Parameterized Complexity of Minimum k Union Problem

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Aditya Kabra

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

April, 2018

Supervisor: Soumen Maity

Certificate

This is to certify that this dissertation entitled Parameterized Complexity of Minimum k Union Problem towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Aditya Kabra at Indian Institute of Science Education and Research under the supervision of Soumen Maity, Associate Professor, Department of Mathematics, during the academic year 2017-2018.

Soumen Maity

Committee:

Soumen Maity

Saket Saurabh

Declaration

I hereby declare that the matter embodied in the report entitled Parameterized Complexity of Minimum k Union Problem 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 Soumen Maity and the same has not been submitted elsewhere for any other degree.

AmK

Aditya Kabra

Acknowledgments

I express my sincere gratitude to Prof. Soumen Maity for supervising me in this project. I am grateful to him for the incredible support and encouragement he has offered me in this past year. Prof. Soumen has aspired me by his teaching in my third and fourth year and motivated me to do a project in this field.

I am indebted to Prof. Saket Saurabh for his constant help and guidance. I am grateful to him for giving me an opportunity to visit the Institute of Mathematical Science and interact with his research group.

I am grateful to Dr. Fahad Panolan for his time. He has offered countless hours selflessly for the discussions which helped me a lot in the completion of this thesis. He has played a vital role in continually pushing me to explore new methods and techniques and to look at the problem from different ways.

I express my gratitude to the faculty members of Department of Mathematics, IISER Pune who have inspired and encouraged me to pursue this academic path. I would also like to thank IISER Pune, for providing me such a fantastic opportunity to learn and explore so many different fields and giving me a chance to undertake this project.

I thank my family who had encouraged me to choose career path in Science. Thanks Maa and Papa, for all the motivation and encouraging words of yours which have kept me going even through the difficult times.

Lastly, I thank my friends Rahul, Shreeya, Ameya, Abhishek, Ashwin, Nidhi, Prachi, Kaustav, Sumit, Anupam, Pritam and Upendra for being with me and listening to all my unrealistic future plans about transforming the world. It would not have been possible to complete this project without your help.

Abstract

Parameterized Complexity offers an alternative perspective to deal with NP-hard problems. For certain problems, when some additional structure (parameter) of the problem is given, there exists efficient algorithms which give exact solutions in polynomial time in input size by separating the combinatorial explosion to the given parameter. Such problems are called Fixed Parameter Tractable and the main goal of the theory is to design such efficient algorithms. Parameterized Complexity theory helps us to determine whether there exists such efficient algorithms for a given problem and helps us to classify the problems in different classes based on their complexity with respect to the parameter.

Covering problems are principal problems in complexity theory. Generally, a family of sets over some finite universe along with an integer k is provided as an input, the goal is to choose k sets from this family, such that the union of these k sets cover the whole universe. A generalization of covering problems is partial cover problems where instead of covering the whole universe, only a specified number of elements are covered with minimum number of sets. Partial Vertex Cover, Partial Dominating Set, and Partial Set Cover are some of the examples of partial cover problems.

A natural variation of such partial cover problem is instead of covering maximum number of elements, the problem is to cover minimum number of elements of the universe by the union of k sets. Minimum k Union is one of such problem, where we are given a family of sets within a finite universe and an integer k and we are asked to choose k sets from this family in order to minimise the size of the union of chosen k sets. Another variation of Minimum k Union problem is Minimum Neighbourhood in Graph problem in which we are asked to select k vertices from the graph such that the size of the neighbourhood formed by these k vertices is minimum.

In this thesis, we study the Minimum k Union Problem, Hall Set Problem and Minimum Neighbourhood Problem in the realms of parameterized complexity. We prove that all of these problems are W[1]-hard.

Contents

Chapter 1

Introduction

In this thesis we consider three optimization problems: Minimum k Union problem, Hall set problem, and Minimum Neighbourhood in Graph. In Minimum k-Union problem (MkU), we are given a set system, and we are asked to select k sets in order to minimize the size of their union. We study MkU, Hall set problem in C_4-free graph, and Minimum Neighbourhood Problem and we prove that all these problems are W[1]-hard.

1.1 Parameterized Complexity

In classical computational complexity theory, the decision problems are broadly divided into two classes P and NP. A decision problem is a problem that can be expressed as a yes-no question of the input values. For example, does there exists a Eulerian trail in the given graph G? or does there exist an independent set of size at most k in a given graph G along with some given $k \in \mathbb{N}$? If a decision problem can be solved in polynomial time then it is contained in the class P. If the solutions of a decision problem can be verified in polynomial time then it belongs to the class NP. The hardest set of these NP problems belongs to the class NP-complete. If every problem in NP can be reduced to a problem Π in polynomial time then Π belongs to the class NP-hard. If a problem is in NP and is NP-hard then it is said to be NP-complete. Approximation algorithms and parameterized algorithms are two ways of dealing with NP-hard problems. Approximation algorithms are algorithms that find approximate solutions to NP-hard problems. While designing

approximation algorithms, our goal is to design a polynomial time algorithm such that the solution it returns is not necessarily optimal, but probably close to optimal solution. The existance of efficient and deterministic algorithms for NP-hard or NP-complete problems is considered unlikely. However, some NP-hard and NP-complete problems can be solved by algorithms that are exponential in the size of a fixed parameter while polynomial in the size of the input. Such problems are called fixed parameters tractable (FPT). The parameter is some structure property of the problem or size of the solution.

