Biclique Partition Problem

Supriya Tiwari

Department of Mathematics

Indian Institute of Science Education and Research, Pune

Supervisor

Dr. Andreas Karrenbauer
Max Planck Institute for Informatics, Saarbrucken

submitted to
Indian Institute of Science Education and Research Pune in
partial fulfillment of the requirements for the

BS-MS Dual Degree Programme

April, 2019

Certificate

This is to certify that this dissertation entitled Biclique Partition Problem should appear Heretowards the partial fulfillment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Supriya Tiwari at Indian Institute of Science Education and Research Pune under the supervision of Dr. Andreas Karrenbauer, Department of Algorithms and Complexity, Max Planck Institute for Informatics during the academic year 2018-2019.

Dr. Andreas Karrenbauer

Committee:

Dr. Andreas Karrenbauer

Dr. Soumen Maity

Declarations

I hereby declare that the matter embodied in the report entitled Biclique Partition Problem should Appear Here are the results of the work carried out by me at the Department of Mathematics, Indian Institute of Science Education and Research, Pune, under the supervision of Dr. Andreas Karrenbauer and the same has not been submitted elsewhere for any other degree.

Supriya Tiwari

Acknowledgements

I would first like to thank my thesis advisor Dr. Andreas Karrenbauer for the patient guidance, encouragement and advice he has provided throughout my time as his student. He always encouraged me to ask my own research questions and at the same time, steered me in the right direction as when he deemed necessary. His invaluable insights has been a cornerstone in the development of this thesis.

I take this opportunity to thank Dr. Soumen Maity. As my teacher, mentor and member of my thesis advisory committee, he has taught me more than I could ever give him credit for here. I am gratefully indebted to his constant guidance over the past years.

I would also like to thank the Department of Algorithms and Complexity at Max Planck Institute for Informatics for providing me with financial support which allowed me to undertake this research.

Finally, I must express my very profound gratitude to my parents for the unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Abstract

A lot of problems that arise in nature can be modelled using graphs. In this thesis, we look at one such problem from display optimization which can be articulated through Biclique Partition problem on Bipartite graphs(BPP). In BPP, given an arbitrary bipartite graph G and an integer k, we have to determine whether there exists a partition of the edge set of G of size k such that each part in the partition is a complete bipartite subgraph. It has been proven that this problem is NP-complete but fixed parameter tractable with exponential kernel. In this thesis, we try to answer if the size of the kernel can be bettered. To do this, we explore tools from partially ordered sets. We present results on how partially ordered sets can induce structures in the solution instances and how these structures can lead to a better kernel.

We also state and prove how the previous kernel does not lead to a polynomial kernel. This gives an incentive to look at certain graph classes and answer if on them BPP is polynomial time solvable or not. This helps us in understanding properties crucial in determining solution. We present new results in this direction. In particular, we prove that BPP is NP-hard on perfect elimination bipartite graph.

Contents

Chapter 1

Introduction

We consider the Biclique Partition problem on bipartite graphs. This problem deals with finding a partition of the edge set of the graph such that graph induced on each part of the partition is a complete bipartite graph. A complete bipartite graph is also called a biclique, in short.

A related problem is that of finding a Biclique Cover of a bipartite graph. Here, one has to find a cover of the edge set such that graph induced on each part of the cover is a biclique.

Unlike Biclique Partition problem, Biclique Cover problem is well studied in literature. In the following thesis, we study the former problem and present new results in this direction. All graphs in consideration are bipartite graphs, if not mentioned otherwise.

1.1 Motivations

Both these problems admit various applications such as in bioinformatics ([1], [2]), finite automata [3], database tiling [4], finite automata [5], OLED display [6]

among many.

In the following section, we present how Biclique Partitions of a graph can be utilised in OLED displays. This has been our major interest in pursuing Biclique Partition problem.

1.1.1 OLED displays

Organic Light Emitting Diode (OLED) is an alternative form of LED used for display screens. There are various advantages of OLED screens over commercially used display technologies such as Liquid Crystral Displays (LCD). The images produced by OLED display have very high contrast as well as close to 180 degrees of viewing angle. The material used in OLED displays is physically flexible. Also, the reaction time of pixels in OLED display is within 10 microseconds which is much less than that of human eye. This allows for seamless video streaming.

Presently, the use OLED screens commercially faces certain limitations. But owing to these advantages, there is a lot of research being undertaken in material science to make OLED commercially viable. There are two types of OLED technologies, namely Active matrix (AM) and Passive Matrix (PM). While the former is more expensive to manufacture, the latter faces longevity issues.

We now present the structure of PM OLED display screens and explain why it has limited lifetime.

For simplicity, we consider binary images. The same can be extrapolated to coloured images. A Passive Matrix OLED display screen is a $m \times n$ matrix consisting of a vertical diode for each pixel or element in the matrix. It also consists of a switch for each row and column. The methodology followed is: $(i, j)^{th}$ pixel is 1 (i.e, white) if and only if i^{th} row's and j^{th} switch is 1 (i.e, on). Thus, each pixel cannot be influenced directly.

To tackle this, each row is displayed in one frame. This is done at a sufficiently

high frame rate such that the human eye catches them all in it's response time. Then, the image perceived is the average of all the frames. If the number of frames is large, then the time for which each frame is shun is quite small. Thus, to have the same average at the end, the pixels have to be shun with higher intensity . This leads to decreased life-time of the diode, one of the major limitations of OLED screens.

This arrangement calls for an algorithmic solution to efficiently display the image. We exploit Biclique Partition problem for this task. Consider a particular arrangement of switches in rows and columns. Then, the cross overs of 1's will result in the pixel being 1, other wise 0. Elaborately, we get a matrix such that submatrix restricted on rows and columns whose switches are on is a 1 matrix (each element of the submatrix is 1). This corresponds to a biclique if we think of the image matrix as the adjacency matrix of a bipartite graph. Here, the rows correspond to one vertex-partition of the bipartite graph and the columns correspond to the other. Since one biclique takes one frame, partitioning the graph into minimum number of bicliques will lead to an increased lifetime of the diode. For example, the following matrix can be decomposed into 3 bicliques which requires 3 frames as compared to number of rows of the matrix which is 4.

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Note that the previous strategy of shining one row at a time is also a biclique decompostion of the matrix since a row is a biclique in itself. This is called the trivial biclique decomposition of the graph. Thus, we aim at trying to find

minimum sized biclique partition of a graph.

1.2 Context of the Study

Unfortunately, both Biclique Partition and Cover problem are NP-complete [7][3]. A natural way of dealing with NP-complete problems is to attack the problem from the viewpoint of parameterized complexity.

In parameterized complexity, a secondary measurement associated with an instance is provided along with instance. This secondary measurement is called *parameter*. Conventionally and in this report, this parameter will be denoted by k. This parameter in some way encapsulates the difficulty of solving the problem. A problem is called *fixed-parameter tractable* if there exists an algorithm which solves the problem in $f(k) \cdot n^{O(1)}$ running time. Here, n is the size of input and $f(k)$ is any arbitrary computable function depending only on k. Thus, if parameter size is small we will be able to solve the problem more efficiently. Note that the parameter can be any property of the input instance like a structural property of the input graph or size of the solution.

An related notion of fixed-parameter tractability is *kernalization*. A polynomial time algorithm is called a *kernalization algorithm* if it transforms an arbitrary input instance to another equivalent instance whose input size is bounded by a function $g(k)$. Note that the function g only depends on k. This function $g(k)$ is called *kernel* size. If $g(k)$ is a polynomial then we call it a *polynomial kernel*. The smaller the kernel size, the better.

In this thesis, we explore the parameterized complexity of Biclique Partition problem and present some new results in this regard.

1.3 Overview of the Thesis

We present related work done in Biclique Partition problem in the next chapter. Chapter 3 states formal definitions and terms used throughout the thesis. Chapter 4 details the parameterized complexity of Biclique Partition and Cover problem. This includes reduction rules leading to kernalization and FPT algorithms for both the problems. Chapter 5 introduces the theory of partially ordered sets. This will be used in proving new results in Chapter 6. Chapter 6 also presents the limitation of the new result obtained. Chapter 7 talks about the classical complexity of Biclique Cover and Partition problem on special graph classes. Finally, Chapter 8 concludes the thesis and presents direction of future work.

Chapter 2

Related work

Consider Edge Clique Cover problem (denoted by ECC). Given an arbitrary graph G and an integer k, the problem is to determine whether the edges of G can be covered by k cliques. One can similarly define Edge Clique Partition problem (denoted by EPP). Both the problems are NP-complete and fixed parameter tractable (parameterized by k)[8],[9]. ECC is one of the first problems proven with doubly exponential lower bound on FPT running time and exponential bound on kernel size, assuming Exponential Time Hypothesis[10]. On the other hand, EPP admits an $O(k^2)$ kernel [11].

Following the suite of covering problems, we define Biclique Cover (BCC) problem as the following: Given a graph G and integer k, determine whether the edges of G can be covered by k bicliques. Similarly, we can define Biclique Partition problem (BPP). Both these problems are NP-complete and fixed-parameter tractable on general as well as bipartite graphs [7]. Also, both the problems admit exponential kernel [12]. Like ECC, it has been proven for BCC that the doubly exponential FPT running time and exponential kernel cannot be bettered in [13], assuming Exponential Time Hypothesis. This almost closes the gap between the best known lower bound and upper bound $O^*(2^{2^k \log k + 2k + \log k})$. But for BPP, there exists a

FPT algorithm with running time $O^*(2^{2k^2+k\log k+k})$[13]. It is an open question if the exponential kernel size of BPP can be bettered. This thesis aim to make advancement in this direction.

Analogously, one can define Biclique Vertex Partition problem(BVP). In BVP, given a graph G and an integer k, one has to determine whether the vertices can be partitioned into k bicliques. BVP is also NP-complete. But unlike all the previous covering problems, BVP is W[2] complete [12], hence not fixed-parameter tractable (assuming FPT $\neq$ W[1]).

