' - 2$.

We initialize G' to be the same as H. Let the degree-one vertices of H be denoted by $q_0, q_1, \ldots, q_{h-1}$. If H has no degree-one vertices, we consider h to be zero. By Property 6.6, we have $h \leq 3$. If $h = 0$ or 1, the modification of G' is simple. We insert a new vertex x in the outer face of G' and add edges between x and all other vertices which were already on the outer face of G' (i.e, add edges between the new vertex x and vertices which were on the outer face of H). This transformation maintains planarity. All vertices in G' except the vertex q_0 (present only when $h = 1$) have degree at least three now. If

$h = 1$, the degree of q_0 has become two in G' at this stage. In this case, let f be a face of the current graph G', containing both q_0 and x. Modify G' by inserting a new vertex y inside f and adding edges from this new vertex to all other vertices belonging to f. As earlier, this transformation maintains planarity. Now, the degree of q_0 becomes 3 and thus G' achieves minimum degree 3. Notice that, when $h = 0$ every edge in $E(G') \setminus E(H)$ is incident at x and when $h = 1$ every edge in $E(G') \setminus E(H)$ is incident at x or y.

If $h = 2$ or $h = 3$, consider a simple closed curve $\mathcal{C}$ in the plane such that (1) the entire graph H (all its vertices and edges) lies inside the bounded region enclosed by $\mathcal{C}$, (2) the vertices of H which lie on $\mathcal{C}$ are precisely the degree-one vertices of H, (3) except for the end points, every edge of H lies in the interior of the bounded region enclosed by $\mathcal{C}$. The region of the outer face of H, bounded by the curve $\mathcal{C}$, can be divided into h regions $R_0, \ldots, R_{h-1}$, where R_i is the region bounded by the edge at q_i, the edge at $q_{(i+1) \mod h}$ and the boundary of the outer face of H and the curve $\mathcal{C}$. (Here onwards, in this subsection we assume that indices of vertices and regions are taken modulo h). Notice that every vertex on the outer-face of H lies on at least one of these regions and q_i lies on the regions R_i and R_{i-1}, for $0 \leq i \leq h - 1$.

When $h = 2$, we insert two new vertices x, y into G'. (See Figure 6.9(a)). Three types of new edges are added in G': (1) between x and y (2) between the vertex x and all the vertices of H which lie on the region R_0 and (3) between y and all the vertices of H which lie on the region R_1. This transformation maintains planarity. (We can imagine x and y to be points on the boundary of the regions R_0 and R_1 respectively, but distinct from any point on the boundary of the outer face of H. Edges between the new vertex x and old vertices on R_0 can be drawn inside R_0 and edges between y and the old vertices on R_1 can be drawn inside R_1. The edges among the new vertices x and y can be drawn outside these regions, except at their end points). Both of the vertices q_0 and q_1 lie in both the regions R_0 and R_1. Therefore, q_0 and q_1 becomes adjacent to both x and y in G' and hence degrees of vertices q_0, q_1, x, y are all at least 3 in G'. Since H was an internally triangulated planar graph, all the degree two vertices of H were on the outer face of H. Therefore, each of them gets at least one new neighbor (x or y) in G'. Therefore, minimum degree of G' is at least 3. In this case also, every edge in $E(G') \setminus E(H)$ is incident at x or y. When $h = 3$, Property 6.6 ensures that the neighbor of one of the degree-one vertices of H has degree at least 3. Without loss of generality, assume that the neighbor of q_0 has degree at least 3 in H. In this case, we insert one new vertex x into G'. (See Figure 6.9(b)). Three types of new edges are added in G': (1) between x and q_0 (2) between q_0 and all the other vertices of H (except the unique neighbor of q_0) which were on the regions R_0 and R_2 (3) between x and all the vertices of H which were on the region R_1. This transformation also maintains planarity. (We can imagine x to be a point on the boundary of



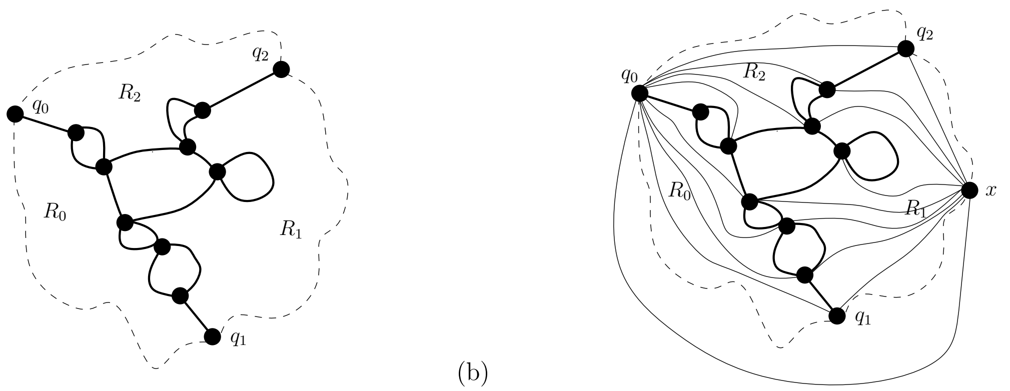


Figure 6.9: (a) Modification done when H has two degree-one vertices. Every edge in $E(G') \setminus E(H)$ is incident at x or y. (b) Modification done when H has three degree-one vertices. Every edge in $E(G') \setminus E(H)$ is incident at q_0 or x.

the region R_1, but distinct from any point on the boundary of the outer face of H. Edges between q_0 and the other vertices on R_0 can be drawn inside R_0 and edges between q_0 and the other vertices on R_2 can be drawn inside R_2. Edges between x and the other vertices on R_1 can be drawn inside R_1. The edges among the new vertices x and q_0 can be drawn outside these regions, except at their end points). Vertices q_1 and q_2 become adjacent to both q_0 and x in G'. Therefore, degrees of q_0, q_1, q_2 are at least 3. In addition, q_0 is also adjacent to x. Therefore, degree of x is also at least three in G'. Suppose vertex v was the (unique) neighbor of q_0 in H. By Property 6.6, v has degree at least three in H and hence also in G'. All degree two vertices of H, which belonged to R_0 or R_2 were non-adjacent to q_0 in H; but are adjacent to q_0 in G'. Thus, they attain degree at least 3 in G'. All degree two vertices of H, which belonged to R_2 gets a new neighbor x in G' and attain degree three. Thus, the minimum degree of G' is at least 3 in this case as well. Every edge in $E(G') \setminus E(H)$ is incident at x or q_0.

From the description above, we can make the following observation.

Property 6.7. *G' is a planar graph of minimum degree at least three, with $|V(H)| + 1 \leq |V(G')| \leq |V(H)| + 2$. There exist either a single vertex u or two vertices u and v in G', such that every edge in $E(G') \setminus E(H)$ has one of its end points at u or v.*

Claim 6.8.1. *The graph G' is 2-connected.*

Proof. In all the different cases above, it is easy to observe that none of the newly inserted vertices can be a cut vertex of G'.

Consider an arbitrary vertex $v \in V(H)$. If v is not a cut vertex of H, then, $H \setminus v$ is connected. Since G' has minimum degree at least 3, any newly added vertex has a neighbor in $V(H) \setminus \{v\}$ in the graph G'. Therefore, $G' \setminus v$ remains connected. Therefore, none of the non-cut vertices of H can be a cut vertex of G'. In particular, none of the degree-one vertices of H can be a cut vertex of G'.

If v is a cut vertex in H, v was on the outer face of H, because H was internally triangulated. It is clear that if two vertices $v_1, v_2 \in V(H)$ are in the same connected component of $H \setminus v$, they are in the same connected component of $G' \setminus v$ as well. If C_1 and C_2 are two components of $H \setminus v$, then we know that there are vertices $v_1 \in V(C_1)$ and $v_2 \in V(C_2)$, such that v_1 and v_2 are neighbors of v on the outer face of H.

When $h \leq 2$, vertices v_1 and v_2 have an edge to at least one of the newly inserted vertices in G'. Since the induced subgraph of G' on the newly inserted vertices is connected, in G' we get a path from v_1 to v_2 in which all the intermediate vertices are newly inserted vertices in G'. When $h = 3$, we have two cases to consider. It is possible that v_1 or v_2 is same as the vertex q_0 itself. If this is not the case, v_1 and v_2 have edges to either q_0 or the new vertex x in G'. In either case, since there is an edge between q_0 and x in G', we get a path from v_1 to v_2 in $G' \setminus v$. Thus, in all cases when $h \geq 3$, any two components C_1 and C_2 of $H \setminus v$ become part of the same connected component of $G' \setminus v$. Moreover, by the construction of G', the degree-one vertices of H and the vertices in $V(G') \setminus V(H)$ are part of the same component of $G' \setminus v$. This implies that $G' \setminus v$ has only a single connected component and hence, v is not a cut vertex of G'.

Thus, G' is 2-connected. □

A lower bound for the cardinality of a maximum matching in G. By Property 6.7 and Claim 6.8.1, the auxiliary graph G' is a 2-connected planar graph of minimum degree at least 3. Let $n' = |V(H)| + t_1$ be the number of vertices of G', where $t_1 = 1$ or $t_1 = 2$ by Property 6.7. By Theorem 6.8, the cardinality of a maximum matching M' in G' is at least $\left\lceil \frac{n'+4}{3} \right\rceil$ when $n' \geq 14$ and $|M'| \geq \lfloor \frac{n'}{2} \rfloor$, otherwise. Since H is a subgraph of G', if we delete the edges in M' which belong to $E(G') \setminus E(H)$, we get a matching M_H of H. Since M' is a matching in G', M' can have at most one edge incident at any vertex of G'. Hence, by Property 6.7, there can be at most two edges in $M' \cap (E(G') \setminus E(H))$. Therefore, we have $|M_H| \geq |M'| - 2$. From this, we get,

$$|M_H| \geq \begin{cases} \left\lceil \frac{|V(H)|+t_1+4}{3} \right\rceil - 2, & \text{when } |V(H)| + t_1 \geq 14 \\ \\ \left\lfloor \frac{|V(H)|+t_1}{2} \right\rfloor - 2, & \text{otherwise} \end{cases}$$

By Property 6.6, G has a matching M of size $|M_H| + t$, where t is an integer, given by $\frac{|V(G)|-|V(H)|}{2}$. By substituting the lower bound for $|M_H|$, we get,

$$|M| \geq \begin{cases} \left\lceil \frac{|V(H)|+t_1+4}{3} \right\rceil - 2 + t, & \text{when } |V(H)| + t_1 \geq 14 \\ \\ \left\lfloor \frac{|V(H)|+t_1}{2} \right\rfloor - 2 + t, & \text{otherwise} \end{cases}$$

Since $t_1 = 1$ or 2 and $t = |V(G)| - |V(H)| \geq 0$, this gives

$$|M| \geq \begin{cases} \left\lceil \frac{|V(G)|-1}{3} \right\rceil, & \text{when } |V(H)| \geq 13 \\ \\ \left\lfloor \frac{|V(G)|-3}{2} \right\rfloor, & \text{otherwise} \end{cases}$$

Whenever $|V(G)| \geq 7$, from the above inequality, we get $|M| \geq \left\lceil \frac{|V(G)|-1}{3} \right\rceil \geq 2$. Since G has at most three vertices of degree one, when $|V(G)| \geq 5$, G cannot be a star with $|V(G)| - 1$ leaves. Therefore, when $|V(G)| \geq 5$, $|M| \geq 2$. When $|V(G)| > 1$, since G is connected, we get $|M| \geq 1$. From this discussion, we can conclude that, in all cases, $|M| \geq \left\lceil \frac{|V(G)|-1}{3} \right\rceil$. This concludes the proof of Lemma 6.7.

As an immediate corollary of Lemma 6.7 and Theorem 6.6, we get:

Theorem 6.9. *For any point set P of n points in general position, $G_{\bigtriangledown}(P)$ contains a matching of size $\left\lceil \frac{n-1}{3} \right\rceil$.*

Some graphs for which our bound is tight. In Figure 6.10 (a), a point set P consisting of 15 points and the corresponding graph $G_{\bigtriangledown}(P)$ is given. This graph has a maximum matching (shown in thick lines) of size $\left\lceil \frac{|P|-1}{3} \right\rceil = 5$. This is the same example as given by Panahi et al. [76]. By adding more triplets of points (a_i, b_i, c_i), $i > 4$, into P, following the same pattern, we can show that for any $n \geq 15$ which is a multiple of 3, there is a point set P of n points in general position, such that a maximum matching in $G_{\bigtriangledown}(P)$ is of cardinality $\left\lceil \frac{|P|-1}{3} \right\rceil$. We can also show that, for any $n \geq 13$, which is one more than a multiple of three, there is a point set P' on n points in general position, such that a maximum matching in $G_{\bigtriangledown}(P')$ is of cardinality $\left\lceil \frac{|P'|-1}{3} \right\rceil$. For example, take the point set $P' = P \setminus \{a_0, b_0\}$ where P is the point set of triplets described in the paragraph above. Figure 6.10 (b) illustrates this for $n = 13$, in which

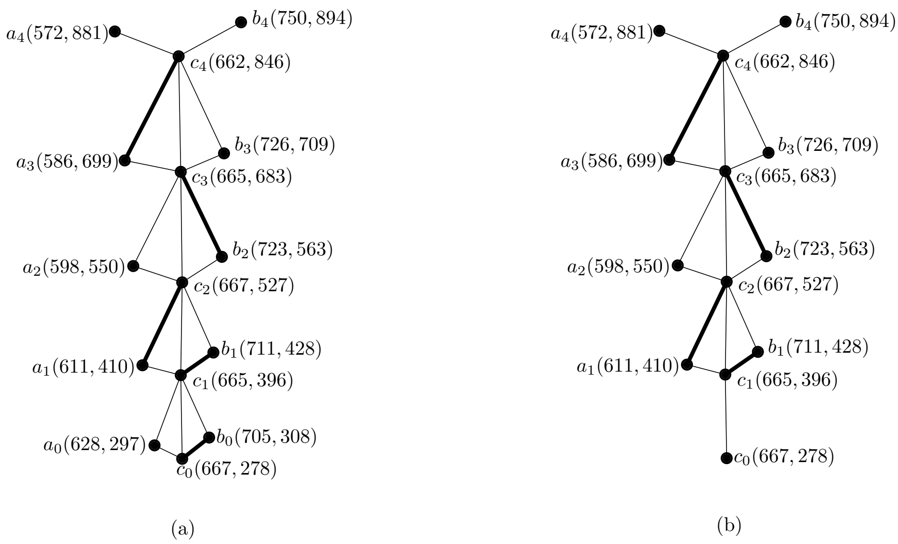


Figure 6.10: (a) A point set P with 15 points in general position, where $G_\triangledown(P)$ has a maximum matching of size $\left\lceil \frac{n-1}{3} \right\rceil = 5$ [76]. (b) A point set P with 13 points in general position, where $G_\triangledown(P)$ has a maximum matching of size $\left\lceil \frac{n-1}{3} \right\rceil = 4$.

case a maximum matching in $G_\triangledown(P')$ has cardinality $\left\lceil \frac{|P'|-1}{3} \right\rceil = 4$. Similarly, for any $n \geq 14$, which is two more than a multiple of three, there is a point set P' on n points in general position, such that a maximum matching in $G_\triangledown(P')$ is of cardinality $\left\lceil \frac{|P'|-1}{3} \right\rceil$. For example, take the point set $P' = P \setminus \{a_0\}$ where P is the point set of triplets described in the paragraph above. From the examples above, it is clear that the bound given in Theorem 6.9 is tight.

6.5.1 A 3-connected down triangle graph without perfect matching

The example given by Panahi et al. [76], for a point set P for which $G_\triangledown(P)$ has a maximum matching of size $\left\lceil \frac{n-1}{3} \right\rceil$, contained many cut vertices. However, for general planar graphs, we get a better lower bound for the size of a maximum matching, when the connectivity of the graph increases. By Theorem 6.8, we know that any 3-connected planar graph on n vertices has a matching of size $\left\lceil \frac{n+4}{3} \right\rceil$, if $n \geq 14$ and has a matching of size $\left\lfloor \frac{n}{2} \right\rfloor$ if $n < 14$ or it is 4-connected. Hence, it was interesting to see whether there exist a point set P in general position, with an even number of points, such that $G_\triangledown(P)$ is 3-connected but does not contain a perfect matching. The answer is positive. Consider the graph given in Figure 6.11 (a), which shows a point set P of 18 points in general position and the corresponding graph $G_\triangledown(P)$. This graph has a maximum matching (shown in thick lines) of size 8. We can follow the pattern and go on adding points a_i, b_i and c_i, for $i > 4$ to the point set such that when

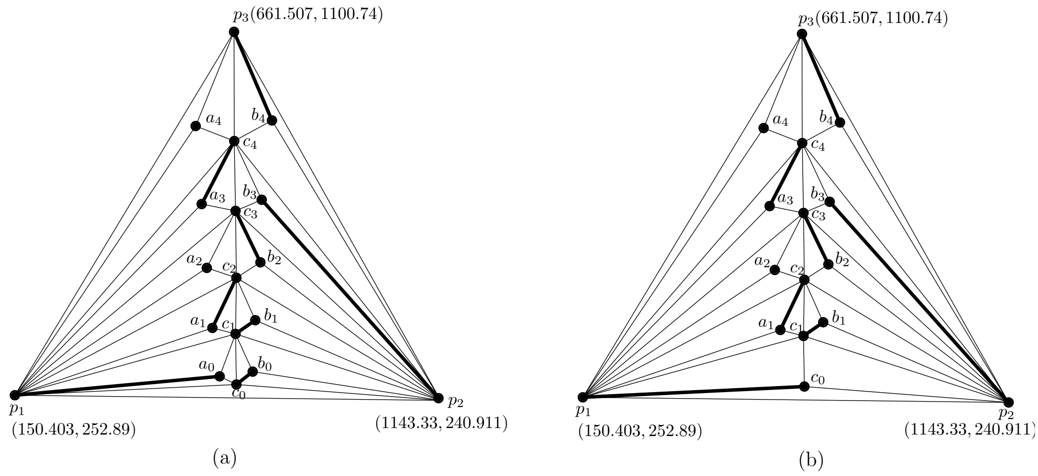


Figure 6.11: (a) A point set P with 18 points in general position, where $G_{\bigtriangledown}(P)$ is 3-connected and has a maximum matching of size $\left\lceil \frac{n+5}{3} \right\rceil$. (b) A point set P with 16 points in general position, where $G_{\bigtriangledown}(P)$ is 3-connected and has a maximum matching of size $\left\lceil \frac{n+5}{3} \right\rceil$. The points with their co-ordinates unspecified have the same co-ordinates as in Figure 6.10.