Let Σ be a finite set of alphabets. Then $\Sigma^\star$ is the set of all strings x defined over Σ. For example, let $\Sigma = \{0, 1\}$, then $\Sigma^\star = \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$. A language L is a subset of $\Sigma^\star$. For example, a language $L = \{01, 11, 1110, \ldots\}$. Every decision algorithm for a decision problem defines a language L

$$L = \{x \in \Sigma^\star \mid A(x) = \text{yes}\}.$$

If algorithm A determines if a given graph G has a Hamiltonian cycle, then the language L for A is all Hamiltonian graphs encoded as strings over Σ.

Definition 1.1.1. (Parameterized Problem) [4] A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite set of alphabets. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.

A vertex cover of a graph $G = (V, E)$ is a set of $Q \subseteq V$ that contains at least one end point of every edge. Given a graph G, finding minimum size $|Q|$ vertex cover is a hard problem. Given G and a positive integer k, the decision version of the vertex cover problem asks, whether G has a vertex cover of size at most k. A pair (G, k) belongs to vertex cover parameterized language L if and only if G is encoded into a string over Σ and G contains a vertex cover of size k. We now recall the definition of fixed parameter tractable problem.

Definition 1.1.2. (Fixed-parameter tractable) [4] A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called fixed-parameter tractable (FPT) if there exists an algorithm A such that given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, A decides whether $(x, k) \in L$ in time bounded by $f(k)|(x, k)|^c$. $f : \mathbb{N} \to \mathbb{N}$ is a computable function, and c is a constant. The complexity class containing all fixed-parameter tractable problems is called FPT.

If one manages to isolate the exponential terms only to the specific parameter and keep the running time of algorithm polynomial in the input size instance then one may be able to computationally solve the problem for the small values of the parameter. Such algorithms are known as fixed parameter algorithms, which manages to keep the running time polynomial in the input instance by restricting the exponential part to the parameter. Even though the problem is still hard, we are able to computationally solve the problems for small values of parameter. The main goal of parameterized complexity theory is to find FPT algorithms for NP-hard or NP-complete problems.

To determine whether graph G has a vertex cover of size at most k is a NP-complete problem. It is known that there exists an $O(n\sqrt{m} + 1.6181^k k^{O(1)})$ algorithm to answer this question where n is the number of vertices and m is the number of edges in the graph.

Now we recall one more complexity class called XP, the algorithms in XP class are less efficient than FPT and have running time bounded by $f(k)|(x,k)|^{g(k)}$.

Definition 1.1.3. (XP) A parameterized problem $L \subseteq \Sigma^* \times N$ is called slice-wise polynomial (XP) if there exists an algorithm A such that given an input instance $(x,k) \in \Sigma^* \times N$, the algorithm A correctly decides whether $(x,k) \in L$ in time bounded by $f(k)|(x,k)|^{g(k)}$. $f, g : N \to N$ are two computable functions. The complexity class containing all slice-wise polynomial problems is called XP.

Example 1.1.4. A clique of a graph $G = (V, E)$ is a subset of vertices $C \subseteq V$ such that every two distinct vertices in C are adjacent; that is, the subgraph induced by C, $G[C]$ is complete. In clique problem we are given a graph G, the task is to find maximum size clique. Given a graph G and a positive integer k, the decision version of clique problem asks whether G has a clique of size k. It is known that clique problem is NP-complete. Clique problem has only an $\mathrm{O}(n^k)$ algorithm where k is the fixed parameter and n is the size of the input graph.

Parameterized complexity helps us in understanding the influence of the different choices of parameters on the complexity of the problem. By classifying the NP-hard problems into two different classes: problems that admit fixed parameter tractable algorithms, and problems that do not admit fixed parameter tractable algorithms, one can determine whether the problems have efficient solutions for given parameters. The parameterized complexity

theory contains both positive set of techniques for developing FPT algorithms and negative set of techniques also called as parameterized intractability to show non existence of such FPT algorithms. One has to use both the instruments in order to classify the problems and design efficient algorithms. For more details on the theory of parameterized complexity refer to [3, 4]

1.2 Introduction to Minimum k Union Problem

Covering problems are classical problems in combinatorial optimization, computer science and complexity theory. They have been widely studied using both optimization algorithms and parameterized algorithms. Vertex Cover, Dominating Set, and Set Cover are few of the variations of the covering problems. For example in the set cover problem, a family $\mathcal{F}$ of sets over a universe U is given, the goal is to cover the whole universe U with the minimum number of sets from F.

A natural generalizations to these covering problems is partial covering problem in which instead of covering all the elements of the universe, the goal is to cover only a specified number of elements with the minimum number of sets. In Partial Set Cover (PSC) the goal is to cover at least t elements of U with minimum number of sets from $\mathcal{F}$.

Another natural problem is to cover at most t elements of U with maximum number of sets from $\mathcal{F}$. In the Minimum k-Union problem (MkU), we are given a set system with m sets and are asked to select k sets in order to minimize the size of their union.

Definition 1.2.1. (Minimum k Union) We are given an universe U of n elements, and a collection of m many sets $\mathcal{F} \subseteq 2^U$, as well as an integer $k \leq m$. The goal is to return a collection $T \subseteq \mathcal{F}$ with $|T| = k$ in order to minimize $|\bigcup_{S \in T} S|$.

Let $U = \{1, 2, 3, 4\}$ and $\mathcal{F} = \{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{2, 4\}, \{1, 2, 4\}, \{2, 3, 4\}\}$ and $k = 3$. Here $n = 4$ and $m = 8$. Then a minimum 3-union is $\{1, 2\}$, union of $\{1\}, \{2\}$, and $\{1, 2\}$.