[13] shows how the exponential kernel size of BCC and BPP can be exploited to obtain an approximation ratio of $O(n/\log n)$.

Chapter 3

Formal Definitions

3.1 Graph terminologies

Definition 3.1. An *undirected graph* G is an ordered triplet $(V(G), E(G), \psi_G)$ comprising of a non-empty set $V(G)$ called *vertices*, a disjoint set $E(G)$ of *edges* and an incidence function ψ_G mapping edges to unordered pair of vertices (not necessarily distinct).

In this thesis, we consider undirected graphs without loops and multiple edges,i.e, ψ_G is one-one and maps edges to a pair of distinct vertices. Thus, G can also be represented by $(V(G), E(G))$ where $E(G) \subseteq \binom{V(G)}{2}$.

Definition 3.2. A graph $G = (V(G), E(G))$ such that $V(G) = X \cup Y$, $X \cap Y = \varnothing$ and elements of $E(G)$ are of the form (x, y) where $x \in X$ and $y \in Y$ is called a *bipartite graph*. Bipartite graphs are represented as $G = (X, Y, E)$.

Definition 3.3. Let $V(G) = \{v_1, v_2, ...v_m\}$. The *adjacency matrix* of the graph G is a $m \times n$ binary matrix whose $(i, j)^{th}$ entry is 1 if and only if $(v_i, v_j) \in E(G)$. Similarly, let $X = \{x_1, x_2, ...x_m\}$ and $Y = \{y_1, y_2, ..., y_n\}$. The *bipartite adjacency matrix* of $G = (X, Y, E)$ is a $m \times n$ binary matrix whose $(i, j)^{th}$ entry is 1 if and only if $(x_i, y_j) \in E(G)$.

For example, consider the following graph

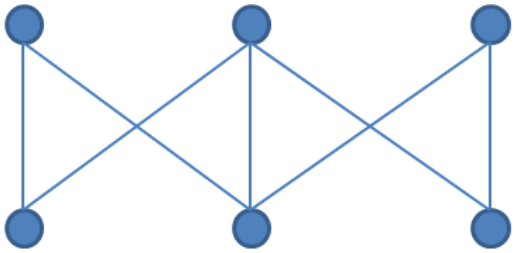

Figure 3.1: Domino graph

The bipartite adjacency matrix of the above graph is

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Definition 3.4. A *biclique* is a bipartite graph such that any two vertices in the two partition are connected by an edge.

Definition 3.5. A *star* is a biclique where one of the partition contains exactly one vertex.

Now we proceed to define biclique partition and cover problem. Consider $G = (X, Y, E)$, a bipartite graph.

Definition 3.6. Let $\mathcal{P} = \{P_1, P_2, ..., P_{|\mathcal{P}|}\}$ be a partition of E. If $G[P_i]$ is a biclique $\forall i \in \{1, 2, .., |\mathcal{P}|\}$, then $\mathcal{P}$ is called a biclique partition of G of size $|\mathcal{P}|$. Given a graph G, the minimum sized biclique partition of G is denoted by $bp(G)$.

Similarly, one can define biclique cover as stated below.

Definition 3.7. Let $\mathcal{C} = \{C_1, C_2, ..., C_{|\mathcal{C}|}\}$ be a cover of E. If $G[C_i]$ is a biclique $\forall i \in \{1, 2, .., |\mathcal{C}|\}$, then $\mathcal{C}$ is called a biclique cover of G of size $|\mathcal{C}|$. Given a graph G, the minimum sized biclique cover of G is denoted by $bc(G)$.

Here, the crucial difference between both the problems is that there are no common edges between two parts of a partition. Below we state the decision version of the biclique partition problem.

BICLIQUE PARTITION:

Input: A bipartite graph $G = (X, Y, E)$

Parameter: An integer k

Problem: Does there exist a biclique partition of G of size at most k?

Decision version of biclique cover problem can be defined similarly. We note that k will be treated as a parameter when treated as a parameterized problem in the later chapters.

Now, we present an example of biclique partition and cover of a graph. Consider the graph in Figure 3.1 again. Below is a minimum biclique partition of size 3 of the domino graph.

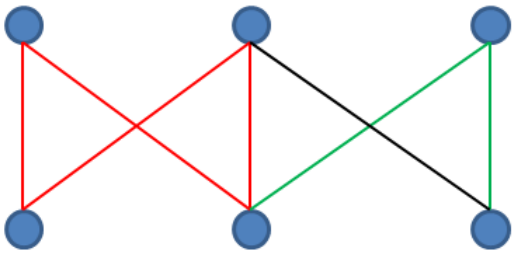

Figure 3.2: Biclique partition of Domino graph of size 3

This is in contrast to the minimum biclique cover of domino graph which is of size 2. Note the common edge shared by both the bicliques.

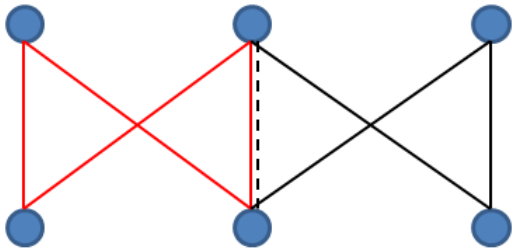

Figure 3.3: Biclique cover of Domino graph of size 2

3.2 Matrix terminologies

Consider A to be a binary matrix of size $m \times n$.

Definition 3.8. The binary rank of A is the minimum k such that $A = B \cdot C$ where B and C are binary matrices of size $m \times k$ and $k \times n$ respectively. We denote binary rank of A by $b(A)$. Note that '$\cdot$' represents multiplication over real numbers.

Definition 3.9. The minimum k such that $A = B \circ C$ where B is a $m \times k$ binary matrix and C is a $k \times n$ binary matrix is called boolean rank of A. Note that '$\circ$' represents boolean multiplication.

The concept of fooling set is formally stated in the context of communication complexity. Below we define fooling set in terms of matrix which suits our purposes.

Definition 3.10. Let $F = \{(x_1, y_1), (x_2, y_2), ..., (x_{|F|}, y_{|F|})\}$ where all $x_i \in [m]$, $y_i \in [n]$ $\forall i \in [|F|]$ are distinct and $A_{x_i y_i} = 1$ $\forall i \in [|F|]$. We call F a fooling set of size $|F|$ if for any two elements $(x_i, y_i), (x_j, y_j)$ in F, either $A_{x_i y_j} = 0$ or $A_{x_j y_i} = 0$.

NOTE: $fool(A)$ is defined to be size of maximum sized fooling set in A. Now we look at the relationship between fooling set and biclique cover/partition problem.

Theorem 3.11. $fool(A) \leq bc(A) \leq bp(A)$

Proof. Consider F to be the maximum sized fooling set in A. Pick any two set of indices from F, say $(x_i, y_i), (x_j, y_j)$. Let the binary matrix A correspond to the adjacency matrix of a bipartite graph. Then, as $A_{x_i y_i} = 1$, $A_{x_i y_i} = 1$ the edges corresponding to (x_i, y_i) and (x_j, y_j) cannot be in the same biclique. This is because either $A_{x_i y_j} = 0$ or $A_{x_j y_i} = 0$. Thus, every element of F is covered in a distinct biclique of the biclique cover. This proves $fool(A) \leq bc(A)$. Since every biclique partition of A is also a biclique cover of A, $bc(A) \leq bp(A)$. Thus , $fool(A) \leq bc(A) \leq bp(A)$ □

3.3 Parameterized complexity

Consider Σ to be a fixed, finite alphabet. Let Σ^* represent the set of all words over Σ. We follow [9] in the following section.

Definition 3.12. A parameterized problem L is a subset of $\Sigma^* \times N$. For an instance $(x, k) \in \Sigma^* \times N$, k is called parameter.

Definition 3.13. A parameterized problem L is called fixed-parameter tractable (in short, FPT) if there exists an algorithm $\mathcal{A}$, a computable function f and a constant c such that given an arbitrary instance (x, k), the algorithm $\mathcal{A}$ outputs whether $(x, k) \in L$ or not in time $O^*(f(k))$ (i.e, $f(k) \cdot n^c$).

A related notion from FPT is that of a kernalization algorithm as defined below.

Definition 3.14. A polynomial time algorithm, $\mathcal{A}$ is called a *kernalization algorithm* if for an arbitrary instance (I, k), $\mathcal{A}$ outputs (I', k') such that: (I, k) is a YES instance iff (I', k') is a YES instance, $|I'| \leq g(k)$ and $|k'| \leq h(k)$.
Note that the functions g, h only depends on k. This function $g(k)$ is called kernel size or simply, *kernel* of the problem. If $g(k)$ is a polynomial then we call it a *polynomial kernel*.

Remark. A parameterized language admits a kernel if and only if it is fixed-parameter tractable.

Chapter 4

Parameterized complexity of Biclique Partition Problem

Both biclique cover and partition problem admit a 2^{2k} kernel using the same set of reduction rules. We present these reduction rules below. As before, let $G = (X, Y, E)$ be a bipartite graph and A be it's adjacency matrix.

4.1 Reduction Rules

Red. 1: If two vertices have same set of neighbours, delete one of the vertex while keeping the parameter unchanged.

Red. 2: If there exists an isolated vertex, then delete that vertex while keeping the parameter unchanged.

Now, we present the viability of these reduction rules.

Lemma 4.1. Red. 1 is safe and can be performed in polynomial time.

Proof. Suppose there exists two vertices $x_1, x_2 \in X$ such that $N(x_1) = N(x_2)$. Let $G' = G - \{x_2\}$. We have to prove (G, k) is a YES instance iff (G', k) is a YES instance.

Suppose (G, k) is a YES instance. Consider the biclique partition of G of size k. We note that removal of a vertex from a biclique results in another biclique. Remove x_2 from all the biclique parts in the partition containing x_2. Since the resultant graph is G', we obtain a biclique partition of size at most k of G'. Thus, (G', k) is a YES instance.