$P = \{a_0, b_0, c_0, \ldots, a_k,\ b_k,\ c_k,\ p_1,\ p_2,\ p_3\}$, $G_{\bigtriangledown}(P)$ is a 3-connected graph with a maximum matching of size $\left\lceil \frac{|P|+5}{3} \right\rceil$. It can be verified that $G_{\bigtriangledown}(P \setminus \{a_0\})$ and $G_{\bigtriangledown}(P \setminus \{a_0, b_0\})$ are also 3-connected and their maximum matchings have size $\left\lceil \frac{|P|+5}{3} \right\rceil$. (See Figure 6.11 (b) for the case when $|P| = 16$). Thus, for 3-connected down triangle graphs corresponding to point sets in general position, the best known lower bound for maximum matching is $\left\lceil \frac{n+4}{3} \right\rceil$ and the examples we discussed above show that it is not possible to improve the bound above $\left\lceil \frac{n+5}{3} \right\rceil$.

6.6 Some properties of $G_{\davidsstar}(P)$

In this section, we prove that for a point set P, the 2-connectivity structure of $G_{\davidsstar}(P)$ is simple and $G_{\davidsstar}(P)$ can have at most $5n - 11$ edges.

6.6.1 Block cut point graph

Let $G(V, E)$ be a graph. A block of G is a maximal connected subgraph having no cut vertex. The block cut point graph of G is a bipartite graph $B(G)$ whose vertices are cut-vertices of G and blocks of G, with a cut-vertex x adjacent to a block X if x is a vertex of block X. The block cut point graph of G gives information about the 2-connectivity structure of G.

Since $G_{\davidsstar}(P)$ is the union of two connected graphs $G_{\bigtriangledown}(P)$ and $G_{\bigtriangleup}(P)$ (Lemma 6.2), it is connected and hence its block-cut point graph is a tree [40]. We will show that the block cut point graph of $G_{\davidsstar}(P)$ is a simple path. We use the following lemma in our proof.

Lemma 6.10. *Let P be a point set and $p \in P$ be a cut vertex of $G_{\davidsstar}(P)$. Then, there exists an $i \in \{1, 2, 3\}$ such that $V_i(p) \neq \emptyset$, $\overline{V_i}(p) \neq \emptyset$ and for all $j \in \{1, 2, 3\} \setminus \{i\}$, $V_j(p) = \emptyset$ and $\overline{V_j}(p) = \emptyset$. Moreover, $G_{\davidsstar}(P) \setminus p$ has exactly two connected components, one containing all vertices in $V_i(p)$ and the other containing all vertices of $\overline{V_i}(p)$.*

Proof. Since p is a cut vertex of $G_{\davidsstar}(P)$, we know that there exist $v_1, v_2 \in P$ that are in different components of $G_{\davidsstar}(P) \setminus p$. We will show that v_1 and v_2 should be in opposite cones with reference to the apex point p.

Without loss of generality, assume that $v_1 \in A_1(p) \cap P \setminus \{p\}$. If $v_2 \in (A_1(p) \cup A_2(p) \cup A_6(p)) \cap (P \setminus \{p\})$, then, $p \notin \bigtriangledown v_1v_2$ and hence by Lemma 6.2, there is a path in $G_{\bigtriangledown}(P)$ between v_1 and v_2 that does not pass through p, which is not possible. Similarly, if $v_2 \in (A_3(p) \cup A_5(p)) \cap (P \setminus \{p\})$, then, $p \notin \bigtriangleup v_1v_2$ and there is a path in $G_{\bigtriangleup}(P)$ between v_1 and v_2 that does not pass through p, which is not possible. Therefore, $v_2 \in A_4(p)$, the cone which is opposite to $A_1(p)$ which contains v_1. Thus any two points v_1 and v_2 which are in different connected components of $G_{\davidsstar}(P) \setminus p$, are in opposite cones around p.

Let C_1 and C_2 be two connected components of $G_{\davidsstar}(P) \setminus p$ with $v_1 \in C_1$ and $v_2 \in C_2$. Without loss of generality, assume that such $v_1 \in V_1(p)$ and $v_2 \in \overline{V_1}(p)$. From the paragraph above, we know that every vertex of $G_{\davidsstar}(P) \setminus p$ which is not in C_1 is in $\overline{V_1}(p)$ and every vertex of $G_{\davidsstar}(P) \setminus p$ which is not in C_2 is in $V_1(p)$. This implies that for all $j \in \{2, 3\}$, $V_j(p) = \emptyset$ and $\overline{V_j}(p) = \emptyset$. This proves the first part of our lemma.

For any $v_1, v_2 \in \overline{V_i}(p)$, we have $p \notin \bigtriangledown v_1v_2$ and hence by Lemma 6.2, there is a path in $G_{\bigtriangledown}(P)$ between v_1 and v_2 that does not pass through p. Similarly, for any $v_1, v_2 \in V_i(p)$, $p \notin \bigtriangleup v_1v_2$ and there is a path in $G_{\bigtriangleup}(P)$ between v_1 and v_2 that does not pass through p. Therefore, there are exactly two connected components in $G_{\davidsstar}(P) \setminus p$, one containing all vertices in $V_i(p)$ and the other containing all vertices of $\overline{V_i}(p)$. $\square$

Theorem 6.11. *Let P be a point set in general position and let k be the number of blocks of $G_{\davidsstar}(P)$. Then, the blocks of $G_{\davidsstar}(P)$ can be arranged linearly as $B_1, B_2, \ldots B_k$ such that, for $i > j$, $B_i \cap B_j$ contains a single (cut) vertex p_i when $j = i + 1$ and $B_i \cap B_j$ is an empty graph otherwise. That is, the block cut point graph of $G_{\davidsstar}(P)$ is a path.*

Proof. If $G_{\davidsstar}(P)$ is two-connected, there is only a single block and the lemma is trivially true.

Since $G_{\davidsstar}(P)$ is a connected graph, its block cut point graph is a tree. Any two blocks can have at most one vertex in common and the common vertex is a cut vertex. From Lemma 6.10, we also know that three or more blocks cannot share a common (cut) vertex. If a block B_i of $G_{\text{✡}}(P)$ is such that, in the block cut point graph of $G_{\text{✡}}(P)$, the node corresponding to block B_i is a leaf node, B_i is adjacent to only one another block and they share a single (cut) vertex.

If the node corresponding to B_i is not a leaf node of the block cut point graph, we know that B_i shares (distinct) common vertices with at least two other blocks $B_{i'}$ and $B_{i''}$. Therefore, two vertices in B_i are cut vertices of $G_{\text{✡}}(P)$. Let v_1, v_2 be these cut vertices. We will show that there cannot be a third such cut vertex in B_i.

By Lemma 6.10, we know that $G_{\text{✡}}(P) \setminus v_1$ has exactly two components and since B_i is 2-connected initially, all vertices of B_i except v_1 are in the same connected component of $G_{\text{✡}}(P) \setminus v_1$. By Lemma 6.10, all vertices of B_i lie in the same (designated) cone with apex v_1. Without loss of generality, assume that all vertices in $B_i \setminus v_1$ are in $V_1(v_1)$. In particular, $v_2 \in V_1(v_1)$ and hence $v_1 \in \overline{V_1}(v_2)$. Similarly, since v_2 is a cut vertex, all vertices of B_i lie in the same (designated) cone with apex v_2. Since $v_1 \in \overline{V_1}(v_2)$, all vertices in $B_i \setminus v_2$ are in $\overline{V_1}(v_2)$. If v_3 is a vertex in B_i, distinct from v_1 and v_2, then from the discussion above, we get $v_3 \in V_1(v_1)$ and $v_3 \in \overline{V_1}(v_2)$. Hence $v_1 \in \overline{V_1}(v_3)$ and $v_2 \in V_1(v_3)$. Suppose v_3 is a cut vertex in $G_{\text{✡}}(P)$. Since v_1 and v_2 are in the same connected component of $G_{\text{✡}}(P) \setminus v_3$, it is a contradiction to Lemma 6.10, that $v_1 \in \overline{V_1}(v_3)$ and $v_2 \in V_1(v_3)$.

Thus, if the node corresponding to B_i is not a leaf node of the block cut point graph of $G_{\text{✡}}(P)$, then exactly two vertices in B_i are cut vertices of $G_{\text{✡}}(P)$. Since no three blocks can share a common vertex by Lemma 6.10, we are done. $\square$

6.6.2 Number of Edges of $G_{\text{✡}}(P)$

Since $G_{\triangledown}(P)$ and $G_{\triangle}(P)$ are planar graphs and $G_{\text{✡}}(P) = G_{\triangledown}(P) \cup G_{\triangle}(P)$, using Euler's theorem, it is obvious that $G_{\text{✡}}(P)$ has at most $2 \times (3n-6) = 6n-12$ edges, where $n = |P|$ [40]. In this section, we show that for any point set P, its $G_{\text{✡}}(P)$ has a spanning tree of a special structure, which will imply that $G_{\text{✡}}(P)$ can have at most $5n-11$ edges.

Lemma 6.12. *For a point set P, the intersection of $G_{\triangledown}(P)$ and $G_{\triangle}(P)$ is a connected graph.*

Proof. We will prove this algorithmically. At any point of execution of this algorithm, we maintain a partition of P into two sets S and $P \setminus S$ such that the induced subgraph of $G_{\triangledown}(P) \cap G_{\triangle}(P)$ on S is connected. When the algorithm terminates, we will have $S = P$, which will prove the lemma.

We start by adding any arbitrary point $p_1 \in P$ to S. The induced subgraph of $G_{\triangledown}(P) \cap G_{\triangle}(P)$ on S is trivially connected now.

At any intermediate step of the algorithm, let $S = \{p_1, p_2, \ldots, p_k\} \neq P$, such that the invariant is true. We will show that we can add a point p_{k+1} from $P \setminus S$ into S, and still maintain the invariant.

For any point $p \in S$, let

$$d_1(p) = \min_{i\in\{1,2,3\}, p'\in V_i(p)\cap P\setminus S} c_i(p, p')$$

$$d_2(p) = \min_{i\in\{1,2,3\}, p'\in \overline{V_i}(p)\cap P\setminus S} \overline{c_i}(p, p')$$

and

$$d(p) = \min(d_1(p), d_2(p))$$

Since $|P \setminus S| \geq 1$, $d(p) < \infty$. Let $d = \min_{p\in S} d(p)$.

Consider $p \in S$ such that $d(p) = d$. By definition of d, such a point exists. Consider the area enclosed by the hexagon around p which is defined by $H_p = \bigcup_{i=1}^{3}\{p' \in C_i(p) \mid c_i(p, p') \leq d\} \cup \bigcup_{i=1}^{3}\{p' \in \overline{C_i}(p) \mid \overline{c_i}(p, p') \leq d\}$. (See Figure 6.12 (a)). We know that there exists a point $q \in P \setminus S$ such that q is on the boundary of H_p. We claim that pq is an edge in $G_{\bigtriangledown}(P) \cap G_{\triangle}(P)$.

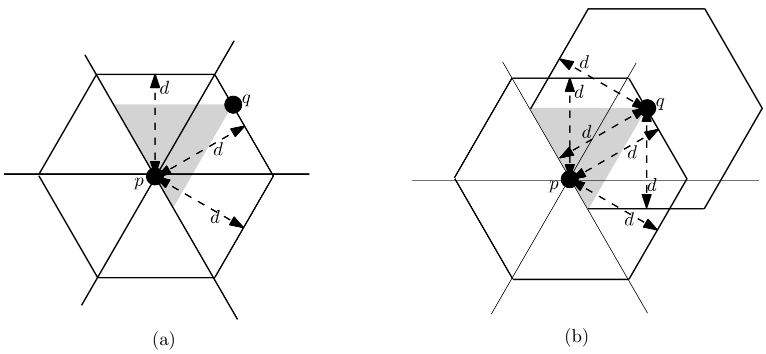


Figure 6.12: (a) Closest point to p. (b) Hexagons around closest pairs.

Let $H_q = \bigcup_{i=1}^{3}\{p' \in C_i(q) \mid c_i(q, p') \leq d\} \cup \bigcup_{i=1}^{3}\{p' \in \overline{C_i}(q) \mid \overline{c_i}(q, p') \leq d\}$, which is a hexagonal area around q. (See Figure 6.12 (b)). Without loss of generality, assume that $q \in C_1(p)$. Note that, by Property 6.1, $c_1(p, q) = \overline{c_1}(q, p) = d$ and hence, $\bigtriangledown pq \cup \triangle pq \subseteq H_p \cap H_q$.

If there exists a point $q' \in (P \setminus \{q\}) \setminus S$ such that q' lies in the interior of H_p, then $d(p) < d$, which is a contradiction. Similarly, if there exists a point $p' \in (P \setminus \{p\}) \cap S$ such that p' lies in the interior of H_q, then $d(p) < d$. This is also a contradiction. Therefore, $H_p \cap H_q \cap (P \setminus \{p, q\}) = \emptyset$. Since, $\bigtriangledown pq \cup \triangle pq \subseteq H_p \cap H_q$, this implies that $\bigtriangledown pq \cap (P \setminus \{p, q\}) = \emptyset$ and $\triangle pq \cap (P \setminus \{p, q\}) = \emptyset$. This implies that pq is an edge in $G_{\bigtriangledown}(P)$ as well as in $G_{\triangle}(P)$.

Since pq is an edge in $G_{\bigtriangledown}(P) \cap G_{\triangle}(P)$, we can add $p_{k+1} = q$ to the set S, thus increasing the cardinality of S by one, and still maintaining the invariant that the induced subgraph of $G_{\bigtriangledown}(P) \cap G_{\bigtriangleup}(P)$ on S is connected. Since we can keep on doing this until $S = P$, we conclude that $G_{\bigtriangledown}(P) \cap G_{\bigtriangleup}(P)$ is connected. □

Theorem 6.13. *For a set P of n points in general position, $G_{\text{✡}}(P)$ has at most $5n - 11$ edges and hence its average degree is less than 10.*

Proof. Since $G_{\bigtriangledown}(P)$ and $G_{\bigtriangleup}(P)$ are both planar graphs we know that each of them can have at most $3n - 6$ edges. From Lemma 6.12, we know that the intersection of $G_{\bigtriangledown}(P)$ and $G_{\bigtriangleup}(P)$ contains a spanning tree and hence they have at least $n-1$ edges in common. From this, we conclude that the number of edges in $G_{\text{✡}}(P) = G_{\bigtriangledown}(P) \cup G_{\bigtriangleup}(P)$ is at most $(3n-6)+(3n-6)-(n-1) = 5n - 11$. Hence,the average degree of $G_{\text{✡}}(P)$ is less than 10. □

Corollary 6.14. *For a set P of n points in general position, its Θ_6 graph has at most $5n - 11$ edges.*

It is still an open problem to decide whether the upper bound on the number of edges, stated in Theorem 6.13 and Corollary 6.14, is tight. Here we give an example showing that this upper bound cannot be improved below $\left(4 + \frac{1}{3}\right) n - 13$. In Figure 6.13, a point set P of 18 points and the corresponding $G_{\text{✡}}(P)$ graph is shown. This graph has 65 edges. By varying the number of triplets of points (a_i, b_i, c_i), $i \geq 0$, in P, following the same pattern, we can show that for any $n \geq 6$ which is a multiple of 3, there is a point set P of n points in general position, such that $G_{\text{✡}}(P)$ has exactly $\left(4 + \frac{1}{3}\right) n - 13$ edges.