While different variations of partial set cover were studied intensively and their parameterized complexity results exist in the literature (for example Partial Set Cover is FPT [5], partial vertex cover is W[1]-complete [8]), only few results are known about Minimum

k Union problem. This may in part because Minimum k Union seems to be significantly harder than Maximum Coverage. In this thesis we initiate parameterized complexity study of Minimum k Union problem and its variations.

We refer to [5, 8] for further details on partial coverings, and minimum k union problem.

1.3 Organization of the thesis

This thesis is largely divided into following four components: Introduction; Fixed Parameter Tractable Algorithms and Intractability; Main Results; and Conclusion.

Chapter 2 consists of introductory definitions, notations and a few algorithmic techniques which will be useful for going through this thesis. Section 2.1 consists of description of Time Complexity, Section 2.2 consists of some basic graph theoretic definitions and notations and the final Section 2.3, consists of tree decomposition of a graph and tree-width.

In Chapter 3, we describe various techniques and algorithms related to parameterized complexity. In Section 3.1, we describe the techniques to prove a parameterized problem is fixed parameter tractable, such as Kernelization, Bounded Search Tree and Treewidth. In Section 3.2, we give a brief description about fixed parameter intractability, parameterized reductions and W-hierarchy.

In chapter 4, we present our solutions. In section 4.1, we show Minimum k Union problem is W[1]-hard by giving a reduction from independent set. In Section 4.2, we study the Hall Set Problem in C_4 free bipartite graph and prove that it is W[1]-hard. In Section 4.3, we study minimum neighbourhood in graph problem and prove that it is W[1]-hard.

In Chapter 5, we conclude the thesis with major contributions of this thesis and list some open problems.

Chapter 2

Preliminaries

In this chapter we recall some basic definitions and relevant results from graph theory used in this thesis. For more details see [1, 2].

We use $\mathbb{N}$ to denote the set of natural numbers. We define $[n]$ to be the set $\{1, 2, \ldots, n\}$. Let U be an universal set. We use 2^U to denote family of all the subsets of U. For any two subsets X and Y of the universe U, we use $X - Y$ to denote the subset of X whose elements are not in Y.

2.1 Time Complexity

The running time of an algorithm is the number of basic operations performed by the algorithm on the worst case. The basic operations include assignment, comparison between two variables, arithmetic operations, etc. For example, running time to multiply two $n \times n$ matrices using classical technique is $T(n) = n^2(2n-1) = 2n^3 - n^2$. We mainly use Big-O notation to represent the time complexity of an algorithm. Let $g : \mathbb{N} \to \mathbb{N}$. We say $T(n) = \mathrm{O}(g(n))$ if there exists positive constants c and n_0 such that $0 \leq T(n) \leq cg(n)$ for all $n \geq n_0$. Thus, to multiply two $n \times n$ matrices using classical technique, we require $O(n^3)$ time.

2.2 Graphs

A *graph* G is represented by $G = (V, E)$. Elements of the set V represents the collection of the vertices of G and elements of the set E represents the collection of the edges of G, satisfying $E \subseteq V \times V$. uv denotes an edge joining vertices u and v in G. The vertices u and v are said to be adjacent if there is an edge joining u and v otherwise they are said to be non-adjacent.

The neighbourhood and closed neighbourhood of the vertex v in G are denoted by $N_G(v)$=$\{u \in V(G) : uv \in E(G)\}$ and $N_G[v] = N_G(v) \cup \{v\}$ respectively. The degree of v, denoted by $d_G(v)$, is equal to $|N_G(v)|$. If all the vertices of the graph have same degree, suppose r then the graph is called r-regular graph.

For a graph H and G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ then graph H is *subgraph* of G. For a $S \neq \emptyset$, subset of $V(G)$, the subgraph induced by elements of S is denoted by $G[S]$. The elements of the set S represents the vertex set of $G[S]$ and the edge set of $G[S]$ is all the elements of $E(G)$ whose both endpoints lies in S.

An edge of the form uu is called a *loop*. *Parallel edges* are two or more edges with the same endpoints. If a graph has no parallel edges and it has no loops, then it is called *simple*. A *multigraph* is a graph where we allow parallel edges and loops. Unless otherwise specified, all the graphs considered are simple in this thesis.

An *undirected graph* is where $uv = vu$, that is, the orientation of the edges doesn't matter. Whereas in a directed graph, each edge has a direction. We denote an edge of directed graph as (u, v) and it is said to be directed from u to v.

A *walk* in G is an alternating sequence $v_0, e_1, v_1 \ldots, e_k, v_k$ of vertices and edges such that $e_i = v_{i-1}v_i \in E(G)$. A walk is called a path if all the vertices v_i's are different. If $v_0 = v_k$ and all the vertices are distinct then it is called a cycle. C_n represents a cycle graph with n vertices. The number of vertices and the number of edges in C_n are same with each vertex having its $degree = 2$; C_n is also called the cycle of length n.

If vertices of a graph G can be partitioned in two subsets (A, B), such that all the edges of G have one endpoint in A and other in B, then the G is said to be a *bipartite graph*. It is known that if a graph does not contains any odd cycles (cycles of odd length), only then it

is bipartite.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, then the union of these two graphs is represented by $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.

If there exists a path between every pair of the vertices in G, then G is *connected graph*. A connected graph without cycles is called a *tree*. A forest is graph without cycles which need not necessary to be connected.

If a graph G can be embedded into a plane then it is called a *planar graph* , in other words, a planar graph can be drawn on a plane in such a way that the edges intersect only at their endpoints.

If in a set of edges M, every vertex of G is incident to at most one edge in M, then M is called Matching.