Suppose (G', k) is a YES instance. Thus, there exists a biclique partition of G' of size k. Consider all the biclique parts in the partition containing x_1. Since $N(x_1) = N(x_2)$, we can add x_2 to all these bicliques to obtain new bicliques. The resultant graph is G, Thus, (G, k) is a YES instance.

We now justify that these operations can be performed in polynomial time. WLOG, assume that $m \leq n$. We apply brute-force to obtain twins. It takes $\binom{n}{2}$ operations to check for twins Y. This can create more twins in X which can be checked in $\binom{m}{2}$ time and then we again check for twins in Y. We do this recursively until there are no twins left. Complexity wise worst case scenario is every time a partition is reduced, we create more twins in the other partition. But after $n + m$ iterations, we exhaust the graph, so we get a $O(n^3)$ running complexity. □

Lemma 4.2. Red. 2 is safe and can be performed in polynomial time.

Proof. Consider an isolated vertex v in a graph G. Since there are no edges incident on v, it does not participate in biclique partitions of G. Thus, given a biclique partition of G of size k, one can remove v to obtain G' and the resultant is a biclique partition of G'. Similar arguments hold for reverse direction.

One can perform a liner check to identify isolated vertices. □

Now we describe how these reduction rules lead to the aforementioned kernel size. We will exploit the following lemma in this regard.

Lemma 4.3. Consider $G = (X, Y, E)$ to be fully reduced i.e, Red. 1 and Red.

2 cannot be further applied. If $|X|$ or $|Y|$ is greater than 2^k then (G, k) is a NO instance.

Proof. Let G be a fully reduced instance. Suppose (G, k) is a YES instance. We have to prove that $|X| \leq 2^k$ and $|Y| \leq 2^k$. We prove by contradiction. Suppose $|X| > 2^k$. Since G is a YES instance, there exists a partition $\mathcal{P}$ of G of size k. We associate with every vertex an incidence vector, a k-tuple defined as follows: $\forall v \in G$, $I_v = (1_{v \in \mathcal{P}_i} : \forall i \in [k])$. Here, $1_{x \in A}$ is the indicator function over an arbitrary set A. We know there are at most 2^k distinct binary k-tuples. Since $|X| > 2^k$, there exists $x_1, x_2 \in X$ such that $I_{x_1} = I_{x_2}$. If I_{x_1} is a zero vector, then x_1 is an isolated vertex and Red. 2 is applicable which is a contradiction. Otherwise, both x_1 and x_2 participate in the same biclique which implies $N(x_1) = N(x_2)$. Thus, Red. 1 is applicable which is again a contradiction. Hence, $|X| \leq 2^k$. Following similar arguments, we prove that $|Y| \leq 2^k$. □

Given any arbitrary instance, we apply Red. 1 and Red. 2 exhaustively. If the obtained instance has more than 2^k vertices, we conclude it is a NO instance. Otherwise, we obtain an equivalent instance whose size is bounded by 2^k. Thus, we obtain exponential kernel.

We now present couple more reduction rules. Though these reduction rules do not lead to a better kernel size, they can still be applied in practice.

Red. 3: If there exists a vertex v of degree 1, remove the vertex $N(v)$ and decrease the parameter by 1.

The next reduction rule was presented in [1] and is only valid for biclique cover problem.

Red. 4: Suppose G is irreducible. If there exists $x \in X$ such that $N(x) = Y$, then remove x without changing the parameter. Analogous result hold for Y.

We present the safeness of these reduction rules.

Lemma 4.4. Red. 3 is safe and can be performed in polynomial time.

Proof. Let v be a vertex of degree 1. Suppose $(G-N(v), k-1)$ is a YES instance. Then there exists a biclique partition of G of size $(k-1)$. Add the star biclique rooted at $N(v)$ to this to obtain graph G. Thus, (G, k) is a YES instance. We now prove the backward direction.

Now suppose (G, k) is a YES instance. Then there exists a biclique partition of G of size k. Consider the part P in the partition containing v. Since v is of degree 1, P is a star. We remove $N(v)$ from all the parts excluding P. As explained in the proof of Red. 1, the resultant parts are also biclique. We modify P to the star biclique rooted at $N(v)$. This is also a biclique partition of G of size k. Removing P from this results in graph $G - N(v)$ and it's biclique partition of size $(k-1)$. Thus, $(G - N(v), k-1)$ is a YES instance.

One can perform a liner check to identify vertices of degree 1. □

Lemma 4.5. Red. 4 is safe and can be performed in polynomial time.

Proof. Suppose there exists $x \in X$ such that $N(x) = Y$. Consider (G, k) to be a YES instance. As explained in the proof of Red. 1, $(G - \{x\}, k)$ is also YES instance.

Suppose $(G - \{x\}, k)$ is a YES instance. Then there exists a biclique cover of size k. Add x to all the parts in the biclique cover. The resultant parts are also bicliques since $N(x) = Y$. To prove that the resultant is also a biclique cover, we have to make sure that all the edges incident on x are covered. Suppose not. That means there exists y in Y such that xy is not covered. By construction, this would mean that y does not belong to any part in the biclique cover of $G - \{x\}$. This is a contradiction as $N_{G-\{x\}}(y) \geq 1$. □

4.2 Matrix Rank and Biclique Partition Problem

The entire graph theoretic notion of biclique cover/partition problem can be studied equivalently in terms of matrices. We now proceed to prove $bp(G) = b(A)$ and $bc(G) = bool(A)$.

Theorem 4.6. $bp(G) = b(A)$

Proof. $b(A) = \min_k\{\exists B \in \{0,1\}^{m\times k}, C \in \{0,1\}^{k\times n} \text{ s.t } A = B \cdot C\}$

$$A = B \cdot C = \sum_{i=1}^{k} B^i \cdot C_j \tag{4.1}$$

Here, X^i represents ith column of X and X_i represents ith row of X. Note that a biclique is a rank one matrix. Thus, by (4.1), we get $b(A)$ exactly represents minimizing k such that A is sum of exactly k bicliques. Hence, $bp(G) = b(A)$. □

Theorem 4.7. $bc(G) = bool(A)$

Proof. Similar to the proof of Theorem 4.6. □

In this thesis, we use both the notions inter-changeably according to context. As a slight abuse of notion, we also write $bp(A)$ to imply $b(A)$.
As a simple corollary of Theorem 4.6, we obtain an inequality for $bp(G)$. This inequality is special in the sense that this is the first result which is true only for biclique partition problem, not for cover problem.

Corollary 4.8. $rank(A) \leq bp(G)$

Proof. This is a straightforward observation of Theorem 4.6. Since $rank(A) \leq b(A) = bp(G)$, we get $rank(A) \leq bp(G)$.
Note that $rank(A)$ is minimum k such that $A = B \cdot C$ where B is a $m \times k$ real

matrix and C is a $k \times n$ real matrix. On the other hand, $b(A)$ is minimum k such that $A = B \cdot C$ where B is a $m \times k$ binary matrix and C is a $k \times n$ binary matrix. Since every binary matrix is a real matrix, we get $rank(A) \leq b(A)$. □

4.3 FPT algorithms

In this section, we present the best known FPT algorithms on Biclique Cover and Partition problem. While the former is a result of Brute force on the exponential kernel, the latter makes use of the binary rank relationship.

4.3.1 Biclique Cover

As stated before, the following algorithm presented in [1] is a Brute force algorithm. This will be applied on the exponential kernel to obtain a $2^{2^{2k} \log k + 3k}$ FPT running time.

Algorithm 1: FPT algorithm for Biclique Cover

Input : A bipartite graph $G = (X, Y, E)$ and a positive integer k with $|X|, |Y| \leq 2^k$

Output: A biclique cover of G of size k

```
1  foreach Partition {E_1, E_2, ..., E_k} of E do
2      foreach i ∈ [k] do
3          if G[E_i] is not a biclique then
4              check if edges from G \ E_i can be added to make G[E_i] a biclique
5              if the above is not possible then
6                  Break
7              end
8              else
9                  Record the biclique formed in E_i
10             end
11         end
12     end
13     Return YES instance with {E_1, E_2, ..., E_k}
14 end
15 Return NO instance
```

We now do the running time analysis of Algorithm 1 below:

Lemma 4.9. Algorithm 1 runs in time $O^*(\frac{2^{2^{2k}\log k + 2k + \log k}}{k!})$.

Proof. In Algorithm 1, we run over all partitions of the edge set of size k and check if each part can be made a biclique by appending edges in it. For appending edges, we check all the edges of the graph bounded by $|V(G)|^2 = 2^{2k}$. Since this has to be done over all the parts in the partition, running time for one particular partition is $k \cdot 2^{2k} = 2^{2k + \log k}$. Total number of non-empty partitions into k parts

of a set F is called Sterling number of second kind, denoted by $S(F,k)$. This has been shown to be asymptotically equal to $\frac{k^{|F|}}{k!}$. Thus, total running time comes out to be $S(E(G),k)\cdot 2^{2k+\log k} = S(2^{2k},k)\cdot 2^{2k+\log k} = \frac{2^{2^{2k}\log k+2k+\log k}}{k!}$. □

A natural follow-up question is that can the kernel size and FPT running time be bettered. This is answered in the following stated results from [13]

Theorem 4.10. *There exists a polynomial time reduction stated as follows: given a 3-SAT ψ instance on n variables and m clauses, produces a bipartite graph G with $|X|+|Y| = O(n+m)$ such that there exists a positive integer $k = O(\log n)$ for which G has a biclique cover of size at most k if and only if ψ is satisfiable.*

Corollary 4.11. Biclique Cover cannot be solved in $2^{2^{o(k)}}$ unless Exponential Time Hypothesis fails.

Corollary 4.12. There exists a $\delta > 0$ such that there is no polynomial time algorithm that produces a kernel for Biclique Cover of size $2^{\delta k}$, unless $P = NP$.