6.7 Conclusion

We have shown that for any set P of n points in general position, any maximum $\bigtriangledown$ (resp. $\bigtriangleup$) matching of P will match at least $2\left(\left\lceil \frac{|P|-1}{3} \right\rceil\right)$ points. This also implies that any half-Θ_6 graph (or equivalently TD - Delaunay graph) for point sets in general position has a matching of size at least $\left\lceil \frac{|P|-1}{3} \right\rceil$. We have also given examples for which this bound is tight. This is in contrast with the case of classical Delaunay graphs, where the size of the maximum matching is always $\left\lfloor \frac{|P|}{2} \right\rfloor$, for non-degenerate point sets. We also proved that when P is in general position, the block cut point graph of its Θ_6 graph is a simple path and that the Θ_6 graph has at most $5n - 11$ edges. It is an interesting question to see whether for every point set in general position, its Θ_6 graph contains a matching of size $\left\lfloor \frac{|P|}{2} \right\rfloor$. So far, we were not able to get any counter examples for this claim and hence we conjecture the following.

Conjecture 6.15. *For every set of n points in general position, its Θ_6 graph contains a matching of size $\left\lfloor \frac{n}{2} \right\rfloor$.*

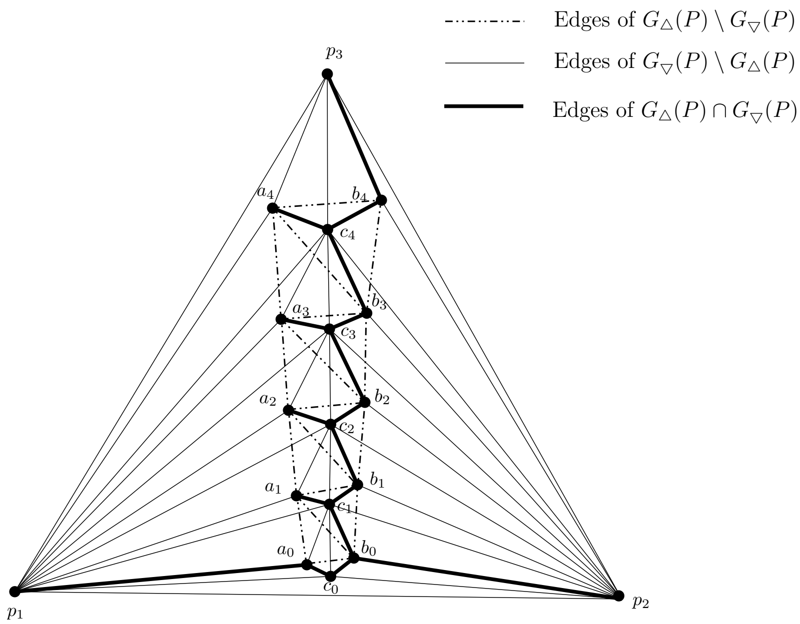


Figure 6.13: A point set P of $n = 18$ points and the corresponding $G_{\text{✡}}(P)$ graph with $\left(4 + \frac{1}{3}\right) n - 13 = 65$ edges.

Chapter 7

Heterochromatic paths in edge colored graphs

In this chapter[1] we give lower bounds for the length of a maximum heterochromatic path in edge colored graphs without small cycles. We show that if G has no four cycles, then it contains a heterochromatic path of length at least $\vartheta(G) - o(\vartheta(G))$ and if the girth of G is at least $4\log_2(\vartheta(G)) + 2$, then it contains a heterochromatic path of length at least $\vartheta(G) - 2$, which is only one less than the bound conjectured by Chen and Li [32] for the general case. Other special cases considered include lower bounds for the length of a maximum heterochromatic path in edge colored bipartite graphs and triangle-free graphs: for triangle-free graphs we obtain a lower bound of $\left\lfloor \frac{5\vartheta(G)}{6} \right\rfloor$ and for bipartite graphs we obtain a lower bound of $\left\lceil \frac{6\vartheta(G)-3}{7} \right\rceil$.

We also prove that if the coloring is such that G has no heterochromatic triangles, then G contains a heterochromatic path of length at least $\left\lfloor \frac{13\vartheta(G)}{17} \right\rfloor$. This improves the previously known $\left\lceil \frac{3\vartheta(G)}{4} \right\rceil$ bound obtained by Chen and Li [34]. We also give a relatively shorter and simpler proof showing that any edge colored graph G contains a heterochromatic path of length at least $\left\lceil \frac{2\vartheta(G)}{3} \right\rceil$.

7.1 Introduction

An edge coloring of a graph is a mapping from its edge set to the set of natural numbers. If a graph G has an edge coloring specified, we call G an edge colored graph. The length of a path P is the number of edges of the path P. Unless specified otherwise, our graphs are finite simple graphs.

[1]Joint work with L. Sunil Chandran and Deepak Rajendraprasad. Communicated to European Journal of Combinatorics.

Let $G(V, E)$ be an edge colored graph. We use color(e) to denote the color given to an edge $e \in E$. (To denote the color given to an edge $(u, v) \in E$, we abuse the above notation and write color(u, v).) A heterochromatic or a rainbow subgraph in G is a subgraph H of G such that for every pair of distinct edges e_1 and e_2 of H, we have color$(e_1) \neq$ color(e_2).

The conditions for the existence of large heterochromatic subgraphs in edge colored graphs are well studied in literature [59, 51, 58, 63]. Erdos et al.[44], Hahn et al. [59] and Albert et al. [8] gave some sufficient conditions on the coloring to guarantee a heterochromatic Hamiltonian cycle in an edge colored complete graph K_n. The conditions for the existence of heterochromatic Hamiltonian paths in infinite complete graphs were studied by Hahn and Thomassen [59] and later by Erdos and Tuza [45].

The number of distinct colors occurring at edges incident at a vertex v of G is called the color degree of v and is denoted by $deg^c(v)$. We use $\vartheta(G)$ to denote the minimum color degree of G, i.e., $\vartheta(G) = \min_{v \in V(G)} deg^c(v)$. Broersma et al. [19] obtained lower bounds for the length of a maximum heterochromatic path in an edge colored graph, in terms of its minimum color degree and minimum neighborhood union conditions. We use $\lambda(G)$ to denote the length of a maximum length heterochromatic path in G. They showed that for every vertex v of G, there exists a heterochromatic path starting at v and of length at least $\left\lceil \frac{\vartheta(G)+1}{2} \right\rceil$. They also showed that if for every pair of vertices x and y of G, the cardinality of the union of the colors given to edges incident with x and y is at least s, then $\lambda(G) \geq \left\lceil \frac{s}{3} \right\rceil + 1$.

Chen and Li [32] reported A. Saito's conjecture that $\lambda(G) \geq \left\lceil \frac{2\vartheta(G)}{3} \right\rceil$ for any edge colored graph G. They showed that $\lambda(G) \geq \vartheta(G) - 1$, if $3 \leq \vartheta(G) \leq 7$, and $\lambda(G) \geq \left\lceil \frac{3\vartheta(G)}{5} \right\rceil + 1$, if $\vartheta(G) \geq 8$. It is easy to see that if $\vartheta(G) = 1$ or 2, then $\lambda(G) \geq \vartheta(G)$. In the same paper, they conjectured that the actual bound could be $\vartheta(G) - 1$ and demonstrated some examples which achieve this bound. Recently, Das et al. [39] gave a simpler and shorter proof showing that $\lambda(G) \geq \left\lceil \frac{3\vartheta(G)}{5} \right\rceil$ for any edge colored graph G.

In an unpublished manuscript from Chen and Li [31], it was shown that if $\vartheta(G) \geq 8$, then $\lambda(G) \geq \left\lceil \frac{2\vartheta(G)}{3} \right\rceil + 1$. Further, in another work [33], they showed that if for every pair of vertices x and y of G, the cardinality of the union of the colors given to edges incident with x and y is at least s, then $\lambda(G) \geq \left\lceil \frac{s+1}{2} \right\rceil$. This was an improvement over the result of Broersma et al. [19]. Later, they [34] also showed that, if the coloring is such that G has no heterochromatic triangles, then $\lambda(G) \geq \left\lceil \frac{3\vartheta(G)}{4} \right\rceil$.

The results in this chapter include the following.

- We give a shorter and simpler proof (compared to those in [32, 31, 39]) showing that for any edge colored graph G, $\lambda(G) \geq \left\lceil \frac{2\vartheta(G)}{3} \right\rceil$.
- If G is an edge-colored triangle-free graph, then $\lambda(G) \geq \left\lfloor \frac{5\vartheta(G)}{6} \right\rfloor$.

- If G is edge colored and is bipartite, then $\lambda(G) \geq \left\lceil \frac{6\vartheta(G)-3}{7} \right\rceil$.

- If G is an edge colored graph without cycles of length four, then $\lambda(G)$ is $\vartheta(G) - o(\vartheta(G))$.

- If G is an edge colored graph without cycles of length less than g, with $g \geq 5$, then $\lambda(G) \geq \left\lceil (\vartheta(G) - 1) - \vartheta(G)^{\frac{g}{\lceil \frac{g}{4} \rceil (g-2)}} \right\rceil$. Note that, when $g = 4\log_2(\vartheta(G)) + 2$, this lower bound reaches $\vartheta - 2$. Thus, in the case of graphs without small cycles, our lower bound is only one less than the $\vartheta(G) - 1$ lower bound conjectured by Chen and Li [32].

- When the girth of G is less than 9, we use some other methods and obtain better lower bounds for $\lambda(G)$, compared to the general bound stated above.

- If the coloring is such that G has no heterochromatic triangles, then $\lambda(G) \geq \left\lfloor \frac{13\vartheta(G)}{17} \right\rfloor$. This is an improvement over the bound obtained by Chen and Li [34].

7.2 A bound for the length of maximum heterochromatic paths

As we mentioned in the introduction, in an unpublished manuscript, Chen and Li reported to prove that if $\vartheta = \vartheta(G) \geq 8$, then $\lambda(G) \geq \left\lceil \frac{2\vartheta}{3} \right\rceil + 1$. In this section, we give a shorter and simpler proof showing that for any edge colored graph G, $\lambda(G) \geq \left\lceil \frac{2\vartheta}{3} \right\rceil$. The ideas used in this proof are refinements of the ideas used for obtaining the $\left\lceil \frac{3\vartheta}{5} \right\rceil$ bound in Das et al. [39]; but we are able to achieve a much stronger result.

If H is a subgraph of G, we use $C(H)$ to denote the colors that appear on edges belonging to the subgraph H. For a vertex $v \in V(G)$, $N(v)$ denotes the set of neighbors of v in G. For a subset $S \subseteq V(G)$, let $N(S) = \bigcup_{v \in S} N(v)$.

Lemma 7.1. *Let G be an edge colored graph and let P be a maximum length heterochromatic path in G. Suppose x is an endpoint of P. If x has a neighbor v such that* $\text{color}(x, v) \notin C(P)$*, then $v \in V(P)$.*

Proof. Suppose P is given by $x = u_0, u_1, \ldots, u_t = y$. If x has a neighbor $v \notin V(P)$ such that $\text{color}(x, v) \notin C(P)$, then $v, u_0, u_1, \ldots, u_t$ will be a heterochromatic path in G which is longer than P, which is a contradiction. $\square$

In the remaining parts of this chapter, we repeatedly use the definitions given below.

Definition 7.1. Let P be a maximum length heterochromatic path in G. Let P be of length t and be given by $x = u_0, u_1, \ldots, u_t = y$. Recall the definition of $C(P)$ we made at the beginning of this section. We call the colors in $C(P)$ as *old colors* and other colors as *new colors*. We define

- $OLD_{\notin y} = \{c \in C(P) \mid$ no edge incident at y has color $c\}$.
- $OLD_{y \to P} = \{c \in C(P) \mid y$ has a neighbor $u_i \in V(P)$ such that $\text{color}(y, u_i) = c\}$.
- $OLD_{y \nrightarrow P} = C(P) \setminus (OLD_{\notin y} \cup OLD_{y \to P})$. Clearly, if $c \in OLD_{y \nrightarrow P}$, then y has a neighbor $z \notin V(P)$ such that $\text{color}(y, z) = c$.
- $NEW_{y \to P} = \{c \in C(G) \setminus C(P) \mid y$ has a neighbor of $u_i \in V(P)$ such that $\text{color}(y, u_i) = c\}$.

Note that $OLD_{\notin y} \uplus OLD_{y \to P} \uplus OLD_{y \nrightarrow P} = C(P)$ and the cardinality of this set is t, because P is heterochromatic.

Lemma 7.2. *Let P be a maximum length heterochromatic path in an edge colored graph G. Suppose P is of length t and y is an endpoint of P. Let $COLOR_{y \to P} = \{\text{color}(y, u_i) \mid u_i \in N(y) \cap V(P)\}$. Then, $|COLOR_{y \to P}|$ is at least $\vartheta - t + |OLD_{\notin y}| + |OLD_{y \to P}|$. Consequently, the total number of neighbors of y in P, $|N(y) \cap V(P)| \geq \vartheta - t + |OLD_{\notin y}| + |OLD_{y \to P}|$.*

Proof. Clearly, $|COLOR_{y \to P}| = |NEW_{y \to P}| + |OLD_{y \to P}|$. By Lemma 7.1, if an edge (y, v) is of a new color, $v \in V(P)$. This implies that $|NEW_{y \to P}| \geq \vartheta - t + |OLD_{\notin y}|$, because y has at least $\vartheta(G)$ colors incident on it and there are only $t - |OLD_{\notin y}|$ old colors among them. Therefore, we have $|COLOR_{y \to P}| = |NEW_{y \to P}| + |OLD_{y \to P}| \geq \vartheta - t + |OLD_{\notin y}| + |OLD_{y \to P}|$ and the statement of the lemma follows. $\square$

Definition 7.2. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. We define $T_P(x) = \{u_i \in N(x) \cap V(P) \mid \text{color}(x, u_i) \notin C(P)\}$ and $M_P(x) = \{u_i \mid u_i$ is the predecessor of a vertex in $T_P(x)$ in the path P from x to $y\}$.

The following observation directly follows from Definition 7.2, by Lemma 7.1.

Lemma 7.3. *Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. Then, $|M_P(x)| = |T_P(x)| \geq \vartheta - t$.*

Lemma 7.4. *Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. Suppose $u_i \in M_P(x)$. Then* $\text{color}(u_i, u_{i+1})$ *belongs to $OLD_{\notin y} \uplus OLD_{y \to P}$.*

Proof. Suppose color$(u_i, u_{i+1}) \notin OLD_{\notin y} \uplus OLD_{y \to P}$. Then, by Definition 7.1, color$(u_i, u_{i+1}) \in OLD_{y \not\to P}$ and y has a neighbor $z \notin V(P)$ such that color(y, z) = color(u_i, u_{i+1}). Then, the path $u_i, u_{i-1}, \ldots, u_0 = x, u_{i+1}, \ldots, u_t = y, z$ is a heterochromatic path in G longer than P, a contradiction. □

Lemma 7.5. *Let P be a maximum length heterochromatic path in G. Let P be of length t and x and y be the endpoints of P. Then, $|N(x) \cap V(P)| \geq 2(\vartheta - t)$ and $|N(y) \cap V(P)| \geq 2(\vartheta - t)$.*

Proof. By Lemma 7.3 and Lemma 7.4, $|OLD_{\notin y} \uplus OLD_{y \to P}| \geq |M_P(x)| \geq \vartheta - t$. Therefore, by Lemma 7.2, $|N(y) \cap V(P)| \geq 2(\vartheta - t)$. By symmetric arguments, we can prove that $|N(x) \cap V(P)| \geq 2(\vartheta - t)$. □

Theorem 7.6. *For any edge colored graph G, there exists a heterochromatic path of length at least $\left\lceil \frac{2\vartheta}{3} \right\rceil$ in G.*

Proof. Let P be a maximum heterochromatic path in G. Suppose P is of length t and y is an endpoint of P. Then, by Lemma 7.5, $|N(y) \cap V(P)| \geq 2(\vartheta - t)$. Since $t \geq |N(y) \cap V(P)|$, we get $t \geq 2(\vartheta - t)$. From this, the statement of the theorem follows. □

7.3 Maximum heterochromatic paths in edge colored graphs without short cycles

In this section we obtain lower bounds for the length of a maximum heterochromatic path in an edge colored graph without short cycles. Special cases considered include triangle free graphs, bipartite graphs and graphs without four cycles. The important result in this section is a lower bound for $\lambda(G)$ as a function of the girth of G and ϑ. As the girth increases, our lower bound becomes closer and closer to $\vartheta - 2$, which is just one less than the bound conjectured by Chen et al. [32]. We extend the ideas developed in the previous section before proceeding further.

Lemma 7.7. *Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. Then, each $u_i \in M_P(x)$ is the end point of a maximum heterochromatic path P_i in G, such that $V(P_i) = V(P)$ and $C(P_i) = C(P) \cup \{\text{color}(x, u_{i+1})\} \setminus \{\text{color}(u_i, u_{i+1})\}$.*

Proof. Let $u_i \in M_P(x)$. By the definition of $M_P(x)$, $u_{i+1} \in N(x)$ and color$(x, u_{i+1}) \notin C(P)$. Note that $i > 0$, because color$(x, u_1) \in C(P)$. The path $u_i, u_{i-1}, \ldots, u_0 = x, u_{i+1}, \ldots, u_t$ is a heterochromatic path in G and the lemma follows. □

Definition 7.3. Let P be a maximum length heterochromatic path in G and be given by $x = u_0,\ u_1, \ldots, u_t = y$. For each $u_i \in M_P(x)$, we define $\chi_i = C(P) \cup \{\text{color}(x, u_{i+1})\}$. Corresponding to each $u_i \in M_P(x)$, we also define $P_i = u_i, u_{i-1}, \ldots, u_0 = x, u_{i+1}, \ldots, u_t$, which is the maximum heterochromatic path given by the proof of Lemma 7.7. The path P_i has u_i and y as its endpoints, $V(P_i) = V(P)$ and $C(P_i) = \chi_i \setminus \{\text{color}(u_i, u_{i+1})\}$.