We give below some definitions of the problems which are used in this thesis.

- In a given graph G, a subset C of V(G) is a *clique* if all the vertices of C are pairwise adjacent.
- In a given graph G, a subset I of V(G) is an *independent set* if all the vertices of I are pairwise non-adjacent.
- In a given graph G, a subset S of V(G) is a *vertex cover* if all the edges of G have at least one endpoint in S. (In a graph G if S is a vertex cover of G, then $G - S$ is an independent set.)
- In a given graph G, a subset D of V(G) is a *dominating set* if all the vertices not in D have at least one neighbour in D.
- In a given graph G, a set H is a *hall set* if G is a bipartite graph with bipartite classes (A,B), and for a set $H \subseteq A$, $|N(H)| < |H|$

2.3 Crown Decomposition

Definition 2.3.1. (Crown Decomposition) A crown decomposition of a graph G is a partition of the vertex set $V(G)$ into three parts C, H and R, such that

1. C is a nonempty, independent set.

2. There are no edges between vertices of C and R. That is, H separates C and R.

3. Let E' be the set of edges between vertices of C and H. Then E' contains a matching of size $|H|$.

We give an example to illustrate a crown decomposition of a graph.

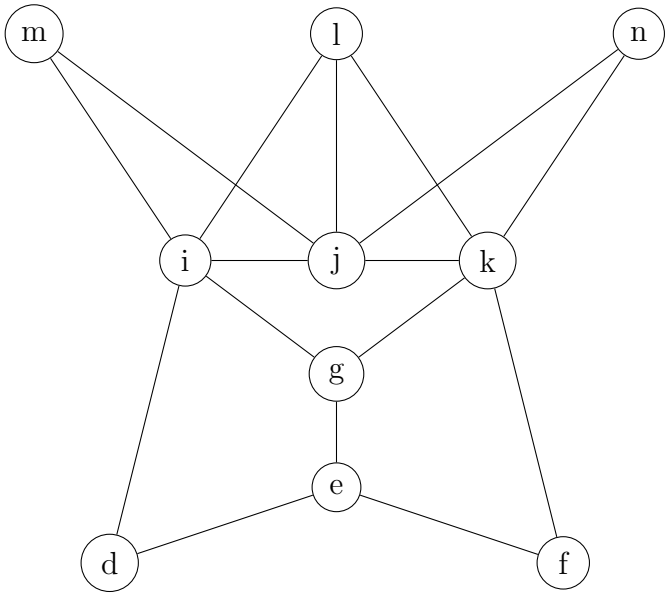


Figure 2.1: Graph G and its Crown decomposition

Figure 2.3.1 shows a graph G and its tree decomposition. The crown decomposition of G is into sets C,H and R such that the set $C = \{l, m, n\}$ is an independent Set, set $H = \{i, j, k\}$ seperates C and $R = \{d, e, f, g\}$. And, there is a matching M of H into C given by the edges $M = \{mi, lj, nk\}$.

2.4 Tree Decomposition of a Graph

Definition 2.4.1. (Tree Decomposition) Let $G = (V, E)$ be a graph. A tree decomposition of G is a pair $(\{V_i | i \in I\}, T = (I, F))$ with $\{V_i | i \in I\}$ a family of subsets of $V(G)$ and T a tree (note that V_i are the vertices of T), with the following properties:

1. $\cup_{i \in I} V_i = V(G)$

2. For every edge $e = (u, v) \in E(G)$, there is an $i \in I$ with both the vertices u and $v \in V_i$

3. for every $v \in V(G)$, the set $\{i | v \in V_i\}$ forms a connected subtree of T.

The treewidth of a tree decomposition $\{V_i | i \in I\}$ is $max_{i \in I}(|V_i| - 1)$. The treewidth of G denoted by $tw(G)$ is the minimum treewidth, taken over all possible tree decomposition of G. We give an example to illustrate tree decomposition of a graph and treewidth.

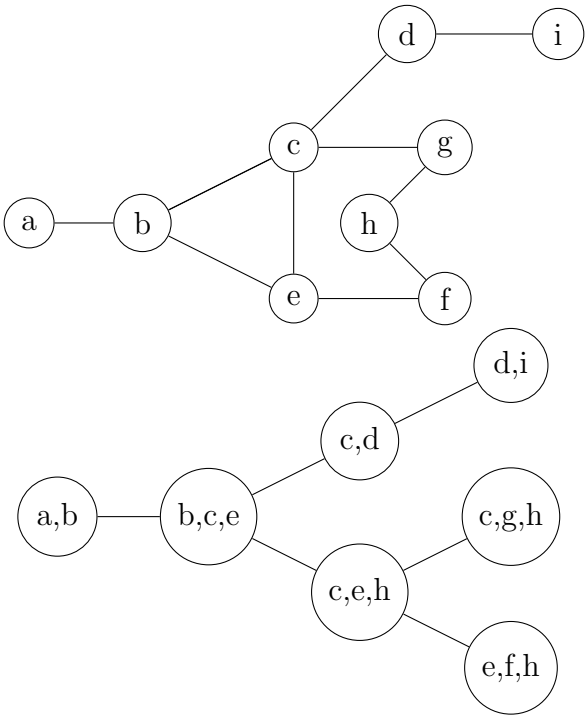


Figure 2.2: Graph G and its tree decomposition

Figure 2.4.1 shows a graph G and its tree decomposition. The tree decomposition of G is $V_1 = \{a, b\}, V_2 = \{b, c, e\}$, $V_3 = \{c, d\}, V_4 = \{d, i\}$, $V_5 = \{c, e, h\}, V_6 = \{c, g, h\}, V_7 = \{e, f, h\}$. The endpoints of each edge lie in one of the V_i's. Consider a vertex, say, $c \in V(G)$; c lies in $V_2 = \{b, c, e\}$, $V_3 = \{c, d\}$, $V_5 = \{c, e, h\}, V_6 = \{c, g, h\}$. Now the subgraph induced by vertices V_2, V_3, V_5, V_6 is a connected subtree of the tree T. The width of this tree decomposition is 2. It is easy to verify that the treewidth of G is 2.