One can refer to [13] for proofs of the above. We note that the gap between the lower bound and upper bound of running times of Biclique Cover is almost closed. Thus, doing better than the simple exponential kernel and brute force FPT algorithm is not possible (under appropriate assumptions).
We now move on to Biclique Partition problem and see if similar results are true.

4.3.2 Biclique Partition

Unlike the doubly exponential lower bound on FPT running time of Biclique Cover problem, we have a FPT algorithm of running time $O^*(2^{3k^2})$ [13]. We present this algorithm below. This algorithm makes use of the binary rank equivalency of the biclique partition problem.

Algorithm 2: FPT algorithm for Biclique Partition

Input : A binary matrix A of size $m \times n$ and a positive integer k with $m, n \leq 2^k$

Output: Two binary matrices B and C of sizes $m \times k$ and $k \times n$ such that $A = B \cdot C$

```
1  foreach {i_1, i_2, ..., i_k} ⊆ [m] and B' ∈ {0,1}^{k×k} do
2      Permute {i_1, i_2, ..., i_k} rows of A to be the first rows of A
3      A' := first k rows of A foreach  j ∈ [n] and C_j ∈ {0,1}^{k×1} do
4          if jth column of A' ≠ B' · C_j then
5              break
6          end
7      end
8      if C is found then
9          Guess the remaining columns of B
10         Return B and C
11     end
12 end
```

We do the running time analysis of Algorithm 2 below:

Lemma 4.13. Algorithm 2 runs in time $O^*(2^{2k^2+k})$

Proof. Guessing k rows of A to be permuted to become the first k rows take $\binom{m}{k}$ time. Guessing B' takes 2^{k^2} time. For each such pair of k rows and guess of B', finding C column-wise takes $2^k \cdot n$ time. Similarly, the remaining rows of B takes $2^k \cdot m$ time. Hence, we get a total running time of $O^*(\binom{m}{k} \cdot 2^{k^2} \cdot 2^k) = O^*(2^{2k^2+k})$ using $m \leq 2^k$. □

Unlike Biclique Cover problem, it is open to determine if the size of kernel of the Biclique Partition problem can be brought down. This thesis focuses on

utilizing tools from poset theory to answer this question which is the subject of the next two chapters.

We also implemented the above stated Algorithm 2 in C++. Appendix A presents a code snippet of the implementation. Below we present the running time matrix obtained for a fixed k = 3. The following has been performed on Intel(R) Core(TM)i3-4005U CPU (1.70 GHz) with 4 GB RAM. It ran Windows 10 and C++ codes were complied using gcc 13.12 with std c++11.

Random matrices of the given size were inputs and the results are average over multiple random sampling. 20 random samples were taken for all the entries. The following is the matrix obtained with standard error.

col/*row*	4	5	6	7	8
4	0.031 ± 0.009	0.048 ± 0.005	0.051 ± 0.002	0.066 ± 0.03	0.071 ± 0.004
5		0.051 ± 0.01	0.069 ± 0.003	0.140 ± 0.015	0.159 ± 0.020
6			0.091 ± 0.07	0.142 ± 0.09	0.232 ± 0.012
7				0.323 ± 0.052	0.412 ± 0.071
8					0.525 ± 0.08

Running time is controlled for k=3. For k = 4, a 6*6 matrix takes 13 sec, 7*7 takes 39.1 sec, 8*8 takes 187.4 sec. From here on, it starts shooting up.

Chapter 5

Partially Ordered Sets

Chapter 5 explores the relationship between partially ordered sets and biclique partition problem. We devote this chapter to introduce partially ordered sets. We follow [14] in this regard.

5.1 Definition

Definition 5.1. A partially ordered set or *poset* $(S, \preceq)$ is a set S, along with a binary relation '$\preceq$' which is transitive that is, if $x \preceq y$ and $y \preceq z$ then $x \preceq z$ and antisymmetric that is, if $x \preceq y$ and $y \preceq x$ then $x = y$.
Two elements x and y are called *comparable* if either $x \preceq y$ or $y \preceq x$.

Definition 5.2. A subset S' of a poset S is called a *chain* if any two distinct elements of S' are comparable.

Definition 5.3. Exactly opposite to the definition of chain, a subset S' of a poset S is called an *antichain* if any two distinct elements of S' are incomparable, that is $\forall x, y \in S'$, neither $x \preceq y$ nor $y \preceq x$.

5.2 Examples

There are several examples of Posets that are encountered in applications. We present some of them below.

- Set inclusion: a family of sets can be partially ordered by taking set-inclusion as a binary relation.
- Integer divisibility: a set of positive integers can be partially ordered by integer divisibility i.e, '$x \preceq y$' iff x divides y in whole parts.
- Reachability in DAG: consider a directed acyclic graph. Then '$x \preceq y$' iff vertex x is reachable from vertex y.
- Vector comparison : a set of vectors of fixed dimension can be partially ordered by vector comparison i.e, '$x \preceq y$' iff $x \leq y$.

It is a trivial exercise to check the above are posets. A pictorial way to realize posets is through *Hasse diagrams*: elements of the set correspond to vertices on a plane arranged and connected in such a fashion that if $x \preceq y$, then there exists an edge xy and x lies below y.
For instance, take $\mathcal{F} = \{\{1\}, \{2\}, \{1,2\}, \{2,3\}, \{3,4\}, \{1,2,3,4\}\}$. We know that $(\mathcal{F}, \subseteq)$ is a poset. Below, we present it's hasse diagram.

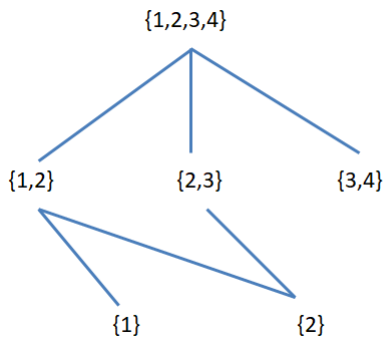

Figure 5.1: Hasse diagram of $\mathcal{F}$

We present examples of chain and antichains in $\mathcal{F}$:

- Let $C_1 = \{\{1,2\},\{1,2,3,4\}\}$, $C_2 = \{\{3,4\},\{1,2,3,4\}\}$. Both C_1 and C_2 are chains in $\mathcal{F}$.
- We call a chain C *maximal chain* if $\nexists$ any element $x \in \mathcal{F}$ such that $x \cup C$ is a chain. For example, C_1 is not a maximal chain as $\{\{1\} \cup C_1\}$ is also a chain. On the other hand, C_2 is a maximal chain as no other element in $\mathcal{F}$ can be added in C_2 to make a bigger chain.
- Similarly, one can define the *maximal antichain*.
- {{3,4},{2,3},{1,2}} is a maximal antichain in $\mathcal{F}$

Definition 5.4. Let X be a finite set. Let $(2^X, \leq)$ be a poset. Here, the binary relation is set inclusion again. Then, a antichains in this system is called a *Sperner family*.

For example, consider all family of all sets with a fixed size. It is a trivial exercise to check these are antichains. Thus, if $|X| = n$, we can obtain antichains of size $\binom{n}{\lfloor n/2 \rfloor}$. Are there larger antichains possible? This is answered in a well known theorem called *Sperner's theorem*.

Theorem 5.5. *Let $F \subseteq 2^X$ where $|X| = n$. If F is an antichain then $|F| \leq \binom{n}{\lfloor n/2 \rfloor}$.*

Remark. We note again that the above theroem is tight as explained before.

5.3 Decomposition theorems

Given a poset $\mathcal{P}$, decomposition of $\mathcal{P}$ deals with finding a partition of $\mathcal{P}$ into mutually disjoint chains or antichains. The lesser the partition size, the better. Consider the following lemma.

Lemma 5.6. Let C be a chain and A be an antichain in a poset $\mathcal{P}$. Then $|C \cap A| \leq 1$.

Proof. Suppose $|C \cap A| > 1$. Pick any two distinct elements x and y in $C \cap A$. Since x and y belong to both a chain and an antichain implies they are both comparable and incomparable. This is a contradiction. □

Thus, consider any chain of size q. Then the poset cannot be partitioned into fewer than q antichains. Is this optimal ? This is answered by the following decomposition theorem.

Theorem 5.7. *Let the largest chain in a poset S be of size q. Then S can be partitioned into q antichains.*

Proof. Let A_i denote the set of all elements x in S such that the longest chain ending at x is of size i. Since the largest chain in S is of size q, $A_i = \varnothing$ for $i > q$. Thus, we get a disjoint union of S as $S = A_1 \cup A_2 \cup ... \cup A_q$. We now prove that any arbitrary A_i is an antichain and then we are done. We prove this by contradiction. Suppose there exists two distinct elements x and y in A_i such that $x \prec y$. Let C_x be the longest chain ending at x of length i. Then, $C_x \cup y$ is a chain of length $i + 1$ ending at y. This implies $y \notin A_i$ which is a contradiction. □

We note that the duality of the arguments presented. Does the same result hold for partition into mutually disjoint chains? This was answered affirmative by Dilworth in 1950 in the famous *Dilworth's Decomposition theorem.*

Theorem 5.8. *(Dilworth's theorem) Let the largest antichain in a poset S be of size q. Then, S can be partitioned into q chains.*

Proof. We prove this by inducting on cardinality of S. Let a be a maximal element in S. For $|S| = 1$. the statement is true trivially. Consider $S' = S \setminus a$. We assume that $S' = C_1 \cup C_2 \cup ... \cup C_q$ where q is the size of largest antichain in

S'. We now prove that that S either contains an antichain of size q+1 or can be written as disjoint union of q chains.