The following lemma is a direct consequence of Lemma 7.5 and Lemma 7.7.

Lemma 7.8. *Let P be a maximum length heterochromatic path in G and be given by $x = u_0,\ u_1, \ldots, u_t = y$. Then, for each $u_i \in M_P(x)$, $|N(u_i) \cap V(P)| \geq 2(\vartheta - t)$.*

Lemma 7.9. *Let P be a maximum length heterochromatic path in G and be given by $x = u_0,\ u_1, \ldots, u_t = y$. For each $u_i \in M_P(x)$, the set $\{\text{color}(u_i, u_j) \mid u_j \in N(u_i) \cap V(P)\} \setminus \chi_i$ has cardinality at least $\vartheta - t - 1$. All edges incident at u_i with colors from this set have their other end point in $V(P)$.*

Proof. Let P_i be the maximum length heterochromatic path in G with u_i as one of its end points, as given by Definition 7.3. By Lemma 7.1, all edges incident at u_i with colors from the set $\{\text{color}(u_i, v) \mid v \in N(u_i)\} \setminus C(P_i)$ have their other end point in $V(P_i)$, which is the same as $V(P)$ by the definition of P_i. This proves the second part of the lemma, because $\chi_i = C(P_i) \cup \{\text{color}(u_i, u_{i+1})\}$, by the definition of P_i.

By Lemma 7.1, we have $\{\text{color}(u_i, v) \mid v \in N(u_i)\} \setminus C(P_i) = \{\text{color}(u_i, u_j) \mid u_j \in N(u_i) \cap V(P)\} \setminus C(P_i)$. From this, we can conclude that the set $\{\text{color}(u_i, u_j) \mid u_j \in N(u_i) \cap V(P)\} \setminus C(P_i)$ has cardinality at least $\vartheta - t$, since the color degree of u_i is at least ϑ and $|C(P_i)| = t$. The first part of the lemma follows, because $\chi_i = C(P_i) \cup \{\text{color}(u_i, u_{i+1})\}$ by the definition of P_i. $\square$

Let P be a maximum length heterochromatic path in G and be given by $x = u_0,\ u_1, \ldots, u_t = y$. Since $t \geq |N(y) \cap V(P)|$, in order to find a lower bound for t, it is enough to lower bound $|N(y) \cap V(P)|$. We can do this by applying Lemma 7.2, if we can get a good lower bound for $|OLD_{\notin y} \uplus OLD_{y \to P}|$. In Lemma 7.4, we saw that for each $u_i \in M_P(x)$, $\text{color}(u_i, u_{i+1})$ belongs to $OLD_{\notin y} \uplus OLD_{y \to P}$. This observation was the crux of the proof of Theorem 7.6. Now, extending this idea, we would like to identify as many edges (u_j, u_{j+1}) of P as we can, such that $\text{color}(u_j, u_{j+1})$ belongs to $OLD_{\notin y} \uplus OLD_{y \to P}$.

Recalling Definition 7.3, we know that corresponding to each $u_i \in M_P(x)$ the path $P_i = u_i, u_{i-1}, \ldots, u_0 = x, u_{i+1}, \ldots, u_t$ is a maximum heterochromatic path in G with $u_t = y$ as one of its endpoints. Since all edges in P_i except the edge (x, u_{i+1}) were also part of P, in order to identify more edges of P whose color contributes to the set $OLD_{\notin y} \uplus OLD_{y \to P}$, a strategy would be to apply Lemma 7.4 to the path P_i for each $u_i \in M_P(x)$, taking care to discard the contribution due to the edge (x, u_{i+1}), since this edge was not in P.

Recall that $M_P(x) = \{\text{predecessor of } u_j \text{ in } P \mid u_j \in N(x) \cap V(P) \text{ and } \text{color}(x, u_j) \notin C(P)\}$. Observe that while applying Lemma 7.4 to the path P, the edges whose colors contributed to the set $OLD_{\notin y} \uplus OLD_{y \to P}$ were from $\{(u_{j-1}, u_j) \mid u_j \in N(x) \cap V(P) \text{ and } \text{color}(x, u_j) \notin C(P)\}$: here u_{j-1} was the predecessor of u_j in P. Intuitively, we can apply Lemma 7.4 to P_i for each $u_i \in M_P(x)$. and the edges of P_i we are now interested in would belong to the set $\{(pred(u_j), u_j) \mid u_j \in N(u_i) \cap V(P_i) \text{ and } \text{color}(u_i, u_j) \notin C(P_i)\} \setminus \{(x, u_{i+1})\}$, where $pred(u_j)$ is the predecessor of u_j in the path P_i from u_i to y. Since $V(P_i) = V(P)$ and $\chi_i = C(P_i) \cup \{\text{color}(u_i, u_{i+1})\}$, this means that we would be interested in the edges of P_i belonging to the set $\{(pred(u_j), u_j) \mid u_j \in N(u_i) \cap V(P) \& \, \text{color}(u_i, u_j) \notin \chi_i\}$. Note that if $u_j \in N(u_i) \cap V(P)$ such that $0 \leq j < i$, then $pred(u_j) = u_{j+1}$ and if $u_j \in N(u_i) \cap V(P)$ such that $j > i + 1$, then $pred(u_j) = u_{j-1}$. Therefore, the edges of interest belong to the set $\{(u_{j+1}, u_j) \mid u_j \in N(u_i) \cap V(P) \text{ such that} j < i \text{ and } \text{color}(u_i, u_j) \notin \chi_i\} \cap \{(u_{j-1}, u_j) \mid u_j \in N(u_i) \cap V(P) \text{ such that } j > i+1 \text{ and } \text{color}(u_i, u_j) \notin \chi_i\}$. This motivates the following definition.

Definition 7.4. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. For each $u_i \in M_P(x)$, we define $\Psi(u_i) = \{u_j \mid u_j \in N(u_i) \cap V(P) \text{ such that} j < i \text{ and } \text{color}(u_i, u_j) \notin \chi_i\} \cup \{u_j \mid u_{j+1} \in N(u_i) \cap V(P) \text{ such that } j > i \text{ and } \text{color}(u_i, u_{j+1}) \notin \chi_i\}$.

The following lemma is an integral part of the remaining proofs presented in this chapter.

Lemma 7.10. *Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. Suppose $u_i \in M_P(x)$. Then $|\Psi(u_i)| \geq \vartheta - t - 1$ and $u_i \notin \Psi(u_i)$. If $u_j \in \Psi(u_i)$, then* $\text{color}(u_j, u_{j+1})$ *belongs to* $OLD_{\notin y} \uplus OLD_{y \to P}$.

Proof. Note that $\text{color}(u_i, u_{i+1}) \in \chi_i$ and therefore, $\{\text{color}(u_i, u_j) \mid u_j \in N(u_i) \cap V(P)\} \setminus \chi_i = \{\text{color}(u_i, u_j) \mid u_j \in N(u_i) \cap V(P) \text{ such that} j < i \text{ and } \text{color}(u_i, u_j) \notin \chi_i\} \cup \{\text{color}(u_i, u_j) \mid u_j \in N(u_i) \cap V(P) \text{ such that } j > i+1 \text{ and } \text{color}(u_i, u_j) \notin \chi_i\}$. Now, the first part of this lemma follows from the definition of $\Psi(u_i)$, using Lemma 7.9. To prove the second part, assume that $u_j \in \Psi(u_i)$. By the first part of this lemma we know that $i \neq j$.

Let $P_i = u_i, u_{i-1}, \ldots, u_0 = x, u_{i+1}, \ldots, u_t = y$ be the maximum length heterochromatic path in G with u_i and y as its end points, given by Definition 7.3. Since $i \neq j$, the edge (u_j, u_{j+1}) belongs to both P_i and P. Therefore, we have $\text{color}(u_j, u_{j+1}) \in C(P_i) \cap C(P)$.

For contradiction, assume that $\text{color}(u_j, u_{j+1}) \notin OLD_{\notin y} \uplus OLD_{y \to P}$. By Definition 7.1, this implies $\text{color}(u_j, u_{j+1}) \in OLD_{y \nrightarrow P}$ and y has a neighbor $z \notin V(P)$ such that $\text{color}(y, z) = \text{color}(u_j, u_{j+1})$. Since $u_j \in \Psi(u_i)$, one of the following cases should occur by the definition of $\Psi(u_i)$:

- **Case 1:** $u_j \in N(u_i) \cap V(P)$ such that$j < i$ and color$(u_i, u_j) \notin \chi_i$.

  Since $\chi_i \supset C(P_i)$, we get color$(u_i, u_j) \notin C(P_i)$. In this case $u_j \in T_{P_i}(u_i)$ and its predecessor in P_i is the vertex u_{j+1} and therefore $u_{j+1} \in M_{P_i}(u_i)$. We apply Lemma 7.7 to the path P_i, with u_i taking the role of x, and u_{j+1} taking the role of u_i to get the following observation: u_{j+1} is an end point of a maximum heterochromatic path P' in G, such that $V(P') = V(P_i) = V(P)$ and $C(P') = C(P_i) \cup \{\text{color}(u_i, u_j)\} \setminus \{\text{color}(u_{j+1}, u_j)\}$. But, we noted that y has a neighbor $z \notin V(P)$ such that color$(y, z) =$ color(u_j, u_{j+1}), which contradicts Lemma 7.1 applied to P'.

- **Case 2:** $u_{j+1} \in N(u_i) \cap V(P)$ such that $j > i$ and color$(u_i, u_{j+1}) \notin \chi_i$.

  Since $\chi_i \supset C(P_i)$, we have color$(u_i, u_{j+1}) \notin C(P_i)$. In this case $u_{j+1} \in T_{P_i}(u_i)$ and its predecessor in P_i is the vertex u_j and therefore $u_j \in M_{P_i}(u_i)$. We apply Lemma 7.7 to the path P_i, with u_i taking the role of x and u_j taking the role of u_i, to get the following observation: u_j is an end point of a maximum heterochromatic path P'' in G, such that $V(P'') = V(P_i) = V(P)$ and $C(P'') = C(P_i) \cup \{\text{color}(u_i, u_{j+1})\} \setminus \{\text{color}(u_j, u_{j+1})\}$. But we noted that y has a neighbor $z \notin V(P)$ such that color$(y, z) =$ color(u_j, u_{j+1}), which contradicts Lemma 7.1 applied to P''.

Therefore, color$(u_j, u_{j+1}) \in OLD_{\notin y} \uplus OLD_{y \to P}$. □

Theorem 7.11. *If G is an edge colored graph which is triangle free, then the length of the maximum heterochromatic path in G is at least $\left\lfloor \frac{5\vartheta}{6} \right\rfloor$.*

Proof. First we note that Lemma 7.5 can be used to derive a weaker bound of $\left\lceil \frac{4\vartheta - 1}{5} \right\rceil$. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. By Lemma 7.5, x has at least $2(\vartheta - t)$ neighbors in $V(P)$. If G is triangle free, u_i and u_{i+1} cannot be simultaneously in $N(y)$. Therefore, $|N(y) \cap V(P)| \le \frac{t+1}{2}$. Thus $2(\vartheta - t) \le \frac{t+1}{2}$, which implies, $t \ge \left\lceil \frac{4\vartheta - 1}{5} \right\rceil$.

We derive a better bound by using Lemma 7.2, Lemma 7.4 and Lemma 7.10. By the arguments in the previous paragraph, $|N(y) \cap V(P)| \le \frac{t+1}{2}$. Since $|N(y) \cap V(P)|$ is at least $\vartheta - t + |OLD_{\notin y}| + |OLD_{y \to P}|$ by Lemma 7.2, we get, $\vartheta - t + |OLD_{\notin y}| + |OLD_{y \to P}| \le \frac{t+1}{2}$. From this, we can make the following observation.

Observation 7.1. $\vartheta + |OLD_{\notin y}| + |OLD_{y \to P}| \le \frac{3t+1}{2}$.

From this observation, it is enough to get a lower bound for $|OLD_{\notin y}| + |OLD_{y \to P}|$ in order to derive a lower bound for t. The observation below, follows from Lemma 7.4 and Lemma 7.10.

Observation 7.2. *For any $u_i \in M_P(x)$, $|OLD_{\notin y} \uplus OLD_{y \to P}| \ge |M_P(x) \cup \Psi(u_i)|$.*

Our approach is to show the existence of a $u_i \in M_P(x)$ such that $|\Psi(u_i) \cup M_P(x)|$ is sufficiently large and then use Observation 7.2 to lower bound $|OLD_{\notin y} \uplus OLD_{y \to P}|$. By Lemma 7.3, $|M_P(x)| \geq \vartheta - t$. Let $M'_P(x)$ be an arbitrary subset of $M_P(x)$ such that $|M'_P(x)| = \vartheta - t$. Let $l = \max\{k \mid u_k \in M'_P(x)\}$. We will show that, if $M'_P(x) \cap N(u_l) = \emptyset$, then taking $u_i = u_l$ suffices and if $M'_P(x) \cap N(u_l) \neq \emptyset$, taking either $u_i = u_l$ or $u_i = u_{l'}$ suffices, where $l' = \max\{k \mid u_k \in M'_P(x) \cap N(u_l)\}$.

- **Case 1:** $M'_P(x) \cap N(u_l) = \emptyset$.

  By the definition of Ψ, if $u_j \in \Psi(u_l)$ for some $j < l$, then $u_j \in N(u_l)$ and by our assumption, $u_j \notin M'_P(x)$. By the maximality of l, if $u_j \in \Psi(u_l)$ for some $j > l$, then $u_j \notin M'_P(x)$. Moreover, by Lemma 7.10, $u_l \notin \Psi(u_l)$. Therefore, $M'_P(x) \cap \Psi(u_l) = \emptyset$. This implies that $|M'_P(x) \cup \Psi(u_l)| = |M'_P(x)| + |\Psi(u_l)| \geq \vartheta - t + \vartheta - t - 1$, by Lemma 7.10. Thus, we have $|M'_P(x) \cup \Psi(u_l)| \geq 2(\vartheta - t) - 1$ and since $M_P(x) \supseteq M'_P(x)$, we get $|M_P(x) \cup \Psi(u_i)| \geq 2(\vartheta - t) - 1$. Therefore, by Observation 7.2, $|OLD_{\notin y} \uplus OLD_{y \to P}| \geq 2(\vartheta - t) - 1$. By Observation 7.1, we get $\frac{3t+1}{2} \geq \vartheta + 2(\vartheta - t) - 1$ and therefore, $t \geq \left\lceil \frac{6\vartheta - 3}{7} \right\rceil \geq \left\lfloor \frac{6\vartheta}{7} \right\rfloor$.

- **Case 2:** $M'_P(x) \cap N(u_l) \neq \emptyset$. Let $l' = \max\{k \mid u_k \in M'_P(x) \cap N(u_l)\}$. Clearly, $l' < l$ and u_l and $u_{l'}$ are both in $M'_P(x)$ and they are adjacent to each other.

  Since G is triangle free, u_l and $u_{l'}$ have no common neighbors. From this, it follows that there is no $u_j \in \Psi(u_l) \cap \Psi(u_{l'})$, with $j < l'$ and there is no $u_j \in \Psi(u_l) \cap \Psi(u_{l'})$, with $j > l$. Moreover, $u_l \notin \Psi(u_l)$ and $u_{l'} \notin \Psi(u_{l'})$, by Lemma 7.10. If there is a $u_j \in \Psi(u_l) \cap \Psi(u_{l'})$, with $l' < j < l$, then $u_j \in N(u_l)$ by the definition of $\Psi(u_l)$ and therefore by the maximality of l', it follows that $u_j \notin M'_P(x)$. Therefore, $\Psi(u_l) \cap \Psi(u_{l'}) \cap M'_P(x) = \emptyset$.

  Moreover, since u_l, $u_{l'}$ are neighbors of each other, $u_{l+1} \notin N(u_{l'})$ and since $l > l'$ it follows that $u_l \notin \psi(u_{l'})$. We also have $u_l \notin \Psi(u_l)$, by Lemma 7.10. From these observations,
  $|\Psi(u_l) \setminus M'_P(x)| + |\Psi(u_{l'}) \setminus M'_P(x)| = |\Psi(u_l)| + |\Psi(u_{l'})| - (|\Psi(u_l) \cap M'_P(x)| + |\Psi(u_{l'}) \cap M'_P(x)|)$
  $\geq |\Psi(u_l)| + |\Psi(u_{l'})| - (|M'_P(x)| - 1)$, since $\Psi(u_l) \cap \Psi(u_{l'}) \cap M'_P(x) = \emptyset$ and $u_l \in M'_P(x) \setminus (\Psi(u_l) \cup \Psi(u_{l'}))$.
  $\geq (\vartheta - t - 1) + (\vartheta - t - 1) - (\vartheta - t - 1)$, by Lemma 7.10.
  $\geq \vartheta - t - 1$.