Chapter 3

Parameterized Complexity

3.1 Techniques to prove FPT

Parameterized algorithms are broadly classified into two categories fixed parameter tractable and fixed parameter intractable. The problems which have algorithms with a running time bounded by $f(k).n^{O(1)}$ (k is some parameter of the problem and n is input value) are called fixed parameter tractable, whereas the rest of the problems who have less efficient algorithms belongs to fixed parameter intractable class. There are various techniques to design fixed parameter tractable algorithms. In this section we study three most popular techniques: Kernelization, Bounded Search Tree and Bounded Treewidth.

3.1.1 Reduction to Kernel

There are various preprocessing algorithms which efficiently solves the easy part of the problem instance and reduces it to its computationally difficult core structure - the kernel of the instance. Kernelization algorithms are a type of preprocessing algorithms that reduces (but not necessarily solve) the given problem instance to an equivalent smaller sized instance in polynomial time in the input size. The size of the output instance given by kernelization algorithm is finite and bounded by a computable function of the parameter.

In this section, we study formal definitions of kernelization and then move on to study the polynomial-time preprocessing algorithms such as crown decomposition lemma, expansion lemma, and linear programming.

Definition 3.1.1. (Kernelization, Kernel) An algorithm A is called a kernelization algorithm for a parameterized problem Q, if given an instance (x, k) of Q the algorithm A returns an equivalent instance (x', k') of Q in polynomial time. Moreover, it is required that $|x'| + k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

We state the following theorem without proof.

Theorem 3.1.2. *[4] A parameterized problem Q is FPT if and only if admits a kernelization algorithm.*

We now illustrate a simple example of kernlization algorithm used to reduce vertex cover problem to its kernel. We first formally define a vertex cover problem in the realms of parameterized complexity.

Definition 3.1.3. (Parameterized Vertex Cover) In the parameterized version of vertex cover problem, as input we are given a graph G and a positive integer k, and the goal is to decide whether there exists a vertex cover of size at most k.

Theorem 3.1.4. *[4] Vertex Cover Problem admits a kernel with* $\mathrm{O}(k^2)$ *vertices and* $\mathrm{O}(k^2)$ *edges.*

Proof. Given an instance (G, k) of the vertex cover problem, we apply following two reductions repeatedly until none of these can be applied anymore on G:

1. If v is an isolated vertex in G, then delete v and get a new instance $(G - v, k)$.
2. If for some v, $d(v) > k$ then add v to the vertex cover and get a new instance as $(G - v, k - 1)$.

Suppose (G, k) is the reduced yes-instance and none of 1 and 2 can be applied anymore. Then G has no isolated vertices, thus every vertex in G is adjacent to at least one vertex of vertex cover. Moreover, each of the vertices has degree at most k. Hence, if S is a vertex cover of G then $|V(G - S)| \leq k|S|$. If G contains a vertex cover of size at most k then

$|V(G)| \leq (k+1)|S| \leq k(k+1)$. Moreover, as every edge is covered by some vertex of vertex cover $E(G) \leq k|S| \leq k^2$.

Let (G, k) be an input instance such that reduction 1 and 2 are not applicable, then if $k < 0$ and $|V(G)| \leq (k+1)k$ or $E(G) \leq k^2$, then we conclude that we are dealing with a no-instance. Thus, we have successfully managed to reduce the vertex cover problem to a kernel of $\mathrm{O}(k^2)$ vertices and $\mathrm{O}(k^2)$ edges.

Next, we discuss Edge Clique Cover problem and see that this problem does not admit polynomial kernel.

Definition 3.1.5. (Edge Clique Cover) Given a graph G and a non-negative integer k, the goal is to decide whether G can be covered by at most k cliques.

Theorem 3.1.6. *[4] There exists a kernel with at most* 2^k *vertices for edge clique cover.*

Proof. Given an instance (G, k) of the Edge Clique Cover problem, we apply following three reductions repeatedly until none of these can be applied anymore on G:

1. Delete isolated vertices.
2. Delete isolated edge and decrease k by 1.
3. If $N[u] = N[v]$ and there is an edge between u and v, then delete v.

We claim that if G has an Edge Clique Cover of size k and none of the reduction rules are applicable, then G has less than 2^k vertices. Suppose $C_1, C_2, ..., C_k$ are k cliques that cover G. Assume G has more than 2^k vertices. We assign a vector b_v of length k to each vertex v of G. The ith bit of b_v is set to 1 if and only if v is present in the i th clique, 0 otherwise. As there are only k positions and each position can take only two values, there can be only distinct 2^k vectors. For the sake of contradiction, suppose there are more than 2^k vertices in G. That means, there are two distinct vertices u and v with the same associated vector. If two vertices have the same associated vector with them, then there can be following two cases: (1) both the vectors are zero vector, in this case both the vertices are isolated and there cannot be isolated vertices in G because of the 1st reduction rule; (2) both the vertices u and v have the same non-zero vector associated with them, in this case u and v are in the

same clique. That means u and v are adjacent and have the same closed neighbours. Hence by reduction rules 3, v has been deleted. Hence, there can be maximum 2^k vertices in G.