Note that any antichain in S' of size q contains one element from each C_i. Let x_i be the maximal element in C_i that belongs to some q element antichain in S'. We now prove that $A = \{x_1, x_2, ..., x_q\}$ forms an antichain. To prove by contradiction, we suppose there exists two distinct elements in A such that $x_i \prec x_j$. Consider the q element antichain of which x_j is a part of. We know that this antichain must contain an element of C_i, say x_i'. Then if $x_i' \prec x_i$, then by transitivity we get $x_i' \prec x_j$ which is a contradiction. $x_i \not\prec x_i'$ because of x_i is a maximal element which belongs to a q sized antichain. This implies $x_i = x_i'$ which is again a contradiction because $x_i \prec x_j$ and hence, cannot be in an antichain. This proves A is an antichain.

Consider $A \cup a$. Suppose this is an antichain. Then we obatin an antichain of size $(q + 1)$ and we are done. Otherwise, there exists $x_i \prec a$. Consider $C' = \{x : x \preceq x_i\} \cup \{a\}$. Then in $S' \setminus C'$, there is not antichain of size q because x_i was the maximal element participating in such an antichain. Thus $S' \setminus C'$ can be partitioned into $(q - 1)$ chains. Add C' to this and we are done. □

To realize the power of this theorem, we state the following remark.

Remark. Marriage theorem is a special case of Dilworth's theorem.

Chapter 6

Biclique Partition Problem and Posets

Summary

In this chapter, we explore how poset theory can be used to obtain better kernel for biclique partition problem. This approach has not been explored before. We obtain some results in this direction and culminate it into a theorem at the end. This will lead to a new technique in identifying smaller kernels. Nevertheless, this approach has it's own limitations. We prove that identifying polynomial kernel will not be possible through this technique.

6.1 Notations

Consider $G = (X, Y, E)$ to be a bipartite graph where $|X| = m$ and $|Y| = n$. Let A represents it's adjacency matrix. Suppose (G, k) is a Yes instance and irreducible.

Then, there exists two binary matrices B and C of size $m \times k$ and $k \times n$ such

that $A = B \cdot C$.

Note that the rows of A are in one-to-one correspondence with rows of B. In other words, every row of A is associated with a binary k-tuple which is precisely the corresponding row in B. Similarly, the columns of A are in one-to-one correspondence with columns of C.

Consider the rows of the adjacency matrix A as a poset, $(A_r, \preceq)$. That is, A_r comprises of the row vectors as elements and the ordering relation '$\preceq$' is set-inclusion. Elaborately, if r_1 and r_2 are two rows of A, '$r_1 \preceq r_2$' iff $supp(r_1) \subseteq supp(r_2)$. Similarly, define poset A_c consisting of columns of A, B_r consisting of rows of B and C_c consisting of columns of C.

6.2 New Results

Result 6.1. Let $S' \subseteq A_r$ be a chain in $(A_r, \preceq)$. Let $G' = G[S' \cup Y]$. Then, bp(G')=$|S'|$. Similar statement is true for $S' \subseteq A_c$.

Proof. Let $S' = \{r_1, r_2, ..., r_{|S'|}\}$ such that $r_1 \leq r_2 \leq ... \leq r_{|S'|}$. Since we assume that (G, k) is irreducible, $r_i \neq r_{i+1} \forall i \in [|S'| - 1]$. This implies $r_1 < r_2 < ... < r_{|S'|}$. Define r_0 as zero vector. Now, we construct a set F as follows: pick any $e \in r_i \setminus r_{i-1} \forall i \in [|S'|]$. Then, F is a fooling set. This is because for any two elements r_{ij}, r_{kl} in F, $r_{il} = 0$. Using Theorem 3.11 we get that $|S'| \leq bp(G')$. By taking the star biclique partition, we know that $bp(G') \leq \min\{|S'|, |Y|\} \leq |S'|$. Implies $bp(G') \leq |S'|$. Thus, we get $bp(G') = |S'|$. One can analogously prove the same result A_c. □

Result 6.2. Let $r_1, r_2 \in B_r$ be comparable. Then, the rows corresponding to r_1 and r_2 in A are also comparable in A_r. Similar statement is true for columns of A.

Proof. Let $r_1, r_2 \in B_r$ be comparable, say $r_1 \leq r_2$. Then, the corresponding rows in A are $r_1 \cdot C$ and $r_2 \cdot C$. We have to prove that $r_1 \cdot C \leq r_2 \cdot C$.
This is true trivially if $C \geq 0$. Since C is a binary matrix, we are done. □

Corollary 6.3. Let $S' \subseteq A_r$ be an antichain in A_r. Then, the rows corresponding to S' in B form an antichain in B_r.

Proof. Suppose not. This implies there exists b_1 and b_2 in B_r such that $b_1 \leq b_2$. Using Result 6.2, this would imply the corresponding rows in A are also comparable. This is a contradiction as we started with an antichain. □

Corollary 6.4. Let $S' \subseteq C_c$ be a chain in C_c. Then, the corresponding columns in A form a chain in C_c.

Proof. We proceed similar to the proof of Corollary 6.3. Suppose not. This implies there exists c_1 and c_2 in columns of A such that $c_1 \not\leq c_2$. Using Result 6.2, this would imply the corresponding columns in C are also incomparable. This is a contradiction as we started with an chain. □

We now prove a new theorem which will help us in building a better technique to kernalize Biclique Partition problem. Below we present some definitions in this regard.

- $\{0,1\}^{p\times q}$:= collection of all $p \times q$ binary matrices
- $\mathcal{B}_{m,k}$:= $\{B \in \{0,1\}^{m\times k} |$ $\nexists$ a zero column in B and the rows of B form an antichain$\}$
- $\mathcal{C}_{k,n}$:= $\{C \in \{0,1\}^{k\times n} |$ $\nexists$ a zero row in C and the columns of C form an antichain$\}$

We define function $F(k)$ as below:
$F(k) := \max\{\min(m,n) | \quad B \in \mathcal{B}_{m,k}, C \in \mathcal{C}_{k,n}$ s.t. $B \cdot C \in \{0,1\}^{m\times n}$ $\}$

One intuition behind defining $F(k)$ in such a way is the following: Since all the YES instances admit the factorization $A = B \cdot C$ where A, B and C are all binary matrices, both B and C need to attain a 'restricted structure' in order to achieve dot product binary. An easy way to to have this would be to use vertical zero blocks in B and horizontal zero blocks in C. Thus, we impose that restraint. Now adding the antichain restriction tightens the structure more as large antichains will exhaust lot more possibilities of dot product being binary.
Below we present an interesting result on $F(k)$ that shows that the conditions in $\mathcal{B}_{m,k}$ of requiring non zero columns can be dropped without any change in the value of $F(k)$. To prove this we define the following:

- $\mathcal{B}'_{m,k} := \{B \in \{0,1\}^{m \times k} |$ the rows of B form an antichain$\}$
- $\mathcal{C}'_{k,n} := \{C \in \{0,1\}^{k \times n} |$ the columns of C form an antichain$\}$

We define function $F'(k)$ as below:
$F'(k) := \max\{\min(m,n) | \quad B' \in \mathcal{B}'_{m,k},\, C' \in \mathcal{C}'_{k,n}$ s.t. $B' \cdot C' \in \{0,1\}^{m \times n}\}$

Result 6.5. $F(k) = F'(k)$

Proof. By definition, we know that $\mathcal{B}_{m,k} \subseteq \mathcal{B}'_{m,k}$ and $\mathcal{C}_{k,n} \subseteq \mathcal{C}'_{k,n}$. Thus,

$$F(k) \leq F'(k) \tag{6.1}$$

We now prove the other direction. Consider $B' \in \mathcal{B}'_{m,k}$, $C' \in \mathcal{C}'_{k,n}$ such that $\min(|B'|, |C'|) = F'(k)$. Corresponding to B' and C', we construct $B_{B'} \in \mathcal{B}_{m,k}$ and $C_{C'} \in \mathcal{C}_{k,n}$ such that $B_{B'} \cdot C_{C'}$ is binary. We give the construction later and

proceed with the proof.

$$\begin{aligned} F'(k) &:= \min(|B'|, |C'|) \\ &= \min(|B_{B'}|, |C_{C'}|) \\ &\leq \max\{min(|B|, |C|) | B \in \mathcal{B}_{m,k}, C \in \mathcal{C}_{k,n} s.t. B \cdot C \text{ is binary}\} \\ &= F(k) \end{aligned} \tag{6.2}$$

Thus,

$$F'(k) \leq F(k) \tag{6.3}$$

Using Eq. (6.1) and (6.3), we obtain $F(k) = F'(k)$. We now give the construction of $B_{B'}$ and $C_{C'}$ as mentioned before. Suppose there does not exist a zero column in B' and zero row in C'. Then, take $B_{B'} = B'$, $C_{C'} = C'$ and we are done. Now assume there exists a zero column in B'. WLOG, assume that the first column is 0. Then, construct $B_{B'}$ as below: Let

$$B' = \begin{bmatrix} 0 & — & b_1 & — \\ 0 & — & b_2 & — \\ \vdots & & \vdots & \\ 0 & — & b_m & — \end{bmatrix}$$

Then,

$$B = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & — & b_2 & — \\ \vdots & & \vdots & \\ 0 & — & b_m & — \end{bmatrix}$$

We now prove that $k \leq F'(k)$. Consider $[I_k|0]^T \in \mathcal{B'}_{m,k}$ and $[I_k|0] \in \mathcal{C'}_{k,n}$. Since the dot product of these matrices is binary, we get $k \leq F'(k)$. Thus, we can assume $k \leq m$. We do the above construction for all zero columns in B'

recursively. Since there can be at most k many 0 columns and $k \leq m$, this is plausible.

We now prove that the rows of B form an antichain. Since the rows of B' were an antichain which implies $B' \setminus \{b_1\}$ is also an antichain. Let the the first row of B be r_1. We have to prove that $(B' \setminus \{b_1\}) \cup \{r_1\}$ is an antichain. Suppose not. Then, $\exists\ 2 \leq i \leq m$ such that either $b_i \leq r_1$ or $r_1 \leq b_i$. Since $r_{11} = 1$ and $b_{i1} = 0, r_1 \not\leq b_i$. This implies $b_i \leq r_1$. But $r_{1j} = 0\ \forall\ 2 \leq j \leq k$ which implies $b_{ij} = 0\ \forall\ 2 \leq j \leq k$. Since $b_{i1} = 0$ by construction, this means b_i is a zero vector which is a contradiction because $B' \setminus \{b_1\}$ is an antichain. Thus, the rows of B are an antichain.