  This implies that either $|\Psi(u_{l'}) \setminus M'_P(x)| \geq \left\lceil \frac{\vartheta - t - 1}{2} \right\rceil$ or $|\Psi(u_l) \setminus M'_P(x)| \geq \left\lceil \frac{\vartheta - t - 1}{2} \right\rceil$. Since $|M'_P(x)| = \vartheta - t$, we get either $|M'_P(x) \cup \Psi(u_l)| \geq \left\lceil \frac{3(\vartheta - t) - 1}{2} \right\rceil$ or $|M'_P(x) \cup \Psi(u_{l'})| \geq \left\lceil \frac{3(\vartheta - t) - 1}{2} \right\rceil$. Since $M_P(x) \supseteq M'_P(x)$, this implies that we have either $|M_P(x) \cup \Psi(u_l)| \geq \left\lceil \frac{3(\vartheta - t) - 1}{2} \right\rceil$ or $|M_P(x) \cup \Psi(u_{l'})| \geq$

$\left\lceil \frac{3(\vartheta-t)-1}{2} \right\rceil$.

This gives $|OLD_{\notin y} \uplus OLD_{y\to P}| \geq \left\lceil \frac{3(\vartheta-t)-1}{2} \right\rceil$ by Observation 7.2. By Observation 7.1, we get $\frac{3t+1}{2} \geq \vartheta + \left\lceil \frac{3(\vartheta-t)-1}{2} \right\rceil$ and therefore, $t \geq \left\lceil \frac{5\vartheta-2}{6} \right\rceil \leq \left\lfloor \frac{5\vartheta}{6} \right\rfloor$.

□

Theorem 7.12. *If G is edge colored and is bipartite, then $\lambda(G) \geq \left\lceil \frac{6\vartheta-3}{7} \right\rceil$.*

Proof. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. By Lemma 7.3, $|M_P(x)| \geq \vartheta - t$. Let $M'_P(x)$ be an arbitrary subset of $M_P(x)$ such that $|M'_P(x)| = \vartheta - t$. Let $l = \max\{k \mid u_k \in M'_P(x)\}$.

If two vertices in $M'_P(x)$ are adjacent, it will create either a three cycle or a five cycle in G, which is not possible, since G is bipartite. Therefore, $M'_P(x) \cap N(u_l) = \emptyset$, where $l = \max\{k \mid u_k \in M'_P(x)\}$ and from Case 1 of the proof of Theorem 7.11, the statement follows. □

Now we turn our attention to the case of graphs without cycles of length 4.

Theorem 7.13. *Let G be an edge colored graph without cycles of length 4. Then $\lambda(G) \geq \vartheta - \sqrt{\frac{2\vartheta}{3}}$.*

Proof. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. To prove the theorem, we will first show that $|V(P)| \geq \frac{(\vartheta-t)+3(\vartheta-t)^2}{2}$. By Lemma 7.3 and Lemma 7.8, we know that $|M_P(x)| \geq \vartheta - t$ and for each $v \in M_P(x)$, $|N(v) \cap V(P)| \geq 2(\vartheta - t)$. Suppose $M'_P(x)$ is a subset of $M_P(x)$ where $M'_P(x) = \{v_1, v_2, \ldots, v_k\}$, with $k = \vartheta - t$. Since G has no four cycles, for $v, w \in M_P(x)$, $|N(v) \cap N(w)| \leq 1$.

Since $V(P) \supseteq \bigcup_{v \in M'_P(x)} N(v) \cap V(P) = V(P) \cap [N(v_1) \uplus (N(v_2) \setminus N(v_1)) \uplus \cdots \uplus (N(v_k) \setminus (N(v_1) \cup N(v_2) \cup \cdots N(v_{k-1})))]$, using the observation from the above paragraph we have $|V(P)| \geq 2(\vartheta-t) + 2(\vartheta-t) - 1 + \cdots + 2(\vartheta-t) - (k-1)$. Since $k = \vartheta - t$, this gives $|V(P)| \geq \frac{(\vartheta-t)+3(\vartheta-t)^2}{2}$. This implies $t + 1 \geq \frac{(\vartheta-t)+3(\vartheta-t)^2}{2}$. It is easy to verify that if $t < \vartheta - \sqrt{\frac{2\vartheta}{3}}$, this leads to a contradiction. Therefore, $t \geq \vartheta - \sqrt{\frac{2\vartheta}{3}}$. □

If the girth of G is at least 7, we can slightly improve the bound given by Theorem 7.13.

Theorem 7.14. *Let G be an edge colored graph of girth at least 7. Then $\lambda(G) > \vartheta - \sqrt{\frac{\vartheta}{2}}$.*

Proof. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. To prove the theorem, we will first show that $|V(P)| \geq 1 + 2(\vartheta - t) + 2(\vartheta - t)^2$. Since G has girth is at least 7, we can make the following observations:

- Each $u_i \in M_P(x)$ has exactly one neighbor in $N(x)$, which is u_{i+1}. Therefore, $|(N(u_i) \setminus N(x)) \cap V(P)| \geq 2(\vartheta - t) - 1$, by Lemma 7.8.
- $N(x) \cap M_P(x) = \emptyset$.
- No two vertices in $M_P(x)$ can be adjacent.
- $N(M_P(x)) \cap \{x\} \cup M_P(x) = \emptyset$. This follows from the second and third observations above.
- For $u_i, u_j \in M_P(x)$, with $i \neq j$, $N(u_i) \cap N(u_j) = \emptyset$. This gives, $|(N(M_P(x)) \setminus N(x)) \cap V(P)| \geq (\vartheta - t)[2(\vartheta - t) - 1]$, by Lemma 7.3 and the first observation above.

From the facts listed above, $V(P) \supseteq \{x\} \uplus [N(x) \cap V(P)] \uplus M_P(x) \uplus [(N(M_P(x)) \setminus N(x)) \cap V(P)]$. Therefore, by Lemma 7.5 and Lemma 7.3 and the last observation above, we get $|V(P)| \geq 1 + 2(\vartheta - t) + (\vartheta - t) + (\vartheta - t)[2(\vartheta - t) - 1]$. On simplification this gives, $t + 1 = |V(P)| \geq 1 + 2(\vartheta - t) + 2(\vartheta - t)^2$. It is easy to verify that if $t \leq \vartheta - \sqrt{\frac{\vartheta}{2}}$, this leads to a contradiction. Therefore, $t > \vartheta - \sqrt{\frac{\vartheta}{2}}$. □

We will be using the following result by Alon et al. [9], in order to derive lower bounds for $\lambda(G)$ in terms of the girth of G.

Lemma 7.15 (Alon et al.[9]). *Let G be a graph of average degree d and girth g. Then, G has at least $4\left(\left\lfloor \frac{d}{2} \right\rfloor\right)^{\frac{g-2}{2}}$ vertices.*

Now, we will obtain a lower bound for the the average degree of the induced subgraph of G on the vertex set $V(P)$ and obtain a lower bound for $|V(P)|$ using Lemma 7.15.

Lemma 7.16. *Let G be an edge colored graph. Let P be a maximum length heterochromatic path in G and P be of length t. If $t \leq \vartheta - 1$, then the average degree of the induced subgraph of G on the vertex set $V(P)$ is at least $\frac{2[(\vartheta - t + 2)(\vartheta - t - 1) + (t+1)]}{t+1}$.*

Proof. From Lemma 7.3 and Lemma 7.8, it follows that the total degree of vertices in $M_P(x)$ in the induced graph on $V(P)$ is at least $2(\vartheta - t)^2$. Also, the degrees of x and y in the induced subgraph on $V(P)$ are at least $2(\vartheta - t)$ by Lemma 7.5, which is at least 2 because $t \leq \vartheta - 1$ by our assumption. Since $x, y \notin M_P(x)$, and the vertices in $V(P) \setminus (M_P(x) \cup \{x, y\})$ have degree at least two in the induced subgraph, the total degree of vertices in the induced subgraph on $V(P)$ is at least $4(\vartheta-t)+|M_P(x)|2(\vartheta-t)+(t+1-|M_P(x)|-2)2 = 4(\vartheta-t-1)+2|M_P(x)|(\vartheta-t-1)+2(t+1)$. By lemma 7.3, this is at least $4(\vartheta-t-1)+2(\vartheta-t)(\vartheta-t-1)+2(t+1)) = 2[(\vartheta-t-1)(\vartheta-t+2)+(t+1)]$. From this, the lemma follows. □

Theorem 7.17. *Let G be an edge colored graph of girth at least g. Then the maximum length heterochromatic path in G has length at least $(\vartheta-1)-(\sqrt{\vartheta})(\frac{\vartheta}{4})^{\frac{1}{g-2}}$.*

Proof. Let P be a maximum length heterochromatic path in G and t be the length of P. If $\vartheta-t\leq 1$, the lemma follows directly. Therefore, noting that $\vartheta-t$ is an integer, we assume $\vartheta-t\geq 2$.

Let G' be the induced subgraph of G on the vertex set $V(P)$. By Lemma 7.16, the average degree d of G' is at least $\frac{2[(\vartheta-t+2)(\vartheta-t-1)+(t+1)]}{t+1}$. Then, $\left\lfloor\frac{d}{2}\right\rfloor \geq \frac{(\vartheta-t+2)(\vartheta-t-1)}{t+1}$. Since $|V(G')|=t+1$, by Lemma 7.15, we get

$t+1\geq 4\left[\frac{(\vartheta-t+2)(\vartheta-t-1)}{t+1}\right]^{\frac{g-2}{2}}$
$\Rightarrow t+1\geq 4\left[\frac{(\vartheta-t-1)^2}{t+1}\right]^{\frac{g-2}{2}}$
$\Rightarrow (t+1)^{\frac{g}{g-2}}\geq 4^{\frac{2}{g-2}}[\vartheta-(t+1)]^2$
$\Rightarrow (t+1)^{\frac{g}{2(g-2)}}\geq 4^{\frac{1}{g-2}}[\vartheta-(t+1)]$
$\Rightarrow (\frac{1}{4})^{\frac{1}{g-2}}(t+1)^{\frac{g}{2(g-2)}}\geq [\vartheta-(t+1)]$
$\Rightarrow \vartheta\leq (t+1)+(\frac{1}{4})^{\frac{1}{g-2}}(t+1)^{\frac{g}{2(g-2)}}$
If $t+1<\vartheta-(\frac{1}{4})^{\frac{1}{g-2}}\vartheta^{\frac{g}{2(g-2)}}$, the above inequality will not be satisfied.
Therefore, $t\geq(\vartheta-1)-(\frac{1}{4})^{\frac{1}{g-2}}\vartheta^{\frac{g}{2(g-2)}}=(\vartheta-1)-(\sqrt{\vartheta})(\frac{\vartheta}{4})^{\frac{1}{g-2}}$. □

Corollary 7.18.
- *If the girth of G is at least 5, G has a maximum heterochromatic path of length at least $(\vartheta-1)-0.63\vartheta^{\frac{5}{6}}$.*
- *If the girth of G is at least 6, G has a maximum heterochromatic path of length at least $(\vartheta-1)-0.71\vartheta^{\frac{3}{4}}$.*

Remark 7.1. *The lower bound given by Theorem 7.17 improves as the girth increases, but it is clear that this bound cannot grow beyond $(\vartheta-1)-\sqrt{\vartheta}$. When the girth is at least 7, the bound given by Theorem 7.14 is better than the bound given by Theorem 7.17. In the remaining parts of this section, we will show how to extend the ideas used in the proof of Lemma 7.16, to obtain a lower bound for $|V(P)|$ much better than the bounds given by Theorem 7.14 and Theorem 7.17, in the case of graphs of larger girth.*

The claim below will be useful for us in deriving a better lower bound for $|V(P)|$.

Claim 7.18.1. *Suppose G has girth g or more, where $g \geq 5$ and let $k = \left\lfloor \frac{g-1}{4} \right\rfloor$. Let P be a maximum length heterochromatic path in G. Let x and y be the endpoint of P and t be the length of P. We can define a sequence of subsets of $V(P)$ given by $M_0, M_1, \ldots M_k, T_1, T_2, \ldots, T_k$ such that the following properties are satisfied for each $0 \leq i \leq k$:*

1. *If $i > 0$, there exists a mapping $parent : M_i \mapsto T_i$ satisfying $(u, parent(u)) \in E(G)$ for each $u \in M_i$ and there exists a mapping $parent : T_i \mapsto M_{i-1}$ satisfying $(v, parent(v)) \in E(G)$ for each $v \in T_i$.*

2. *There exists a path of length $2i$ from x to u for each $u \in M_i$ and there exists a path of length $2i - 1$ from x to v for each $v \in T_i$.*

3. *The sets M_0, T_1, $M_1 \ldots, T_i, M_i$ are pairwise disjoint.*

4. *$|M_0| = 1$, $|M_1| \geq (\vartheta - t)$ and $|T_i| \geq |M_{i-1}|(\vartheta - t - 1)$. If $i \geq 2$, $|M_i| \geq |M_{i-1}|(\vartheta - t - 1)$.*

5. *For every $u \in M_i$ there exist a maximum heterochromatic path in G with u and y as its endpoints and its vertex set the same as $V(P)$. This path will be denoted by $path(u)$.*

6. *For every $u \in M_i$, $|N(u) \cap V(P)| \geq 2(\vartheta - t)$.*

Proof. We inductively construct the sequence of sets $M_0, M_1, \ldots M_k$ and $T_1, T_2, \ldots, T_k$. We start with the definition of M_0, M_1 and T_1.

Define $M_0 = \{x\}$ and $T_1 = \{u \in N(x) \cap V(P) \mid \mathrm{color}(x, u) \notin C(P)\}$. For each $u \in T_1(x)$, we define $parent(u) = x$. Define $M_1 = \{v \mid v$ is the predecessor of a vertex $u \in T_1(x)$ in the path P from x to $y\}$. If $v \in M_1$ is the predecessor of $u \in T_1(x)$ in the path P from x to y, we define $parent(v) = u$.

Recall the definition of $T_P(x)$ and $M_P(x)$ and note that $T_1 = T_P(x)$ and $M_1 = M_P(x)$. From this observation and the girth condition, it is easy to verify that at this stage, properties 1 to 3 in Claim 7.18.1 are satisfied by M_0, M_1 and T_1 and by applying Lemma 7.5, Lemma 7.7 and Lemma 7.8, property 4, property 5 and property 6 in Claim 7.18.1 can also be verified. Assume that after stage $i - 1$, the sets $M_0, M_1, \ldots M_{i-1}$ and $T_1, T_2, \ldots, T_{i-1}$ are already defined and they satisfy all the properties in Claim 7.18.1. Now we describe how to construct T_i and M_i.

Note that by the induction hypothesis, for each $u \in M_{i-1}$, $path(u)$ is a maximum heterochromatic path in G with u and y as its endpoints and its vertex set the same as $V(P)$. Therefore, by applying Definition 7.2 to $path(u)$ we have $T_{path(u)}(u) = \{v \in N(u) \cap V(P) \mid \mathrm{color}(u, v) \notin C(path(u))\}$. Clearly, for each $v \in T_{path(u)}(u)$, $(v, u) \in E(G)$. Note that if $v \in T_{path(u)}(u) \setminus \{parent(u)\}$, then $v \notin M_0 \uplus M_1 \uplus \cdots \uplus M_{i-1} \uplus T_1 \uplus T_2 \uplus \cdots \uplus T_{i-1}$ by the girth condition. Hence for each $v \in T_{path(u)}(u) \setminus \{parent(u)\}$ there is a path of length $2i - 1$ from x

to v in G, since by the induction hypothesis there is a path of length $2(i-1)$ between x and $u \in M_{i-1}$.

Consider $u \in M_{i-1}$. Since each vertex in $M_0 \uplus M_1 \uplus \cdots \uplus M_{i-1} \uplus T_1 \uplus T_2 \uplus \cdots \uplus T_{i-1}$ has a path of length at most $2(i-1)$ to x by our inductive assumption, for each $v \in T_{path(u)}(u) \setminus \{parent(u)\}$ the only neighbor of v in the set in $M_0 \uplus M_1 \uplus \cdots \uplus M_{i-1} \uplus T_1 \uplus T_2 \uplus \cdots \uplus T_{i-1}$ should be the vertex u, by the girth condition. We claim that *if* $v \in T_{path(u)}(u) \setminus \{parent(u)\}$*, then* $v \notin T_{path(w)}(w) \setminus \{parent(w)\}$*, for any* $w \in M_{i-1}$ *where* $w \neq u$*.* Otherwise, v will be adjacent to two different vertices u and w in M_{i-1}, a contradiction. Therefore if $u, w \in M_{i-1}$ and $u \neq w$, then the sets $T_{path(u)}(u) \setminus \{parent(u)\}$ and $T_{path(w)}(w) \setminus \{parent(w)\}$ are disjoint. We define

$$T_i = \biguplus_{u \in M_{i-1}} \Big[T_{path(u)}(u) \setminus \{parent(u)\}\Big].$$

Now we define the mapping $parent : T_i \mapsto M_{i-1}$ as follows:

For each $u \in M_{i-1}$,

define $parent(v) = u$, for each $v \in T_{path(u)}(u) \setminus \{parent(u)\}$

Note that the mapping above is well defined by the definition of T_i. From the above description, it is also clear that for each $v \in T_i$ there exists a path of length $2i-1$ from x to v and $(v, parent(v)) \in E(G)$. Now, the girth condition ensures that T_i is disjoint from $M_0 \uplus M_1 \uplus \cdots \uplus M_{i-1} \uplus T_1 \uplus T_2 \uplus \cdots \uplus T_{i-1}$. For each $u \in M_{i-1}$, by Lemma 7.3 applied to $path(u)$, $|T_{path(u)}(u)| \geq \vartheta - t$ and therefore, $|T_{path(u)}(u) \setminus \{parent(u)\}| \geq \vartheta - t - 1$. This implies that $|T_i| \geq |M_{i-1}|(\vartheta - t - 1)$. Thus, the sets $M_0, M_1, \ldots, M_{i-1}, T_1, T_2, \ldots, T_{i-1}, T_i$ satisfy all the required properties in Claim 7.18.1. Now we define M_i.