We now give another very useful technique in kernelization known as Crown's Lemma.

Lemma 3.1.7. *(Crown Lemma). Given a graph G with no isolated vertices and $|V| \geq 3k+1$. Then, there is a polynomial-time algorithm that either finds a matching of size $k+1$; or finds a crown decomposition of G.*

We skip the proof of Crown Lemma and illustrate the use of this technique using Vertex Cover problem.

Theorem 3.1.8. *Vertex Cover admits a kernel with at most $3k$ vertices.*

Proof. GIven (G, k) an instance of vertex cover problem, we deleter all the isolated vertices from G and name this process as reduction rule 1. Now, If $V(G) > 3k$, we apply Crown lemma on G and find either a $k+1$ size matching or divide vertex sets $V(G) = C \cup H \cup R$ representing a crown decomposition.

If a matching of size $k+1$ exists in G, to obtain a vertex cover of size $k+1$, from each edge of the matching we have to choose one vertex. Thus, the vertex cover size will be more than k, therefore we conclude that it's a no instance.

In the latter case, if M is matching of H into C. To cover the edges of M, from $H \cup C$ we must choose at least $|M| = |H|$ vertices in each vertex cover . Set H covers all the edges of G that are incident to $H \cup C$. Hence, the minimum vertex cover of G must contain H and thus (G, k) is reduced into $(G - H, k - |H|)$.

Hence, unless $H = \emptyset$ Crown Lemma can be always applied and the graph can be reduced as long as $V(G) > 3k$. Therefore, G admits a kernel of size at most $3k$ vertices.

3.1.2 Bounded Search Trees

The idea here is to identify some search spaces whose sizes are bounded by a function of the parameter. Thus the algorithm effectively branches into number of sub-problems that are solved one by one. This can be viewed as a search tree where branching algorithm branches

on the leaves and discovers the solution. If depth of such a recursion tree is bounded by k and polynomial time is spent on each node, then the resulting branching algorithm runs in FPT time. Let I be an instance of a minimization problem. In a branching step we generate simpler instances $I_1, ..., I_l$ $(l \geq 2)$ from I of the same problem. We recursively apply branching step on $I_1, ..., I_l$ $(l \geq 2)$ until they become simple or trivial.

We illustrate the bounded search tree technique using vertex cover problem. We base our algorithm on following two observations in the vertex cover problem:

1. Any vertex cover in a graph G must contain either a vertex v or all of the vertices of $N(v)$. Otherwise, all the edges won't be covered.

2. If the maximum degree of the vertices in the graph is 2 then one can solve the Vertex Cover problem in polynomial time.

Now, we generate a recursive algorithm as following: First, find a maximum degree vertex v in the graph. If $d(v) \leq 2$, the instance can be solved in polynomial time and if $d(v) > 2$, the algorithm recursively branches on two cases: either v or $N(v)$ are in vertex cover. When v is in vertex cover, we can decrease our parameter by 1, and when $N(v) \geq 3$ is in vertex cover, we decrease our parameter by at least 3.

Hence, we obtain following two subtrees: $T(k-1)$ and $T(k-3)$ for a given tree $T(k)$. Our recursive relation is represented as follows:

$$T(i) = T(i-1) + T(i-3) \text{ if } i \geq 3 \text{ or else}$$
$$T(i) = 1 \text{ if } \quad i < 3$$

The above relation not only bounds the size of the search tree but also the time spend at each node of the tree. Using standard mathematical techniques, it can be proved that $T(k) < 1.466^k$. Hence, the running time of the algorithm which solve's parameterized vertex cover is $O(1.466^k n^{O(1)})$

3.1.3 Bounded Treewidth

A tree decomposition of a graph G is a mapping of G into a tree that can be used to define the treewidth of G. See Section 2.4 for details. If a graph G has a smaller treewidth, then many NP-completer problems which are intractable become efficiently solvable for G.

For a parameterized problem, if a treewidth is bounded by some function of the given parameter, then fast dynamic programming or MSO formula along with the Courcelle's theorm can be used to obtain fixed parameter tractable algorithm. We illustrate the dynamic programming technique through the example of weighted independent set problem:

Definition 3.1.9. (Weighted Independent Set) Given a graph G with real valued vertex weights w, $w : V(G) \to \mathbb{R}$, the goal is to find an independent set in G with maximum possible total weight.

Theorem 3.1.10. *Given a weighted graph $G = (V, E)$, $|V| = n$ along with it's tree decomposition of width at most k. The time required to find a maximum Weighted Independent Set in G is $2^k k^{\mathrm{O}(1)} n$.*

Proof. In this thesis, we just provide an outline of the proof.

It is well known that, a nice tree decomposition of a graph of at most same width can be constructed in polynomial time, if a tree decomposition of the graph is given. Hence, we assume the given tree decomposition is nice.

Let $T = (T, \{X_t\}_{t \in V(T)})$ represents a nice tree decomposition of G with width at most k, with root r.

Let t be a node in T, V_t be the bag asssociated with t and $L_t \supset V_t$ contains all the vertices of the union of all subtrees of T which are rooted at t.

Due to the border condition, the induced subgraph by L_t is only connected by the vertices of V_t to the rest of the graph. Hence, we can generate a dynamic programming algorithm to solve this problem in smaller subgraphs and by giving a bottom up approach, find out the solution in the given T.