We can analogously construct C from C' and prove the same line of results. We now explain that $B \cdot C$ is a binary matrix and then we are done. Since $B' \cdot C'$ is binary and we are replacing the rows of B' and columns of C' with unit vectors, the dot product still remains binary. Hence, $B \cdot C$ is a binary matrix . □

Thus, we work with $F(k)$ in computations as the search space is smaller. We now proceed to prove the main theorem on $F(k)$ and it's proof. In the following section, we see how this might help in finding a better kernel.

Theorem 6.6. *Suppose (G, k) is a Yes instance. Then, min(m, n)$\leq k \cdot F(k)$.*

Proof. Suppose (G, k) is a Yes instance. Then, using Result 6.1, we know that the largest chain in A_r and A_c is of size k. Then using Theorem 5.7, we know that A_r can be partitioned into at most k antichains. Note that $|A_r| = |X| = m$. Pick the largest antichain among this partition and call it A'. Using Corollary 6.3, we know that the rows corresponding to A' in B forms an antichain. Call it B'. Thus, using Theorem 5.7, we conclude that

$$m \leq k \cdot |A'| = k \cdot |B'| \tag{6.4}$$

Note that $B' \cdot C$ is a binary matrix. Consider a chain in C_c, say C''. Using Corollary 6.4, we know that the corresponding columns in A is also a chain, say A''. Using Result 6.1 again, we know that $|A''| \leq k$. As $|A''| = |C''|$ due to the one-one correspondence implies $|C''| \leq k$. As C'' was any arbitrary chain in C_c, size of the largest chain in C_c is at most k. Then using Theorem 5.7 again, C_c can be partitioned into at most k antichains. Pick the largest antichain out of this partition and call it C'. Therefore, (using Theorem 5.7 again) we conclude that

$$n \leq k \cdot |C'| \tag{6.5}$$

Again by construction, $B' \cdot C'$ is a binary matrix. Also, rows of B' form an antichain, so does the columns of C'. Thus $\min(|B'|,|C'|) \leq F'(k)$. Using Result 6.5, we get $\min(|B'|,|C'|) \leq F(k)$. This implies $\min(k.|B'|,k.|C'|) = k \cdot \min(|B'|,|C'|)$ $\leq k \cdot F(k)$. Using (6.4) and (6.5), we know that $\min(m,n) \leq \min(k.|B'|,k.|C'|) \leq$ $k \cdot F(k)$. Thus, $\min(m,n) \leq k \cdot F(k)$. □

6.3 Kernel through posets

6.3.1 Technique

We now address how Theorem 6.6 can be used to obtain kernels. We note that Theorem 6.6 talks about $min(m,n)$ but in order to obtain a kernel, one should bound $max(m,n)$. Our next result talks about this issue.

Result 6.7. For Biclique Partition problem, bounding $min(m,n)$ by a subexponential function will lead to a better FPT running time of Algo 2.

Explanation. We observe the running time analysis of Algo 2. Without loss of generality, we assume that $m \leq n$. The running time depends polynomially in n and hence, does not contribute in the O^* notation. Thus, if we bound

$min(m, n) = m$ by a better function, the O^* running time would drop.

So, as a slight abuse of notation, we refer to $min(m, n)$ as size of the instance. Suppose one can bound $F(k)$ by a subexponential function, say $g(k)$. If $min(m, n) > k \cdot g(k)$, then it is a NO instance. Otherwise, size of the instance is bounded by $k \cdot g(k)$.

We also have computationally evaluated the some values of F(k). We present them in the table below.

k	$F(k)$
3	3
4	7
6	8
7	11
8	15
9	21

We now compare the previous kernel size and the new kernel size obtained for these values.

k	2^k	$k \cdot F(k)$
3	8	9
4	16	28
6	64	48
7	128	77
8	256	120
9	512	189

We observe that through the poset technique, we obtain better kernel sizes for $k = 6, 7, 8, 9$. Also, we observe that the growth of the new kernel is quite controlled

as compared to the known exponential kernel. This gives us a incentive to look for a suitable bound for $F(k)$.

6.3.2 Limitations

In this section, we talk about the limitations of the kernalization technique described in the previous section. As described before, bounding $F(k)$ by a suitable function can lead to a better kernel. We now prove that this bounding function is not polynomial. Thus, through this technique, we will not be able to conclude whether there exists a polynomial kernel or not.

To prove this, we construct two 'large' matrices B and C such that $B \in \mathcal{B}_{m,k}$, $C \in \mathcal{C}_{k,n}$ and $B \cdot C$ is a binary matrix.

As $F(k) := \max \{\min(m,n) | \quad B \in \mathcal{B}_{m,k}, C \in \mathcal{C}_{k,n} \text{ s.t. } B \cdot C \in \{0,1\}^{m \times n} \}$, we obtain a large instance in a set over which we are maximizing size.

We now present these matrices in the following pages. Fix $i \in [k]$.

$$B = \begin{pmatrix} \overbrace{}^{i} & \overbrace{}^{(k-i)} \\ 1\ 1\ 1\ 0\ 0\ \cdots\ 0\ 0\ 0 & 1\ 0\ 0\ \cdots\ 0\ 0 \\ 1\ 1\ 0\ 1\ 0\ \cdots\ 0\ 0\ 0 & 0\ 1\ 0\ \cdots\ 0\ 0 \\ 1\ 0\ 1\ 1\ 0\ \cdots\ 0\ 0\ 0 & \vdots\ \ \ddots\ \ \vdots \\ 0\ 1\ 1\ 1\ 0\ \cdots\ 0\ 0\ 0 & 0\ 0\ 0\ \cdots\ 0\ 1 \\ \ \ \vdots\ \vdots\ \vdots & \vdots\ \vdots\ \vdots \\ \ \ \vdots\ \vdots\ \vdots & \vdots\ \vdots\ \vdots \\ \ \ \vdots\ \vdots\ \vdots & \vdots\ \vdots\ \vdots \\ \ \ \vdots\ \vdots\ \vdots & \vdots\ \vdots\ \vdots \\ 0\ 0\ 0\ 0\ 1\ \cdots\ 0\ 1\ 1 & 1\ 0\ 0\ \cdots\ 0\ 0 \\ 0\ 0\ 0\ 0\ 1\ \cdots\ 1\ 0\ 1 & 0\ 1\ 0\ \cdots\ 0\ 0 \\ 0\ 0\ 0\ 0\ 1\ \cdots\ 1\ 1\ 0 & \vdots\ \ \ddots\ \ \vdots \\ 0\ 0\ 0\ 0\ 0\ \cdots\ 1\ 1\ 1 & 0\ 0\ 0\ \cdots\ 0\ 1 \end{pmatrix}$$

Construction of B is as follows:

- Consider submatrix formed by the first i columns of B. The rows of this submatrix consist of Sperner family of size $\binom{i}{\lfloor i/2 \rfloor}$.
- Consider the submatrix formed by the last $(k-i)$ columns of B. It consists of repeating blocks of I stacked on top of each other until we exhaust $\binom{i}{\lfloor i/2 \rfloor}$ rows. We note that the bottom most block might just be the first few rows of I.

$$C = \begin{pmatrix} \overbrace{}^{i} & \overbrace{}^{(k-i)} \\ 1\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 & 0\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 & 0\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 \\ 0\ 0\ 1\ 0\ 0\ \cdots\ 0\ 0\ 0 & 0\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 \\ \vdots\ \ \vdots\ \ \ddots & \vdots\ \ \vdots\ \ \ddots\ \ \vdots \\ 0\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 1 & 0\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 & 1\ 1\ 1\ 0\ 0\ \cdots\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 & 1\ 1\ 0\ 1\ 0\ \cdots\ 0\ 0\ 0 \\ \ \ \vdots\ \ \vdots\ \ \vdots & \ \ \vdots\ \ \vdots\ \ \vdots \\ \ \ \vdots\ \ \vdots\ \ \vdots & \ \ \vdots\ \ \vdots\ \ \vdots \\ \ \ \vdots\ \ \vdots\ \ \vdots & \ \ \vdots\ \ \vdots\ \ \vdots \\ 0\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 & 0\ 0\ 0\ 0\ 1\ \cdots\ 1\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ \cdots\ 0\ 0\ 0 & 0\ 0\ 0\ 0\ 0\ \cdots\ 1\ 1\ 1 \end{pmatrix}$$

Construction of C is as follows:

- Consider submatrix formed by the first i columns of C. The first i rows of this submatrix is I. This is followed by $\binom{(k-i)}{\lfloor (k-i)/2 \rfloor}$ many zero vectors.
- Consider the submatrix formed by the last $(k-i)$ columns of C. The first i rows of this submatrix is a zero block. The next $\binom{(k-i)}{\lfloor (k-i)/2 \rfloor}$ rows is the sperner family of this size.

We note that $B \in \mathcal{B}_{m,k}$ and $C^T \in \mathcal{C}_{k,n}$.

Lemma 6.8. $B \cdot C$ is a binary matrix.