Define $M_i = \bigcup_{v \in T_i} \{\text{predecessor of } v \text{ in } path(parent(v))\}$

If $u' \in M_i$ is the predecessor of $v \in T_i$ in $path(parent(v))$, clearly $(u', v) \in E(G)$. We claim that *for each* $u' \in M_i$*, there exist a unique* $v \in T_i$ *such that* u' *is the predecessor of* v *in* $path(parent(v))$. If this claim was not true, v will have two distinct neighbors in the set T_i, which contradicts the girth condition. We use the above claim for the following two purposes.

1. We note that $|M_i| = |T_i|$.

2. We define the mapping $parent : M_i \mapsto T_i$ as follows:

   If $u' \in M_i$ is the predecessor of $v \in T_i$ in $path(parent(v))$, then define $parent(u') = v$.

   This mapping is well defined, because of the claim we proved in the previous paragraph.

By this definition, for each $u' \in M_i$ we have $(u', parent(u')) \in E(G)$. We can see that there exists a path of length $2i$ from x to u' for each $u' \in M_i$, because there is already a path of length $2i-1$ from x to $parent(u')$ by the induction hypothesis and $u' \notin M_0 \uplus M_1 \uplus \cdots \uplus M_{i-1} \uplus T_1 \uplus T_2 \uplus \cdots \uplus T_{i-1} \uplus T_i$, by the girth condition. The girth condition also ensures that M_i is disjoint from $M_0 \uplus M_1 \uplus \cdots \uplus M_{i-1} \uplus T_1 \uplus T_2 \uplus \cdots \uplus T_i$. We also get $|M_i| \geq |M_{i-1}|(\vartheta - t - 1)$, because we have seen that $|M_i| = |T_i|$ and $|T_i| \geq |M_{i-1}|(\vartheta - t - 1)$. Thus, M_i satisfies properties 1 to 4 in Claim 7.18.1. It remains to show that M_i will satisfy properties 5 and 6 also.

Consider $u' \in M_i$ and let $v = parent(u')$ where $v \in T_i$. Suppose $u \in M_{i-1}$ is the $parent(v)$. By our definitions, $v \in T_{path(u)}(u)$ and u' is the predecessor of v in $path(u)$. This implies $u' \in M_{path(u)}(u)$ and therefore, we can apply Lemma 7.3 to $path(u)$.

> For each $u' \in M_i$ we define $path(u')$ to be the maximum heterochromatic path with endpoints u' and y obtained as per Definition 7.3 by applying Lemma 7.7 to $path(u)$, where $u = parent(parent(u'))$.

For each $u' \in M_i$, we have $V(path(u')) = V(P)$ by Definition 7.3 and we get $|N(u') \cap V(P)| \geq 2(\vartheta - t)$, by applying Lemma 7.5 to $path(u')$. This shows that M_i satisfies properties 5 and 6 in Claim 7.18.1 as well.

Thus, the sets $M_0, M_1, \ldots M_i$ and $T_1, T_2, \ldots, T_i$ satisfy all the required properties in Claim 7.18.1 and the statement of the claim follows. □

By the above claim, $|V(P)| \geq |M_0| + |T_1| + |M_1| + \cdots + |T_k| + |M_k|$. From this we can derive a lower bound for the length of P, using property 4 of Claim 7.18.1. Alternatively, the observations above can be used to derive a lower bound for the average degree of the induced subgraph of G on $V(P)$, which could then be used in Lemma 7.15 to derive a lower bound for the length of P. We use the latter approach, as it seems to yield a better lower bound.

Lemma 7.19. *Let G be an edge colored graph of girth g or more, $g \geq 5$. Let P be a maximum length heterochromatic path in G and P be of length t. If $t \leq \vartheta - 1$, the average degree of the induced subgraph of G on the vertex set $V(P)$ is at least $\frac{2[(\vartheta-t-1)^{\lceil \frac{g}{4} \rceil}+(t+1)]}{t+1}$.*

Proof. Assume that G has girth g or more, where $g \geq 5$ and P is given by $x = u_0, u_1, \ldots, u_t = y$. Also assume that $t \leq \vartheta - 1$. Let $d(P)$ denote the average degree of the induced subgraph of G on the vertex set $V(P)$ and $\Gamma(P)$ denote the total degree of the induced subgraph of G on the vertex set $V(P)$. When $5 \leq g \leq 8$, by Lemma 7.16, $d(P) \geq \frac{2[(\vartheta-t+2)(\vartheta-t-1)+(t+1)]}{t+1} \geq \frac{2[(\vartheta-t-1)^{\lceil \frac{g}{4} \rceil}+(t+1)]}{t+1}$. Therefore, we can assume that $g \geq 9$.

Let $k = \left\lfloor \frac{g-1}{4} \right\rfloor$ and $M_0, M_1, \ldots M_k, T_1, T_2, \ldots, T_k$ be as given by Claim 7.18.1. Let $V_1 = M_0 \uplus M_1 \uplus \cdots \uplus M_k$ and $V_2 = V(P) \backslash V_1$. By Claim 7.18.1, $V_1 \subseteq V(P)$

and for each $v \in V_1$, $|N(v) \cap V(P)| \geq 2(\vartheta - t)$ and $|M_k| \geq (\vartheta - t)(\vartheta - t - 1)^{k-1}$. By Lemma 7.5, $|N(x) \cap V(P)| \geq 2(\vartheta - t)$ and $|N(y) \cap V(P)| \geq 2(\vartheta - t)$. This implies $|N(x) \cap V(P)| \geq 2$ and $|N(y) \cap V(P)| \geq 2$, since $t \leq \vartheta - 1$ by our assumption. Since all nodes in $V(P)$ other than x, y are internal nodes of the path P, it is clear that for each $v \in V_2$, $|N(v) \cap V(P)| \geq 2$.

Since $V(P) = V_1 \uplus V_2$, from the above observations we get,

$$\begin{aligned}
&\Gamma(P) = \textstyle\sum_{u \in V_1} |N(u) \cap V(P)| + \sum_{u \in V_2} |N(u) \cap V(P)| \\
&\geq |V_1| 2(\vartheta - t) + 2|V_2| \\
&= 2|V_1|(\vartheta - t) + 2(t + 1 - |V_1|) \\
&= 2|V_1|(\vartheta - t - 1) + 2(t + 1) \\
&\geq 2|M_k|(\vartheta - t - 1) + 2(t + 1) \\
&\geq 2(\vartheta - t)(\vartheta - t - 1)^{k-1}(\vartheta - t - 1) + 2(t + 1) \\
&\geq 2[(\vartheta - t - 1)^{k+1} + (t + 1)] \\
&= 2[(\vartheta - t - 1)^{\lceil \frac{g}{4} \rceil} + (t + 1)], \\
&\text{since } k = \left\lfloor \tfrac{g-1}{4} \right\rfloor, \text{ which implies } k + 1 = \left\lceil \tfrac{g}{4} \right\rceil.
\end{aligned}$$

This implies that if G has girth g or more, where $g \geq 5$, the average degree of the induced subgraph of G on the vertex set $V(P)$ satisfies $d(P) \geq \frac{2[(\vartheta - t - 1)^{\lceil \frac{g}{4} \rceil} + (t+1)]}{t+1}$. □

Theorem 7.20. *Let G be an edge colored graph of girth $g \geq 5$. Then the maximum length heterochromatic path in G has length at least* $(\vartheta - 1) - \vartheta^{\frac{g}{\lceil \frac{g}{4} \rceil (g-2)}}$

Proof. Let t be the length of a maximum length heterochromatic path P in G. If $\vartheta - t \leq 1$, the lemma follows directly. Therefore, since $\vartheta - t$ is an integer, we assume that $\vartheta - t \geq 2$. By Lemma 7.19, the average degree d of the induced subgraph of G on $V(P)$ is at least $\frac{2[(\vartheta - t - 1)^{\lceil \frac{g}{4} \rceil} + (t+1)]}{t+1}$. Using Lemma 7.15, we get, $t + 1 \geq 4 \left(\left\lfloor \frac{d}{2} \right\rfloor \right)^{\frac{g-2}{2}} \geq 4(\vartheta - t - 1)^{\lceil \frac{g}{4} \rceil \frac{g-2}{2}}$. To satisfy this, we should have $t + 1 \geq \vartheta - \vartheta^{\frac{g}{\lceil \frac{g}{4} \rceil (g-2)}}$ or $t \geq (\vartheta - 1) - \vartheta^{\frac{g}{\lceil \frac{g}{4} \rceil (g-2)}}$. □

Corollary 7.21.

- *If the girth of G is at least* 9*, then* $\lambda(G) \geq (\vartheta - 1) - \vartheta^{\frac{9}{21}}$.
- *If the girth of G is at least* 10*, then* $\lambda(G) \geq (\vartheta - 1) - \vartheta^{\frac{10}{24}}$.
- *If the girth of G is at least* 13*, then* $\lambda(G) \geq (\vartheta - 1) - \vartheta^{\frac{13}{44}}$.
- *If the girth of G is at least* $4 \log_2(\vartheta) + 2$*, then* $\lambda(G) \geq \vartheta - 2$.

Remark 7.2. *When the girth of G is smaller than* 9*, the lower bounds given by Theorem 7.14 and Theorem 7.17 are better than the lower bound given by Theorem 7.20. However, as the girth increases, the bound given by Theorem 7.20 outperforms the bounds given by Theorem 7.14 and Theorem 7.17.*

7.4 Maximum heterochromatic paths in heterochromatic triangle-free graphs

If a graph G is edge colored in such a way that each triangle in G is colored with at most two colors, we say that the coloring of G is a Gallai Coloring. If the edges of a triangle are colored with three distinct colors, we call it a Gallai triangle or a heterochromatic triangle. Chen and Li [34] showed that if an edge colored graph G has no heterochromatic triangles, then G has a heterochromatic path of length at least $\left\lceil \frac{3\vartheta(G)}{4} \right\rceil$. In this section, we give a proof showing that this bound can be improved to $\left\lfloor \frac{13\vartheta(G)}{17} \right\rfloor$.

Let G be a Gallai colored graph and let P be a maximum length heterochromatic path in G. Let P be of length t and be given by $x = u_0$, $u_1, \ldots, u_t = y$ and $OLD_{\notin y}$, $OLD_{y \to P}$, $OLD_{y \nrightarrow P}$, $NEW_{y \to P}$ be defined as in Section 7.2. Recall that $T_P(x) = \{u_i \in N(x) \mid \text{color}(x, u_i) \notin C(P)\}$ and $M_P(x) = \{u_i \mid u_i \text{ is the predecessor of a vertex in } T_P(x) \text{ in the path } P \text{ from } x \text{ to } y\}$.

Let T'_x be a subset of $T_P(x)$ of cardinality $\vartheta - t$ chosen in such a way that if $u_i, u_j \in T'_x$, then $\text{color}(x, u_i) \neq \text{color}(x, u_j)$. We can define T'_x this way because there are at least $\vartheta - t$ distinct new colors incident at x and by Lemma 7.1, all such edges should have their other end point in $V(P)$.

We define $M'_x = \{u_i \mid u_i \text{ is the predecessor of a vertex in } T'_x \text{ in the path } P \text{ from } x \text{ to } y\}$. By this definition, $M'_x \subseteq M_P(x)$ and $|M'_x| = \vartheta - t$. Corresponding to each $u_i \in M'_x$, let $\chi_i = C(P) \cup \{\text{color}(x, u_{i+1}\}$, as in Section 7.3.

Lemma 7.22. *Let G be a Gallai colored graph. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. Let M'_x be as defined above and $u_i \in M'_x$. Then* $\text{color}(u_i, u_{i+1})$ *belongs to $OLD_{\notin y} \uplus OLD_{y \to P}$. Moreover, if $u_i, u_j \in M'_x$ with $i \neq j$, then $|i - j| > 1$.*

Proof. The first part of this lemma follows from Lemma 7.4, because $M'_x \subseteq M_P(x)$.

Now we show that the second part of the lemma follows from the Gallai coloring property of G. For contradiction, assume that two consecutive vertices in P, say, $u_i, u_{i+1} \in M'_x$. This implies that u_{i+1}, u_{i+2} both belong to T'_x. By the definition of T'_x, we have $\text{color}(x, u_{i+1}) \notin C(P)$, $\text{color}(x, u_{i+2}) \notin C(P)$ and $\text{color}(x, u_{i+1}) \neq \text{color}(x, u_{i+2})$. This implies that the vertices x, u_{i+1}, u_{i+2} induce a Gallai triangle in G, a contradiction. □

Lemma 7.23. *Let G be a Gallai colored graph. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. Let M'_x be as defined earlier. Then, there exists a vertex $u_m \in M'_x$ such that we can choose $\left\lceil \frac{(\vartheta - t - 2)}{4} \right\rceil$ distinctly colored edges from u_m to $V(P) \setminus M'_x$ such that: (i) no chosen edge from u_m is to $u_{m'+1}$ where $u_{m'} \in M'_x$ with $m' > m$. (ii) no chosen edge has its color from the set χ_m.*

Proof. We will describe the construction of a subgraph H'' of G and show the existence of a $u_m \in V(H'')$ with the desired properties. To construct H'', we do the following.

Step 1: In this step, we construct a subgraph H of G. We will be defining the subgraph H by specifying its edge set $E(H)$. The vertex set of H will be implicitly taken as the union of the endpoints of edges in $E(H)$. We will be carefully selecting the edges to add to H so that no edge of H has its color from the set $C(P)$. To select the edges of H, we do the following.

Consider the vertices in M'_x in the order they appear in the heterochromatic path P from x to y. While considering $u_i \in M'_x$, do the following:

1. If no edge incident at u_i has been included in H so far, for each color $c \notin \chi_i$ which is present at u_i in G, do the following: Choose exactly one edge of color c incident at u_i and include in H, giving preference to an edge from u_i to another vertex in M'_x, if one exists.

2. If some edge incident at u_i has been already included in H, for each color $c \notin \chi_i$ which is present at u_i in G do the following: If no edge incident at u_i of color c has been added into H so far, choose exactly one edge of color c incident at u_i in G and include it in H, giving preference to an edge from u_i to another vertex in M'_x, if one exists.

From the above procedure, it is clear that all the edges added have at least one end point in $M'_x \subseteq V(P)$. Since none of the edges added while considering $u_i \in M'_x$ have color from the set χ_i, by Lemma 7.9, all the selected edges have their both end points in $V(P)$. Therefore, $V(H) \subseteq V(P)$. From Lemma 7.9, it is also clear that for each $u_i \in M'_x$, the color degree of u_i in H is at least $\vartheta - t - 1$.

In the next step, we will *clean up* H, by deleting some of its edges. However, before proceeding to the next step, let us get a lower bound for the total number of edges in H. Clearly, $M'_x \subseteq V(H)$ and all edges in H have one of its end points in M'_x. For each vertex $u_i \in M'_x$, let b_i represent the number of vertices in M'_x other than u_i, which are non-adjacent to u_i in H. Thus, for each $u_i \in M'_x$, there are $|M'_x| - 1 - b_i = \vartheta - t - 1 - b_i$ edges in H from u_i to other vertices in M'_x. This implies that the number of edges of H in the induced subgraph on M'_x is $s_1 = \frac{1}{2}\sum_{u_i \in M'_x}(\vartheta - t - 1 - b_i)$.

We claim that if a color c is repeated at $u_i \in M'_x$, all edges of color c incident at u_i have their other end point in M'_x itself. To see this, assume that $(u_i, u_j) \in E(H)$ with $u_j \notin M'_x$ and $\text{color}(u_i, u_j) = c$. Since $u_j \notin M'_x$, this edge should have been added in Step 1 while considering u_i. By rule 2 of the procedure mentioned in Step 1, this implies that color c was not present at u_i before considering u_i and there are no edges incident at u_i of color c with its other endpoint also in M'_x. Rule 2 also ensures that while considering u_i the only edge of color c incident at u_i added to $E(H)$ is (u_i, u_j). Since there are no edges incident at u_i of color c with its other endpoint also in M'_x, at the end of Step 1 the only edge of color c incident at u_i in H would be (u_i, u_j), proving our claim. Similarly, if color(x, u_{i+1}) occurs at $u_i \in M'_x$, all edges of this color incident at u_i have their other end point in M'_x itself, because no edge of color color(x, u_{i+1}) was selected while considering u_i, since color$(x, u_{i+1}) \in \chi_i$. To make our later arguments easier, imagine that for each color present at u_i in H, the vertex $u_i \in M'_x$ puts a red-flag on all except one edge of that color incident at u_i in H. Let r_i represent the number of red-flags at u_i. From the definitions of b_i and r_i, for each $u_i \in M'_x$, the number of distinct colored edges occurring at u_i in H, with their other end point in M'_x is $|M'_x| - 1 - b_i - r_i = \vartheta - t - 1 - b_i - r_i$.