We define the terms of the algorithm as $c[t, S]$ for all nodes t and set S, such that $S \subseteq V_t$ the following way:

$c[t, S]$=Max weight set $\{S^e : S \subseteq S^e \subseteq L_t, S^e \cap V_t = S$ and S^e is independent. $\}$

Note that $c[t, S] = -\infty$ if no such S^e exists. We are looking for $c[r, \emptyset]$ as $L_r = V(G)$ and $V_r = \emptyset$,in the given graph G the term represents maximum weighted independent set.

We now describe the recursive formulas for computing the values of nodes from previously computed node:

- Leaf node: As all leaf nodes are empty, hence the independent set is empty, thus in a leaf node t, $c[t, \emptyset] = 0$.
- Introduce node t with child t' with some introduced vertex $v \notin V_{t'}$: Thus, given any independent subset S of V_t, we get two cases: if $v \notin S$ then $c[t, S] = c[t', S]$ and if $v \in S$ then $c[t, S] = c[t', S \backslash v] + w(v)$.
- Forget node t with child t' with some forgotten vertex $w \in V_{t'}$: Thus, given any independent subset S of V_t, we choose maximum of the following two cases: $c[t, S] = max\{c[t', S], c[t', S \cup w]\}$
- Join node t with children t_1, t_2 with $V_t = V_{t_1} = V_{t_2}$: Let S be any independent subset of V_t; Then $c[t, S] = c[t_1, S] + c[t_2, S] - w(S)$.

As treewidth of the graph is at most k, for each node t, $|V_t| \leq k + 1$. We compute $2^{|V_t|} \leq 2^{k+1}$ values of $c[t, S]$ at node t. And, each $c[t, S]$ term can be computed in $k^{\mathrm{O}(1)}n$. Hence, the at each node t, the running time to compute all values of $c[t, S]$ is $2^k k^{\mathrm{O}(1)}n$, as the number of nodes of is of the order $O(kn)$. Thus, the total running time is $2^k k^{\mathrm{O}(1)}n$.

On planar graphs, Shifting Technique can be used to give a tree decomposition with treewidth of linear bound.

Corollary 3.1.11. *For a planar graph G, consider a vertex v in G. Let L_i be a set of vertices of G at distance exactly i from v. Then for any $i, j \geq 1$, a tree decomposition of subgraph $G_{i,i+j-1} = G[L_i \cup L_{i+1} \cup, ..., L_{i+j-1}]$ of treewidth at most $3j + 1$ can be obtained in polynomial time.*

We now give the algorithm for the Shifting Technique:

1. Given a input planar graph G, select a vertex u in the graph G. Starting at u, run Breadth First Search (BFS) in G.

2. For any choice of $l, m \geq 1$, consider the levels $l, l+1, ..., l+m-1$ of BFS, the treewidth of the subgraph $G_{l,l+m-1}$ induced by these vertices does not exceed $3m+1$ because of the above mentioned corollary.

3. For solving some problem in G, let us assume a set S is a solution of size k that has some property in G. We further assume that S exists.

4. As our set S is of size k, taking $m = k+1$, we can make sure that out of every $k+1$ layers of BFS, there exist one layer that do not contain any vertex of S. We consider these layers which do not contain S as $a(k+1)+q$ for some particular $q \in \{0, 1, ..., 3k+2\}$ and for $a = 0, 1, 2, \ldots$.

5. By removing these above layers from G we get a new disjoint graph G_q which still contains all the elements of set S. Such G_q is a disjoint union of graphs $G_{(k+1)a+q+1,(k+1)(a+1)+q-1}$ for $a = 0, 1, 2, \ldots$ plus possibly G_v^{q-1} if $q > 0$.

6. Even by not selecting any vertices of S from the levels of the form $a(k+1)+q$, a tree decomposition of G_q of width bounded by $3k+1$ can be still constructed.

7. To find whether there exists a solution S in G_q, on its obtained tree decomposition we run a dynamic-programming algorithm. If a solution S exists in G, then the solution will survive in at least one of the graphs G_q for some $q \in \{0, 1, ..., k\}$ and iterating through all possible q, a solution can be uncovered.

We now Illustrate the Shifting Technique using the Subgraph Isomorphism Problem.

Definition 3.1.12. (Subgraph Isomorphism) Given graphs G and H, the goal is to find whether there exist a subgraph of G which is isomorphic to H.

Theorem 3.1.13. *[4] Given planar graphs G and H, an instance (G, H) of Subgraph Isomorphism can be solved in $f(|V(H)|).|V(G)|^{O(1)}$ time.*

Proof. Let $k = |V(H)|$. Using Shifting Technique, one can partition $V(G)$ into $k+1$ subsets $V_0 \cup V_1 \cup ... \cup V_k$ in such a way that the graph $G - V_j$ for all $j \in \{0, ..., k\}$ has a tree decomposition bounded by treewidth $3k+1$.

By using a lemma which shows that in $f(|V(H)|, t).|V(G)|^{O(1)}$ time, an instance of subgraph isomorphism can be found out (t is the treewidth). As $t = 3k + 1$ for each $G - V_j$, the running time will be $f(k).n^{O(1)}$ on every graph $G - V_j$.

As the vertices of G are divided into $k + 1$ sets, there exists at least one set among $V_0 \cup V_1 \cup ... \cup V_k$ which is disjoint from set H of size k. Suppose, V_j is that disjoint set. Hence, the subgraph H is contained in $G - V_j$. Thus, by trying out each of the $G - V_i$ for all $i \in \{0, ..., k\}$, and then solving the subgraph isomorphism problem in them, one solves subgraph isomorphism problem in G.