Proof. Consider any arbitrary row r in B and row c in C. We prove that $r \cdot c^T$ is binary. Suppose $r \cdot c^T = 0$. Then we are done.
Otherwise, $\exists j \in [k]$ such that $r_j = c_j = 1$. We consider two cases. Suppose $j \leq i$. Then, by construction of C, we know that this is possible only in the first i rows of C. But the first i rows are unit vectors which implies $r \cdot c$ is binary. Now, suppose $i < j \leq k$. Again by construction of C, we know that this is only possible in the rows after the first i rows. Then,

$$\begin{aligned} r \cdot c &= \sum_{l=1}^{k} r_l \cdot c_l \\ &= \sum_{p=1}^{i} r_p \cdot c_p + \sum_{q=i+1}^{k} r_q \cdot c_q \\ &= \sum_{p=1}^{i} r_p \cdot 0 + \sum_{q=i+1}^{k} r_q \cdot c_q \\ &= \sum_{q=i+1}^{k} r_q \cdot c_q \\ &= \sum_{q=i+1, q\neq j}^{k} r_q \cdot c_q + r_j \cdot c_j \\ &= \sum_{q=i+1, q\neq j}^{k} 0 \cdot c_q + 1 \cdot 1 \\ &= 1 \end{aligned} \tag{6.6}$$

Thus, $B \cdot C$ is a binary matrix. □

Hence, $F(k)$ is lower bounded by $min(|B|, |C|)$.

$$\begin{aligned} |B| &= \binom{i}{i/2} \\ |C| &= i + \binom{(k-i)}{(k-i)/2} \end{aligned} \tag{6.7}$$

Thus, we get

$$\max_{i \in [k]} \left\{ \min \{ \binom{i}{i/2}, i + \binom{(k-i)}{(k-i)/2} \} \right\} \leq F(k) \tag{6.8}$$

Pick $i = k/2$ and we get an approximate lower bound of $F(k)$ using Stirling's approximation of binomial coefficient.

$$\frac{\sqrt{2}}{\sqrt{k}} \cdot 2^{k/2} \leq F(k) \tag{6.9}$$

Hence, we cannot get a polynomial bound for $F(k)$. Nevertheless, there is still room for improvement from the previous 2^k kernel as the gap could be closer to $\sqrt{2k}2^{k/2}$ than towards 2^k.

Chapter 7

Biclique Partition Problem restricted on graph classes

Below we briefly define some special graph classes and present if BPP and BCC can be solved in P on them or not. We conclude with a new result in this direction.

7.1 Definitions

Definition 7.1. A bipartite graph is *chordal bipartite* if each cycle of length at least 6 has a chord. A chord in a cycle is an edge, not part of the cycle, between two vertices of the cycle.

Definition 7.2. An edge xy in a bipartite graph is called *bisimplicial edge* if $N(x)$ and $N(y)$ induce a biclique.

Definition 7.3. A bipartite graph G is called *perfect elimination bipartite* graph if there exists a sequence of edges $\{e_1, e_2, ..., e_l\}$ such that e_i is bisimplicial in the graph $G \setminus S_i$ and $G \setminus S_l$ has no edges. Here, S_i is union of all endpoints of the edges $\{e_1, e_2, ...e_i\}$ and $S_0 = G$. The sequence is $\{e_1, e_2, ..., e_l\}$ is called *perfect edge elimination ordering.*

Definition 7.4. A sequence $\{e_1, e_2, ..., e_l\}$ is called *perfect edge without vertex elimination ordering* if e_i is bisimplicial in the graph $G \setminus S_i$ and l equals $|E|$. Here, S_i is union of the edges $\{e_1, e_2, ...e_i\}$ and $S_0 = G$.
An equivalent definition of chordal bipartite graph is as follows:
A bipartite graph is chordal bipartite if it admits a perfect edge without vertex elimination ordering.

Remark. One can easily construct a perfect edge elimination ordering from a perfect edge without vertex elimination ordering. This implies chordal bipartite graphs is a subclass of perfect elimination bipartite graph.

Definition 7.5. A bipartite graph is called a *convex bipartite* graph if there exists an permutation of the columns of the biadjacency matrix such that all the 1s appear consecutively in the rows of the matrix after the permutation or vice-versa.

Definition 7.6. A *strong ordering* in a bipartite graph G is an ordering of vertices in X and Y such that for any edge ab and $a'b'$ such that $a <_X a'$ and $b <_Y b'$, ab' and $a'b$ is an edge.

Definition 7.7. The following two definitions are equivalent:
1. A graph G is distance-hereditary if it is connected and every induced path is isometric; that is, if the distance function in every induced subgraph of G is the same as in G itself.
2. A graph G is distance-hereditary if it can be construct from a single vertex by the following operations:

- Adding a pendant vertex: A new vertex that is adjacent to precisely one existing vertex of the graph.
- Creating true twins: For an existing vertex x add a new vertex y with neighbours $N[x]$.

- Creating false twins: For an existing vertex x add a new vertex y with neighbours $N(x)$.

A graph which is both bipartite and distance hereditary is called *bipartite distance hereditary* graph.

7.2 Previous Work

Following the suite of parameterized complexity, results of Biclique Cover on special graph classes are more studied than that of Biclique Partition problem. We see in [15] that Biclique Cover problem is in P in bipartite permutation graph, bipartite distance hereditary graph and bipartite C_4 free graph. We also see in [15] that Biclique Cover problem is NP-complete on chordal bipartite graphs. In [16], we see a stronger result which proves that both biclique cover and partition number are equal on bipartite domino-free graph and can be computed in polynomial time.

[16] also states how Biclique Cover problem can be solved in P for convex bipartite graphs in with reference to a related problem in [17]. In [18], we see that Biclique partition problem is in P for convex bipartite graph. In the diagram below, we present these results in a concise form.

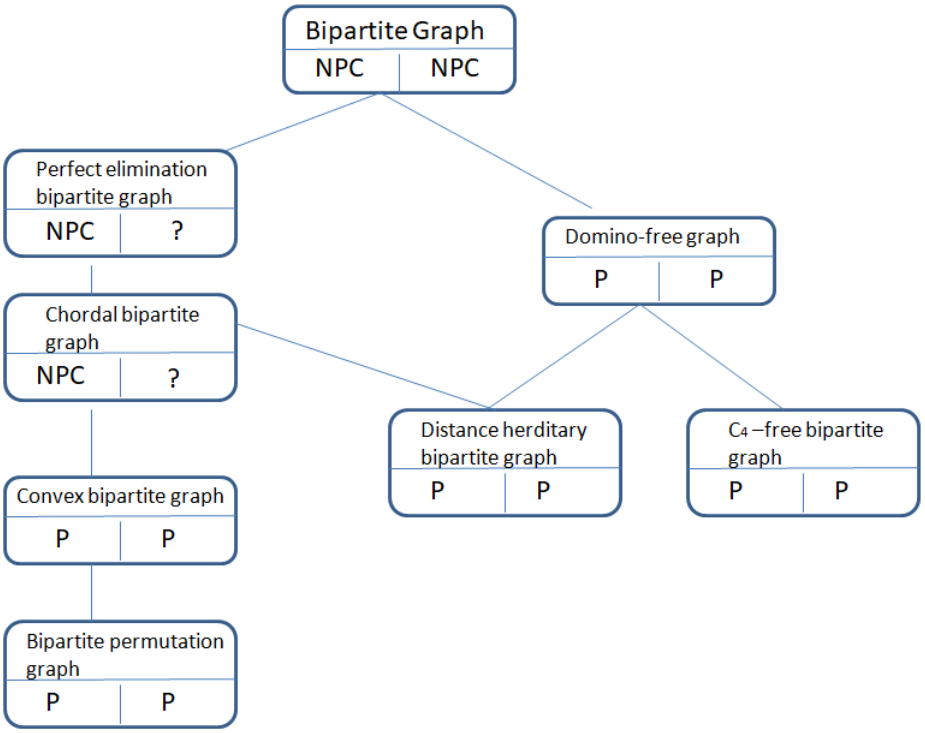

Figure 7.1: Biclique Cover and Biclique Partition on special graph classes: Left block represents Biclique Cover and the right block represents Biclique Partition

7.3 New results

7.3.1 Biclique Partition Problem on Perfect Elimination Bipartite graph

In this section, we prove that Biclique partition problem is NP-complete when restricted on perfect elimination bipartite graph. The proof idea is along the lines of the proof in [19] which proves Biclique Vertex Partition problem is NP-complete on perfect elimination bipartite graph.

Theorem 7.8. *There exists a polynomial-time reduction from Biclique Partition problem on general bipartite graph to Biclique Partition problem on Perfect elimination bipartite graph.*

Proof. Consider any arbitrary bipartite graph $G = (X, Y, E)$ and construct G' as stated: Make a copy of G. Now, consider the vertices in $X = \{X_1, X2, ..., X_m\}$. Corresponding to each vertex X_i, add 3 more vertices A_i, B_i and C_i such that X_i, A_i, B_i and C_i form a $K_{2,2}$ by adding edges $\{X_iA_i, X_iB_i, B_iC_i, C_iA_i\}$. The below diagram makes the construction more clear.

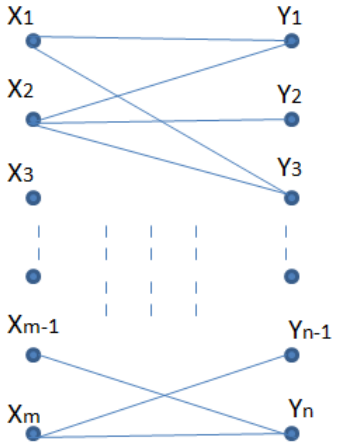

Figure 7.2: G

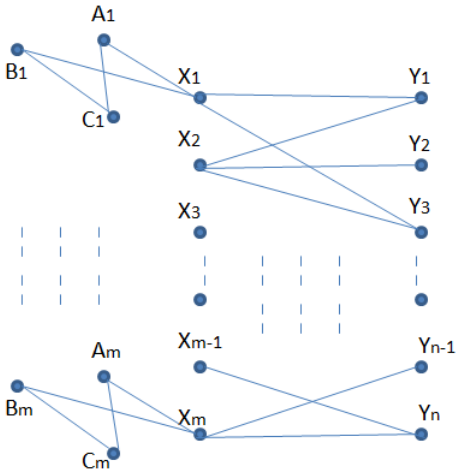

Figure 7.3: G'

We first prove that G' is a perfect elimination bipartite graph. This is evident by observing that $\{B_1C_1, B_2C_2, ..., B_mC_m, A_1X_1, A_2X_2, ..., A_mX_m\}$ is an edge elimination scheme. Also, G' is a bipartite graph with bipartition $(X \cup \{C_1, C_2, ..., C_m\}, Y \cup \{A_1, B_1, A_2, B_2, ..., A_m, B_m\})$.