Since the color degree of u_i in H is at least $\vartheta - t - 1$, this implies that there are at least $b_i + r_i$ edges from u_i to $V(H) \setminus M'_x$ in H. Therefore, the total number of edges in H with exactly one end point in M'_x and the other point in $V(H) \setminus M'_x$ is at least $s_2 = \sum_{u_i \in M'_x} (b_i + r_i)$. Thus the total number of edges in H is at least $s_1 + s_2 = \frac{1}{2} \sum_{u_i \in M'_x} (\vartheta - t - 1 - b_i) + \sum_{u_i \in M'_x} (b_i + r_i)$. Re-arranging the terms in the summation and simplifying, we get

$$|E(H)| \geq \sum_{u_i \in M'_x} \frac{(\vartheta - t - 1)}{2} + \sum_{u_i \in M'_x} \frac{b_i}{2} + \sum_{u_i \in M'_x} r_i$$

Since $|M'_x| = \vartheta - t$, we get

$$|E(H)| \geq \frac{(\vartheta - t)(\vartheta - t - 1)}{2} + \sum_{u_i \in M'_x} \frac{b_i}{2} + \sum_{u_i \in M'_x} r_i \tag{7.1}$$

Step 2: The objective here is to delete some edges from H to make sure that in the resultant graph H'' (i) there are no edges of the form (u_i, u_{j+1}) where u_i, u_j both belong to M'_x with $j > i$ (ii) the induced subgraph on M'_x is triangle free and (iii) there are at least $\frac{(\vartheta-t)(\vartheta-t-1)}{2}$ edges.

Construction of H'' is done in two stages. Initialize $H' = H$ and consider the edges of H one by one and if the edge being considered is violating condition (i) above, delete that edge from H'. Once all the edges of H have been processed this way, it is clear that H' will satisfy condition (i). Now, initialize $H'' = H'$ and repeat the following procedure until the induced subgraph of H'' on M'_x becomes triangle free: if $u_i, u_j, u_k \in M'_x$ and they induce a triangle in H', we know that at least two edges of this triangle have the same color, because G has no Gallai triangles. We choose two edges e_1, e_2 of this triangle such that color(e_1) = color(e_2). Since this color is repeating at the common end point of e_1 and e_2, at least one of these edges would have got a red-flag from their common end point. We delete one of the edges e_1 and e_2, making sure that the deleted edge had got a red-flag from the common end point of e_1 and e_2. It is clear that H'' satisfies both conditions (i) and (ii). We claim that H'' has at least $\frac{(\vartheta-t)(\vartheta-t-1)}{2}$ edges.

Consider an edge $e \in E(H) \setminus E(H')$. From the procedure we followed in Step 2, e is an edge from u_i to u_{j+1}, where $u_i, u_j \in M'_x$ with $j > i$. First note that color$(u_i, u_{j+1}) \neq$ color(u_j, u_{j+1}), since color$(u_i, u_{j+1}) \notin C(P)$ by the construction of H and color$(u_j, u_{j+1}) \in C(P)$. We claim that $(u_i, u_j) \notin E(H)$. If this was not the case, since G has no Gallai triangles, color$(u_i, u_j) \in \{$color(u_j, u_{j+1}), color$(u_i, u_{j+1})\}$. If color$(u_i, u_j) =$ color(u_j, u_{j+1}), which is an old color, we have $(u_i, u_j) \notin E(H)$, a contradiction. Now, consider the case when color$(u_i, u_j) =$ color(u_i, u_{j+1}). By Lemma 7.22, we know that $u_{j+1} \notin M'_x$. Therefore, since $u_j \in M'_x$ with $j > i$, if color$(u_i, u_j) =$ color(u_i, u_{j+1}), by our preference rules while adding edges to H, the edge color(u_i, u_j) would have got preference over the edge (u_i, u_{j+1}), and (u_i, u_{j+1}) would not have been added to $E(H)$ while considering u_i, raising a contradiction.

Thus, with each deleted edge (u_i, u_{j+1}), where $u_i, u_j \in M'_x$ with $j > i$, we can associate the missing edge (u_i, u_j) of the graph H. This implies that, there is an injective mapping from $E(H) \setminus E(H')$ to the set of missing edges between vertices in M'_x in H. Therefore we get $|E(H) \setminus E(H')| \leq$ the number of missing edges between vertices in M'_x in the graph H. But, by the definition of b_i, the number of missing edges in H between vertices in M'_x is $\sum_{u_i \in M'_x} \frac{b_i}{2}$. This gives, $|E(H')| \geq |E(H)| - \sum_{u_i \in M'_x} \frac{b_i}{2}$. Thus, using inequality 7.1 above, we get:

$$|E(H')| \geq \frac{(\vartheta - t)(\vartheta - t - 1)}{2} + \sum_{u_i \in M'_x} r_i \tag{7.2}$$

Consider an edge $e \in E(H') \setminus E(H'')$. By our construction, e was part of a triangle in H formed by three vertices in M'_x, and color$(e) =$ color(e') for another edge e' of this triangle and the common end point u_i of e and e' had placed a red flag on e. Thus, each edge $e \in E(H') \setminus E(H'')$ had a red flag on it. Therefore, $|E(H') \setminus E(H'')| \leq$ the total number of edges with red flags on them, which is at most $\sum_{u_i \in M'_x} r_i$ by the definition of r_i. From this we get, $|E(H'')| \geq |E(H')| - \sum_{u_i \in M'_x} r_i$.

Thus, by inequality 7.2 above, we have $|E(H'')| \geq \frac{(\vartheta-t)(\vartheta-t-1)}{2}$ and thus, H'' satisfies all the properties (i), (ii) and (iii) stated at the beginning of Step 2.

Now, we proceed to prove the lemma. Since the induced subgraph of H'' on M'_x is triangle free, this induced subgraph has at most $\left\lfloor \frac{|M'_x|^2}{4} \right\rfloor = \left\lfloor \frac{(\vartheta-t)^2}{4} \right\rfloor$ edges, by Turan's [2] theorem [40]. Since all edges in H'' have one of their end points in M'_x, the number of edges in H'' with exactly one end point in M'_x and the other end point in $V(H'') \setminus M'_x$ is at least $|E(H'')| - \left\lfloor \frac{|M'_x|^2}{4} \right\rfloor$ which is at least $\frac{(\vartheta-t)(\vartheta-t-1)}{2} - \frac{(\vartheta-t)^2}{4} = \frac{(\vartheta-t)^2}{4} - \frac{(\vartheta-t)}{2}$. By pigeonhole principle, this implies that there exists a vertex $u_m \in M'_x$ such that there are at least $\left\lceil \frac{(\vartheta-t-2)}{4} \right\rceil$ edges in H'' incident at u_m with their other end point in $V(H'') \setminus M'_x$. By construction, we have made sure that none of these edges have a color from the set χ_m and none of them have a vertex $u_{m'+1}$ as their other end point, where $u_{m'} \in M'_x$

[2] A triangle free graph on n vertices has at most $\left\lfloor \frac{n^2}{4} \right\rfloor$ edges

with $m' > m$. We also have $V(H'') \subseteq V(P)$, because we had $V(H) \subseteq V(P)$ to start with. Thus, the lemma holds. $\square$

Let G be a Gallai colored graph. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. Let $u_m \in M'_x$ be the vertex satisfying the conditions specified in Lemma 7.23 and let $\Psi(u_m)$ be defined as in Definition 7.4. The observation below follows from Lemma 7.10, by noting that $u_m \in M_P(x)$.

Observation 7.3. *For each $u_j \in \Psi(u_m)$, $\text{color}(u_j, u_{j+1})$ belongs to $OLD_{\notin y} \uplus OLD_{y \to P}$.*

We use the following lemma very crucially for obtaining the bound $\lambda(G) \geq \left\lfloor \frac{13\vartheta}{17} \right\rfloor$. Recall that $COLOR_{y \to P} = \{\text{color}(y, u_i) \mid u_i \in N(y) \cap V(P)\}$.

Observation 7.4. $|COLOR_{y \to P}| \geq 2(\vartheta - t) + \left\lceil \frac{(\vartheta - t - 2)}{4} \right\rceil$*. Moreover, there are at least $\vartheta - t$ new colors in the set $COLOR_{y \to P}$.*

Proof. By the definition of $\Psi(u_m)$, from Lemma 7.23 we get $|\Psi(u_m) \setminus M'_x| \geq \left\lceil \frac{(\vartheta - t - 2)}{4} \right\rceil$. Since $|M'_x| \geq \vartheta - t$, this implies, $|M'_x \cup \Psi(u_m)| \geq (\vartheta - t) + \left\lceil \frac{(\vartheta - t - 2)}{4} \right\rceil$. This gives $|OLD_{\notin y} \uplus OLD_{y \to P}| \geq (\vartheta - t) + \left\lceil \frac{(\vartheta - t - 2)}{4} \right\rceil$, by Lemma 7.22 and Observation 7.3. Now, by Lemma 7.2, $|COLOR_{y \to P}| \geq 2(\vartheta - t) + \left\lceil \frac{(\vartheta - t - 2)}{4} \right\rceil$. Moreover, there are at least $\vartheta - t$ new colors in the set $COLOR_{y \to P}$, because the color degree of y is at least ϑ and all edges of new colors incident at y have their other end points in $V(P)$, by Lemma 7.1. $\square$

The above observation allows us to make the following definition.

Definition 7.5. Let G be a Gallai colored graph. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. Let $D(y)$ represent a set of $2(\vartheta - t) + \left\lceil \frac{(\vartheta - t - 2)}{4} \right\rceil$ neighbors of y in $V(P)$ such that no two edges from y to $D(y)$ have the same color and at least $\vartheta - t$ of them are of new colors. For some i and j with $0 \leq i \leq j < t$, the consecutive set of vertices $u_i, u_{i+1}, \ldots, u_j$ of the heterochromatic path P are said to form a block of neighbors of y in P if $u_{j+1} \notin D(y)$ and if $i > 0$, u_{i-1} also does not belong to $D(y)$, but for each $i \leq k \leq j$, $u_k \in D(y)$. We denote this block as $B_{i,j}$.

Lemma 7.24. *Let G be a Gallai colored graph. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. The blocks of neighbors of y in P partition the set $D(y)$. If $u_i, u_j \in D(y)$ with $i \neq j$ such that both (y, u_i) and (y, u_j) are of new colors, then u_i and u_j must belong to two different blocks.*

Proof. The first part of this lemma directly follows from the definition of $D(y)$. Suppose $u_i, u_j \in D(y)$ with $i \neq j$ such that both (y, u_i) and (y, u_i) are of new colors. Without loss of generality, assume that $i < j$. For contradiction, assume that both u_i and u_j belong to the same block of neighbors of y in P. This implies that all vertices in the subpath $P' = u_i, u_{i+1}, \ldots, u_j$ of the heterochromatic path P are neighbors of y in G, through distinctly colored edges from y. Notice that for each edge (u_k, u_{k+1}) of the subpath P', the vertex triple (u_k, u_{k+1}, y) induce a triangle in G, which is not a Gallai triangle. We know that the edges in P' are colored with $(j-i)$ distinct old colors, the edges $(u_i, y), (u_{i+1}, y), \ldots, (u_j, y)$ are all distinctly colored, and both (y, u_i) and (y, u_j) are of new colors. To avoid Gallai triangles, each of the $(j-i)$ distinct old colors that occurred on the subpath P' should appear on the edges $(u_{i+1}, y), (u_{i+2}, y), \ldots, (u_{j-1}, y)$, which are only $j-i-1$ in number. Since this is impossible, we can infer that u_i and u_j cannot belong to the same block of neighbors of y in P. □

Theorem 7.25. *Let G be a Gallai colored graph with minimum color degree ϑ. Then G contains a heterochromatic path of length at least $\left\lfloor \frac{13\vartheta}{17} \right\rfloor$.*

Proof. Let P be a maximum length heterochromatic path in G and be given by $x = u_0, u_1, \ldots, u_t = y$. Let $D(y) = B_{i_1,j_1} \uplus B_{i_2,j_2} \uplus \cdots \uplus B_{i_k,j_k}$ be the partition of $D(y)$ into blocks, such that $0 \le i_1 \le j_1 < i_2 \le j_2 < i_3 \cdots < i_k \le j_k < t$. From this we get, $\sum_{l=1}^{k} (|B_{i_l,j_l}| + 1) \le t+1$, the number of vertices in the path P. Since $\sum_{l=1}^{k} |B_{i_l,j_l}| = |D(y)| = 2(\vartheta - t) + \left\lceil \frac{(\vartheta-t-2)}{4} \right\rceil$, we get

$$2(\vartheta - t) + \left\lceil \frac{(\vartheta - t - 2)}{4} \right\rceil + k \le t + 1$$

By the definition of $D(y)$, there are at least $\vartheta - t$ distinctly colored edges with new colors from y to $D(y)$ and using Lemma 7.24, we can infer that the number of blocks $k \ge \vartheta - t$. Therefore, the above inequality gives,

$$3(\vartheta - t) + \left\lceil \frac{(\vartheta - t - 2)}{4} \right\rceil \le t + 1$$

This implies, $t \ge \left\lceil \frac{13\vartheta}{17} - \frac{6}{17} \right\rceil \ge \left\lfloor \frac{13\vartheta}{17} \right\rfloor$ □

7.5 Conclusion

We have shown that when the girth of a graph G is as high as $4\log_2(\vartheta) + 2$, it contains a heterochromatic path of length at least $\vartheta - 2$, which is only one less than the bound conjectured by Chen and Li [32]. A weaker requirement that G just does not contain four cycles is enough to guarantee a lower bound of $\vartheta - o(\vartheta)$. We have also shown that if G has no heterochromatic triangles, then it contains a heterochromatic path of length at least $\left\lfloor \frac{13\vartheta}{17} \right\rfloor$, an improvement over the existing result. The conjecture of $\vartheta - 1$ lower bound by Chen and Li [32] for the length of maximum heterochromatic paths in general graphs remains open.

Chapter 8

Conclusion

In the first part of this thesis, we studied algorithmic questions on the boxicity and cubicity of graphs. In the general case, we have exhibited polynomial time $o(n)$ factor approximation algorithms for computing the boxicity and cubicity. As a corollary, a $o(n)$ factor approximation algorithm for computing the partial order dimension of finite posets and a $o(n)$ factor approximation algorithm for computing the threshold dimension of split graphs were also derived. Since polynomial time approximations for any of these problems within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$ is considered unlikely, no significant improvement is expected in the approximation factor. We have given FPT approximations for boxicity with several interesting edit distance parameters. Though it is known that boxicity and cubicity are FPT with respect to minimum vertex number as the parameter, similar results for other edit distance parameters are not known yet. Moreover, except for the minimum vertex cover number parameter, FPT approximation algorithms for cubicity are not known with other edit distance parameters.

For many special graph classes including bipartite, co-bipartite and split graphs, the hardness result as in the general case holds for both boxicity and cubicity. Not many approximation algorithms for these problems were known previously for any well-known graph class. We have given polynomial time algorithms for a constant factor approximation for computing the boxicity of circular arc graphs, and an additive two approximation for computing the boxicity of some important subclasses of circular arc graphs. A polynomial time algorithm for approximating the cubicity of circular arc graphs is also obtained, which gives a constant factor approximation up to an additive error of $\log n$. We believe that computing the boxicity and the cubicity of circular arc graphs is NP-Hard; however, obtaining a formal proof for this fact remains open.

We obtained a constant factor approximation algorithm for computing the cubicity of trees which runs in deterministic polynomial time and a randomized algorithm running in polynomial time to get the corresponding cube representation. Devising a deterministic algorithm for the latter part or derandomizing our randomized algorithm would be an interesting problem. It is not yet clear whether computing the cubicity of trees is NP-Hard. We believe that recognizing trees of cubicity two (i.e, recognizing trees which are intersection graphs of axis-parallel squares) itself is NP-Hard.

The second part of this thesis described a polynomial time algorithm to add edges to a connected outerplanar graph G of pathwidth p to produce a supergraph of G, which is 2-vertex-connected, outerplanar and of pathwidth $O(p)$. Using this result, a constant factor approximation algorithm for computing minimum height planar straight line grid drawings of outerplanar graphs can be derived, by extending the existing algorithm for 2-vertex connected outerplanar graphs. This closes an open problem raised by Biedl [14]. The factor of approximation could be improved, if it was possible to better optimize the procedure of edge addition so that the blow up in the pathwidth is still lower.

In the third part of this thesis, we studied the cardinality of fixed orientation equilateral triangle matchings of point sets in general position. This was done by studying the structural and geometric properties of an associated geometric graph, which is the same as a triangle distance Delaunay graph of the point set and is also equivalent to a half-θ_6 graph of the point set. It was shown that for a set of n points in general position, the cardinality of maximum matchings in TD Delaunay graphs (equivalently half-θ_6 graphs) is at least $\left\lceil \frac{n-1}{3} \right\rceil$ and there are point sets for which this bound is tight.

It was also shown that for a set of n points in general position, its θ_6 graph can have at most $5n - 11$ edges. It is still an open problem to decide whether this upper bound is tight, since we do not have any examples for which the number of edges exceeds $\left(4 + \frac{1}{3}\right) n - 13$. It is an interesting question to see whether for every point set in general position, its Θ_6 graph contains a matching of size $\left\lfloor \frac{n}{2} \right\rfloor$. So far, we were not able to get any counter examples for this claim.

In the last part of the thesis, we studied lower bounds for $\lambda(G)$-the length of maximum heterochromatic paths in edge colored graphs. We showed that for graphs without four cycles, $\lambda(G) \geq \vartheta(G) - o(\vartheta(G))$, where $\vartheta(G)$ is the minimum color degree of G. If the girth is at least $4\log_2(\vartheta(G)) + 2$, then we obtained $\lambda(G) \geq \vartheta(G) - 2$. The conjecture that $\lambda(G) \geq \vartheta(G) - 1$ for any edge colored graph G is still open.

Bibliography

[1] Ábrego, B.M., Arkin, E., FernÃąndez-Merchant, S., Hurtado, F., Kano, M., Mitchell, J., Urrutia, J.: Matching points with squares. Discrete and Computational Geometry **41**, 77–95 (2009)

[2] Abueida, A.A., Busch, A.H., Sritharan, R.: A min-max property of chordal bipartite graphs with applications. Graphs and Combinatorics **26**(3), 301–313 (2010)

[3] Adiga, A., Bhowmick, D., Chandran, L.S.: The hardness of approximating the boxicity, cubicity and threshold dimension of a graph. Discrete Appl. Math. **158**, 1719–1726 (2010)

[4] Adiga, A., Bhowmick, D., Chandran, L.S.: Boxicity and poset dimension. SIAM J. Discrete Math. **25**(4), 1687–1698 (2011)

[5] Adiga, A., Chandran, L.S.: Cubicity of interval graphs and the claw number. J. Graph Theory **65**, 323–333 (2010)

[6] Adiga, A., Chandran, L.S., Sivadasan, N.: Lower bounds for boxicity. To appear in Combinatorica. CoRR **abs/0806.3175** (2008)

[7] Adiga, A., Chitnis, R., Saurabh, S.: Parameterized algorithms for boxicity. In: ISAAC, pp. 366–377 (2010)

[8] Albert, M., Frieze, A., Reed, B.: Multicoloured Hamilton cycles. Electr. J. Comb. **2**, #R10 (1995)

[9] Alon, N., Linial, S.H.N.: The Moore bound for irregular graphs. Graphs and Combinatorics **18**, 53–57 (2002)

[10] Bellatoni, S., Hartman, I.B.A., Przytycka, T., Whitesides, S.: Grid intersection graphs and boxicity. Discrete Math. **114**, 41–49 (1993)

[11] Bereg, S., Mutsanas, N., Wolff, A.: Matching points with rectangles and squares. Comput. Geom. Theory Appl. **42**(2), 93–108 (2009)

[12] Bhowmick, D., Chandran, L.S.: Boxicity of circular arc graphs. Graphs and Combinatorics **27**, 769–783 (2011)

[13] Biedl, T.: Small drawings of outerplanar graphs, series-parallel graphs, and other planar graphs. Discrete Comput. Geom. **45**(1), 141–160 (2011)

[14] Biedl, T.: A 4-approximation for the height of 2-connected outer-planar graph drawings. WAOA 2012 (2012)

[15] Bonichon, N., Gavoille, C., Hanusse, N., Ilcinkas, D.: Connections between theta-graphs, Delaunay triangulations, and orthogonal surfaces. In: Proceedings of the 36th international conference on Graph-theoretic concepts in computer science, WG'10, pp. 266–278. Springer-Verlag (2010)

[16] Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. **13**(3), 335–379 (1976)

[17] Bose, P., Carmi, P., Collette, S., Smid, M.: On the stretch factor of convex Delaunay graphs. Journal of Computational Geometry **1**, 41–56 (2010)

[18] Breu, H., Kirkpatrick, D.G.: Unit disk graph recognition is NP-hard. Computational Geometry **9**, 2–34 (1998)

[19] Broersma, H., Li, X., Woeginger, G., Zhang, S.: Paths and cycles in colored graphs. Australasian J. Combin. **31**, 297–309 (2005)

[20] Bruhn, H., Chopin, M., Joos, F., Schaudt, O.: Structural parameterizations for boxicity. To appear in WG'14. CoRR **abs/1402.4992** (2014)

[21] Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Inf. Process. Lett. **58**(4), 171–176 (1996)

[22] Cai, L.: Parameterized complexity of vertex colouring. Discrete Applied Mathematics **127**(3), 415–429 (2003)

[23] Cameron, K., Sritharan, R., Tang, Y.: Finding a maximum induced matching in weakly chordal graphs. Discrete Math. **266**, 133–142 (2003)

[24] Cao, Y., Chen, J., Liu, Y.: On feedback vertex set: New measure and new structures. In: H. Kaplan (ed.) SWAT, pp. 93–104. Springer (2010)

[25] Chalermsook, P., Laekhanukit, B., Nanongkai, D.: Graph products revisited: Tight approximation hardness of induced matching, poset dimension and more. SODA 2013, pp. 1557–1576 (2013)

[26] Chandran, L., Mannino, C., Oriolo, G.: On the cubicity of certain graphs. Inf. Process. Lett. **94**(3), 113–118 (2005)

[27] Chandran, L.S., Das, A., Shah, C.D.: Cubicity, boxicity, and vertex cover. Discrete Mathematics **309**(8), 2488–2496 (2009)

[28] Chandran, L.S., Mathew, K.A.: An upper bound for cubicity in terms of boxicity. Discrete Mathematics **309**(8), 2571–2574 (2009)

[29] Chandran, L.S., Sivadasan, N.: Boxicity and treewidth. J. Comb. Theory Ser. B **97**, 733–744 (2007)

[30] Chartrand, G., Harary, F.: Planar permutation graphs. Ann. Inst. Henri Poincaré, Nouv. Sér., Sect. B **3**, 433–438 (1967)

[31] Chen, H., Li, X.: Color degree and color neighborhood union conditions for long heterochromatic paths in edge-colored graphs. CoRR **abs/0512144** (2005)

[32] Chen, H., Li, X.: Long heterochromatic paths in edge-colored graphs. Electr. J. Comb. **12**, #R33 (2005)

[33] Chen, H., Li, X.: Color neighborhood union conditions for long heterochromatic paths in edge-colored graphs. Electr. J. Comb. **14**(1), #R77 (2007)

[34] Chen, H., Li, X.: Long heterochromatic paths in heterochromatic triangle free graphs. Utilitas Mathematica **85**, 3–11. (An older version is available at http://arxiv.org/abs/0804.4526) (2011)

[35] Chew, L.P.: There are planar graphs almost as good as the complete graph. Journal of Computer and System Sciences **39**(2), 205–219 (1989)

[36] Chvátal, V., Hammer, P.L.: Aggregation of inequalities in integer programming. Annals of Discrete Mathematics: Studies in Integer Programming **1**, 145–162 (1977)

[37] Clarkson, K.: Approximation algorithms for shortest path motion planning. In: Proceedings of the nineteenth annual ACM symposium on Theory of computing, STOC '87, pp. 56–65. ACM (1987)

[38] Cozzens, M.B.: Higher and multi-dimensional analogues of interval graphs. Ph.D. thesis, Department of Mathematics, Rutgers University, New Brunswick, NJ (1981)

[39] Das, A., Suresh, P., Subrahmanya, S.V.: Rainbow path and minimum degree in properly edge colored graphs. In: Eurocomb (2013. (Also available at http://arxiv.org/abs/1312.5067))

[40] Diestel, R.: Graph Theory Fourth Edition. Springer-Verlag (2010)

[41] Dillencourt, M.: Toughness and Delaunay triangulations. In: Proceedings of the third annual symposium on Computational geometry, SCG '87, pp. 186–194. ACM (1987)

[42] Dujmovic, V., Morin, P., Wood, D.R.: Path-width and three-dimensional straight-line grid drawings of graphs. In: Revised Papers from the 10th International Symposium on Graph Drawing, GD '02, pp. 42–53. Springer-Verlag, London, UK (2002)

[43] Dushnik, B., Miller, E.W.: Partially ordered sets. American Journal of Mathematics **63**(3), 600–610 (1941)

[44] Erdös, P., Nesetril, J., Rödl, V.: On some problems related to partitions of edges of a graph. Graphs and other combinatorial topics, Proc. 3rd Czech. Symp., Prague 1982, Teubner-Texte Math. 59 pp. 54–63 (1983)

[45] Erdös, P., Tuza, Z.: Rainbow subgraphs in edge-colorings of complete graphs. Annals of Discrete Mathematics **55**, 81–88 (1993)

[46] Esperet, L.: Boxicity of graphs with bounded degree. Eur. J. Comb. **30**(5), 1277–1280 (2009)

[47] Feder, T., Hell, P., Huang, J.: List homomorphisms and circular arc graphs. Combinatorica **19**, 487–505 (1999)

[48] Fellows, M., Lokshtanov, D., Misra, N., Mnich, M., Rosamond, F., Saurabh, S.: The complexity ecology of parameters: An illustration using bounded max leaf number. Theor. Comp. Sys. **45**(4), 822–848 (2009)

[49] Fellows, M.R., Hermelin, D., Rosamond, F.A.: Well-quasi-orders in subclasses of bounded treewidth graphs. In: IWPEC, pp. 149–160 (2009)

[50] Felsner, S., Francis, M.C.: Contact representations of planar graphs with cubes. In: Proceedings of the 27th ACM Symposium on Computational Geometry, SoCG '11, pp. 315–320 (2011)

[51] Frieze, A., Reed, B.: Polychromatic Hamilton cycles. Discrete Mathematics **118**(1–3), 69–74 (1993)

[52] Ganian, R.: Twin-cover: Beyond vertex cover in parameterized algorithmics. In: Proceedings of the 6th International Conference on Parameterized and Exact Computation, IPEC'11, pp. 259–271. Springer-Verlag, Berlin, Heidelberg (2012)

[53] GarcÃŋa, A., Hurtado, F., Noy, M., Tejel, J.: Augmenting the connectivity of outerplanar graphs. Algorithmica **56**(2), 160–179 (2010)

[54] Garey, M.R., Johnson, D.S., Miller, G.L., Papadimitriou, C.H.: The complexity of coloring circular arcs and chords. SIAM J. Alg. Disc. Meth. **1**(2), 216–227 (1980)

[55] Golumbic, M.C., Lewenstein, M.: New results on induced matchings. Discrete Appl. Math. **101**, 157–165 (2000)

[56] Govindan, R., Langston, M.A., Yan, X.: Approximating the pathwidth of outerplanar graphs. Inf. Process. Lett. **68**(1), 17–23 (1998)

[57] Grohe, M.: Computing crossing numbers in quadratic time. J. Comput. Syst. Sci. **68**, 285–302 (2004)

[58] Gyarfas, A., Ruszinko, M., Sarkozy, G.N., Schelp, R.H.: Long rainbow cycles in proper edge-colorings of complete graphs. Australasian Journal Of Combinatorics **50**, 45–53 (2011)

[59] Hahn, G., Thomassen, C.: Path and cycle sub-ramsey numbers and an edge-colouring conjecture. Discrete Mathematics **62**, 29–33 (1986)

[60] Hopcroft, J., Tarjan, R.: Algorithm 447: Efficient algorithms for graph manipulation. Commun. ACM **16**(6), 372–378 (1973)

[61] Kant, G.: Augmenting outerplanar graphs. Journal of Algorithms **21**(1), 1–25 (1996)

[62] Keil, J.M.: Approximating the complete Euclidean graph. In: No. 318 on SWAT 88: 1st Scandinavian workshop on algorithm theory, pp. 208–213. Springer-Verlag (1988)

[63] Kostochka, A.V., Yancey, M.: Large rainbow matchings in edge-coloured graphs. Combinatorics, Probability & Computing **21**(1–2), 255–263 (2012)

[64] Kratochvíl, J.: A special planar satisfiability problem and a consequence of its NP-completeness. Discrete Appl. Math. **52**(3), 233–252 (1994)

[65] Krauthgamer, R., Lee, J.R.: The intrinsic dimensionality of graphs. Combinatorica **27**(5), 551–585 (2007)

[66] Lin, M.C., Szwarcfiter, J.L.: Characterizations and recognition of circular-arc graphs and subclasses: A survey. Discrete Mathematics **309**(18), 5618–5635 (2009)

[67] Linial, N., London, E., Rabinovich, Y.: The geometry of graphs and some of its algorithmic applications. Combinatorica **15**(2), 215–245 (1995)

[68] Maffray, F.: On the coloration of perfect graphs. In: Recent Advances in Algorithmic Combinatorics, pp. 65–84. CMS books in Mathematics (2003)

[69] Marx, D.: Parameterized coloring problems on chordal graphs. Theor. Comput. Sci. **351**(3), 407–424 (2006)

[70] Marx, D., Schlotter, I.: Obtaining a planar graph by vertex deletion. Algorithmica **62**(3–4), 807–822 (2012)

[71] Mazoit, F.: The branch-width of circular-arc graphs. In: LATIN, pp. 727–736 (2006)

[72] McConnell, R.M.: Linear-time recognition of circular-arc graphs. Algorithmica **37**(2), 93–147 (2003)

[73] Narasimhan, G., Smid, M.: Geometric Spanner Networks. Cambridge University Press (2007)

[74] Niedermeier, R.: Invitation to fixed-parameter algorithms (2002)

[75] Nishizeki, T.: Lower bounds on the cardinality of the maximum matchings of planar graphs. Discrete Mathematics **28**, 255–267 (1979)

[76] Panahi, F., Mohades, A., Davoodi, M., Eskandari, M.: Weak matching points with triangles. In: Proceedings of the 23rd Annual Canadian Conference on Computational Geometry (2011)

[77] Quest, M., Wegner, G.: Characterization of the graphs with boxicity $\leq$ 2. Discrete Math. **81**, 187–192 (1990)

[78] Roberts, F.S.: On the boxicity and cubicity of a graph. In: Recent Progresses in Combinatorics, pp. 301–310. Academic Press, New York (1969)

[79] Robertson, N., Seymour, P.D.: Graph minors. iii. planar tree-width. J. Comb. Theory, Ser. B **36**(1), 49–64 (1984)

[80] Rosgen, B., Stewart, L.: Complexity results on graphs with few cliques. Discrete Mathematics and Theoretical Computer Science **9**, 127–136 (2007)

[81] Scheinerman, E.R.: Intersection classes and multiple intersection parameters of graphs. Ph.D. thesis, Princeton University (1984)

[82] Schnyder, W.: Embedding planar graphs on the grid. In: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms, SODA '90, pp. 138–148 (1990)

[83] Shah, C.D.: Boxicity, cubicity, and vertex cover. M. Sc Thesis, IISc Bangalore **http://clweb.csa.iisc.ernet.in/chintan/chintan_thesis.pdf** (2008)

[84] Shrestha, A.M.S., Tayu, S., Ueno, S.: On orthogonal ray graphs. Discrete Appl. Math. **158**(15), 1650–1659 (2010)

[85] Skodinis, K.: Construction of linear tree-layouts which are optimal with respect to vertex separation in linear time. J. Algorithms **47**(1), 40–59 (2003)

[86] Soulignac, F.: On proper and Helly circular-arc graphs. Ph.D. thesis, Universidad de Buenos Aires (2010)

[87] Suchan, K., Todinca, I.: Pathwidth of circular-arc graphs. In: WG'07, pp. 258–269 (2007)

[88] Suderman, M.: Pathwidth and layered drawings of trees. Int. J. Comput. Geometry Appl. **14**(3), 203–225 (2004)

[89] Sundaram, R., Singh, K.S., Rangan, C.P.: Treewidth of circular-arc graphs. SIAM J. Discret. Math. **7**, 647–655 (1994)

[90] Syslo, M.M.: Characterizations of outerplanar graphs. Discrete Mathematics **26**(1), 47–53 (1979)

[91] Thomassen, C.: Interval representations of planar graphs. J. Comb. Theory Ser. B **40**, 9–20 (1986)

[92] Trotter Jr., W.T.: Combinatorial problems in dimension theory for partially ordered sets. Problèmes combinatoires et théorie des graphes, Colloque Internationeaux C.N.R.S. (260), 403–406 (1978)

[93] Trotter Jr., W.T.: A characterization of Robert's inequality for boxicity. Discrete Math. **28**(3), 303–313 (1979)

[94] Tucker, A.C.: Matrix characterizations of circular-arc graphs. Pacific J. of Mathematics **19**, 535–545 (1971)

[95] Tucker, A.C.: Structure theorems for some circular-arc graphs. Discrete Mathematics **7**(1,2), 167–195 (1974)

[96] Villanger, Y.: Proper interval vertex deletion. In: V. Raman, S. Saurabh (eds.) IPEC'10, pp. 228–238. Springer (2010)

[97] Villanger, Y., Heggernes, P., Paul, C., Telle, J.A.: Interval completion is fixed parameter tractable. SIAM J. Comput. **38**(5), 2007–2020 (2008)

[98] Yannakakis, M.: The complexity of the partial order dimension problem. SIAM J. Alg. Disc. Meth. **3**(3), 351–358 (1982)

[99] Yu, C.W., Chen, G.H., Ma, T.H.: On the complexity of the chain subgraph cover problem. Theor. Comput. Sci. **205**(1-2), 85–98 (1998)

Index