We now prove that $bp(G') = bp(G) + m$.

This can be easily proved using the reduction rules described in Chapter 4. Note that $N(B_i) = N(A_i)$ for all i between 1 and m. Thus, using Reduction 1, we can delete a twin and the biclique partition number of the remaining graph remains the same. Remove B_i from the graph for all i between 1 and m. In the remaining graph, C_i becomes a vertex of degree 1. We use Reduction rule 3 and remove all A_i for i between 1 and m. Each removal causes the biclique partition number of the remaining graph to drop by 1. Since there are m such removals, we biclique number drops by m. The final remaining graph has all the C_i(s) as isolated vertices. We use Reduction rule 2 to remove these. This does not change the biclique partition number. The final remaining graph now is G. Thus, $bp(G) = bp(G') - m$ or $bp(G') = bp(G) + m$. □

Chapter 8

Conclusion and Future Work

We deal with two related problems, Biclique Cover and Partition problem. Biclique Cover problem is extensively studied in literature unlike its counter problem, Biclique Partition. It is known that both problems admit exponential kernel. While it is has been proven that the kernel size for the former cannot be bettered assuming ETH, we don't know any such bound for the later. In order to tackle this, we try to bring some structure into solution instances of Biclique Partition problem using poset theory. We prove that bounding $\min\{m, n\}$ is enough and that it is less than $k \cdot F(k)$. We present certain values of $F(k)$ generated computationally. We try to generalize these matrices and obtain a sub-exponential lower bound for $F(k)$. This rules out the possibility of obtaining a polynomial kernel through this technique. From a theoretical point of view, computing or bounding $F(k)$ is interesting and will lead to better insights into the problem.

We then try to answer whether Biclique Partition problem can be solved in polynomial time or not on certain graph classes. In this regard, we prove that Biclique Partition problem is NP-complete on perfect elimination bipartite graphs. We note that all the special graph classes for which the classical time complexity of both the problems is known, either both of them are NP-complete or both of

them are polynomial time solvable. One interesting problem would be to find a graph class on which this nature differs. This will be helpful in getting better insights into the problem. One important observation is to make use of the rank lower bound which is there for biclique partition problem but not in bilcique cover problem. We also note that rank of a random binary matrix can be quite large and thus, solution size, which is lower bounded by rank, might not be the best parameter for the problem. A better parameter would be to consider $k' = k - rank(A)$. It will be interesting to explore if Biclique Partition problem admits a polynomial kernel with k' as the paramter

References

[1] I. Nor, D. Hermelin, S. Charlat, J. Engelstadter, M. Reuter, O. Duron, and M.-F. Sagot, "Mod/resc parsimony inference: Theory and application," *Information and Computation*, vol. 213, pp. 23 – 32, 2012. Special Issue: Combinatorial Pattern Matching (CPM 2010). 1, 16, 19

[2] D. S. Nau, G. Markowsky, M. A. Woodbury, and D. B. Amos, "A mathematical analysis of human leukocyte antigen serology," *Mathematical Biosciences*, vol. 40, no. 3-4, pp. 243–270, 1978. 1

[3] T. Jiang and B. Ravikumar, "Minimal nfa problems are hard," *SIAM Journal on Computing*, vol. 22, no. 6, pp. 1117–1141, 1993. 1, 4

[4] F. Geerts, B. Goethals, and T. Mielikäinen, "Tiling databases," in *International Conference on Discovery Science*, pp. 278–289, Springer, 2004. 1

[5] H. Gruber and M. Holzer, "Inapproximability of nondeterministic state and transition complexity assuming p$\neq$ np," in *International Conference on Developments in Language Theory*, pp. 205–216, Springer, 2007. 1

[6] A. Karrenbauer, "Matching techniques ride to rescue oled displays," in *International Conference on Combinatorial Optimization and Applications*, pp. 110–122, Springer, 2009. 1

[7] J. Orlin, "Contentment in graph theory: Covering graphs with cliques,"

Indagationes Mathematicae (Proceedings), vol. 80, no. 5, pp. 406 – 424, 1977. 4, 6

[8] S. M. Cioaba, *The NP-completeness of some edge-partitioning problems*. PhD thesis, M. Sc. Thesis, Queens University at Kingston, Canada, 2002. 6

[9] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized algorithms*, vol. 3. Springer, 2015. 6, 12

[10] M. Cygan, M. Pilipczuk, and M. Pilipczuk, "Known algorithms for EDGE CLIQUE COVER are probably optimal," *CoRR*, vol. abs/1203.1754, 2012. 6

[11] E. Mujuni and F. Rosamond, "Parameterized complexity of the clique partition problem," in *Proceedings of the fourteenth symposium on Computing: the Australasian theory-Volume 77*, pp. 75–78, Australian Computer Society, Inc., 2008. 6

[12] H. Fleischner, E. Mujuni, D. Paulusma, and S. Szeider, "Covering graphs with few complete bipartite subgraphs," *Theoretical Computer Science*, vol. 410, no. 21-23, pp. 2045–2053, 2009. 6, 7

[13] S. Chandran, D. Issac, and A. Karrenbauer, "On the Parameterized Complexity of Biclique Cover and Partition," in *11th International Symposium on Parameterized and Exact Computation (IPEC 2016)* (J. Guo and D. Hermelin, eds.), vol. 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 11:1–11:13, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. 6, 7, 21

[14] S. Jukna, *Extremal combinatorics: with applications in computer science*. Springer Science & Business Media, 2011. 24

[15] H. Müller, "On edge perfectness and classes of bipartite graphs," *Discrete Mathematics*, vol. 149, no. 1-3, pp. 159–187, 1996. 44

[16] J. Amilhastre, M. Vilarem, and P. Janssen, "Complexity of minimum biclique cover and minimum biclique decomposition for bipartite domino-free graphs," *Discrete Applied Mathematics*, vol. 86, no. 2, pp. 125 – 144, 1998. 44

[17] D. S. Franzblau and D. J. Kleitman, "An algorithm for covering polygons with rectangles," *Information and Control*, vol. 63, no. 3, pp. 164–189, 1984. 44

[18] T. Lavynska, "On biclique cover and partition problems," 44

[19] O. Duginov, "Partitioning the vertex set of a bipartite graph into complete bipartite subgraphs," *Discrete Mathematics and Theoretical Computer Science*, vol. 16, no. 3, pp. 203–214, 2014. 45

[20] D. A. Gregory, N. J. Pullman, K. F. Jones, and J. Lundgren, "Biclique coverings of regular bigraphs and minimum semiring ranks of regular matrices," *Journal of Combinatorial Theory, Series B*, vol. 51, no. 1, pp. 73 – 89, 1991.

[21] H. Fleischner, E. Mujuni, D. Paulusma, and S. Szeider, "Covering graphs with few complete bipartite subgraphs," *Theoretical Computer Science*, vol. 410, no. 21, pp. 2045 – 2053, 2009.

[22] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan, "Fast exact and heuristic methods for role minimization problems," in *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT '08, (New York, NY, USA), pp. 1–10, ACM, 2008.

[23] S. Jukna and A. Kulikov, "On covering graphs by complete bipartite subgraphs," *Discrete Mathematics*, vol. 309, no. 10, pp. 3399 – 3403, 2009.

[24] M. C. Golumbic and C. F. Goss, "Perfect elimination and chordal bipartite graphs," *Journal of Graph Theory*, vol. 2, pp. 155–163, 1 1978.

[25] M. J. Pelsmajer, J. Tokaz, and D. B. West, "New proofs for strongly chordal graphs and chordal bipartite graphs," *preprint*, 2004.

Appendix A

Appendix

Listing A.1: Code snippet from the implementation of Algorithm 2

```
1  int biclique_partition(vector <int> &A, vector <int> &B,
2                         vector <int> &C, int row, int col, int k )
3  {
4      vector<int>I(k);
5      I[k-1] = row;
6      for(int i=k-1;i>0;--i)
7          I[i-1] = I[i]-1;
8      int c=0;
9      int res ;
10     vector<int> one_com(k);
11     for(int i=0;i<k;++i)
12         one_com[i] = (i+1);
13     do
14     {
15         res =0;
16         if (c == 0)
17             ++c;
```

```
18          else
19              get_comb(I,row,k);
20          vector <int> one((k*k),1);
21          vector <int> one_k(k,1);
22          vector <int> B1((k*k),0);
23          do
24          {
25              int j,counter_1;
26              guess_vec(B1,(k*k));
27              for (j=0; j<col;++j)
28              {
29                  vector <int> C1(k,0);
30                  do
31                  {
32                      counter_1 = 0;
33                      guess_vec(C1,k);
34                      vector <int> prod(k,0);
35                      for (int u=0;u<k;++u)
36                          for (int v=0;v<k;++v)
37                              prod[u] = prod[u] +( B1[(u*k)+v]*C1[v]);
38                      for(int u =0;u<k;++u)
39                          if(A[((I[u]-1)*col)+j] != prod[u])
40                          {
41                              counter_1 = 1;
42                              break;
43                          }
44                      if(counter_1 == 0)
```

```
45                          {
46                              for (int u=0;u<k;++u)
47                                  C[(u*col)+j] = C1[u];
48                              break;
49                          }
50                      } while (C1 != one_k);
51                      if((C1 == one_k)&& (counter_1 == 1))
52                          break;
53                  }
54                  if((j == col)&&(counter_1 == 0))
55                  {
56                      res = get_remaining_B(A,B,C,I,B1,row,col,k);
57                      if (res == 1)
58                          return res;
59                  }
60              } while (B1 != one);
61          }while (I != one_com);
62          return res;
63  }
```