3.2 Fixed Parameter Intractability

To rule out certain problems are not FPT, there is a notion of lower bound. This lower bound theory is is similar to the NP-completeness theory. But there is one difference that in parameterized complexity, there are different hardness classes, unlike in classical complexity where all the NP-hard problems are reducible to each other. The hardness classes in parameterized complexity are represented by $W[1], W[2], \ldots$.

The primary assumption here is $FPT \neq W[1]$ which is a stronger assumption than $P \neq NP$.

For our purposes, we mainly try to rule out the existence of an FPT algorithm for MkU problem. It is known that CLIQUE parameterized by solution size is $W[1]$-complete. This means that $W[1]$ is the set of all problems that can be obtained through a parameterized reduction from CLIQUE parameterized by solution size.

3.2.1 Problem Reductions

Definition 3.2.1. (Parameterized reduction) Suppose $A, B \subseteq \Sigma * \times \mathbb{N}$ are two parameterized problems. An algorithm is called a parameterized reduction from A to B, if given an instance (x, k) of A, it outputs an instance $(x', k'0)$ of B such that

1. (x, k) is a yes-instance of A if and only if (x', k') is a yes-instance of B,

2. $k' \leq g(k)$ for some computable function g, and

3. the running time of this algorithm is $f(x).|x|^{O(1)}$ for some computable function f.

Note that not all polynomial time reductions are parameterized reductions due to above mentioned second property and the functions f and g are non-decreasing.

The following theorems holds for parameterized reduction which we state without the proof.

Theorem 3.2.2. *[4] Let parameterized problem B be FPT. If there is a parameterized reduction from A to B, then A is also FPT.*

Theorem 3.2.3. *[4] If there are parameterized reductions from A to B and a parameterized reduction from B to C, then there is a parameterized reduction from A to C.*

We now proceed with the illustration of reduction technique by giving an example of reduction from Clique to Hall Set Problem. We first define a Hall Set Problem and then give a reduction.

Definition 3.2.4. (Hall Set Problem) In a bipartite graph G with bipartite classes (A, B) and given a positive integer k, does there exists a Hall Set of size at most k?

Theorem 3.2.5. *[4] There is a parameterized reduction from Clique to Hall Set on general bipartite graph.*

Proof. Given a graph G, we construct a bipartite graph H with bipartition (A, B) as follows. The vertices of the class A correspond to the edges $e = (uv)$ of G, the vertices of the class B correspond to the vertices v of G. The vertex $e = (u, v)$ of A is adjacent to the two vertices $u, v \in B$. Moreover, an additional set of $\binom{k}{2} - k - 1$ vertices are introduced into B. These additional vertices are made adjacent to every vertex of A. Thus $|A| = |E(G)|$ and $|B| = |V| + \binom{k}{2} - k - 1$.

Suppose there exists a Clique of size at most k in the graph G. The corresponding $\binom{k}{2}$ edges of G that forms Clique are the chosen vertices of the Set S in A. The vertices of set S has k neighbours (the vertices of clique) in Set B, along with the constructed $\binom{k}{2} - k - 1$ neighbours. Hence, the size $N(S)$ is $\binom{k}{2} - 1$. As $|S| \geq |N(S)|$, the set S formed by the edges of Clique is a Hall Set in the bipartite graph H.

Conversely, let S be a Hall set in H. We first determine the possible choices for the size of S. Every vertex of S has at least $\binom{k}{2} - k + 1 = \binom{k-1}{2}$ neighbours in B (from the construction). Hence, the size of S must be greater than $\binom{k-1}{2}$. If $|S| = \binom{k-1}{2}$, then it is easy to observe that $|N(S)| \geq (k-1) + \binom{k}{2} - k - 1 = \binom{k}{2} - 2$. For S to be a Hall set its size must be $\binom{k}{2} - 1$ or $\binom{k}{2}$. If $|S|$ is $\binom{k}{2} - 1$ or $\binom{k}{2}$, then $|N(S)|$ is $\binom{k}{2} - 1$. Thus the minimum possible size of a Hall set is $\binom{k}{2}$. Let S be a Hall set of size $\binom{k}{2}$. As $|N(S)| \leq \binom{k}{2} - 1$, the nodes in S are the edges of a clique of size k in G.

3.2.2 Complexity Classes

The hard parameterized problems have different levels of complexities unlike the NP-complete problems in which all problems are equivalent and can be reduced to each other in polynomial time. For example, Clique can be reduced to Dominating Set but there is no parameterized reduction in other direction as per our current knowledge. Thus, there is a hierarchy in hard parameterized problem, where Dominating Set is harder than the Clique. To capture the exact complexity of hard parameterized problems, Downey and Fellows introduced the concept of the W-hierarchy.

We introduce the terms used for describing W-hierarchy in the following section.

Definition 3.2.6. (Boolean Circuit) A boolean circuit is a directed acyclic graph where the nodes are labeled in the following way:

- every node of indegree 0 is an input node,
- every node of indegree 1 is a negation node,
- every node of indegree $\geq$ is either an and-node or an or-node.

Additionally, exactly one of the nodes with outdegree 0 is labeled as the output node (in addition to being, for example, an or and node).

The input nodes are assigned value 0-1, the value of other nodes can be determined subsequently. We say the assignment satisfies the circuit if the value of the output node is 1 for the given input values. The weight of the assignment is the number of input gates that receives value 1.

Verifying the given assignment satisfies the circuit can be done in polynomial time, but it is an NP-complete problem to find whether a circuit has a satisfying assignment or not.

Definition 3.2.7. (Weighted Circuit Satisfiability (WCS) Problem) Given a Boolean circuit C and a positive integer k, the goal is to decide whether C has a satisfying assignment of weight exactly k.

The *depth* of the circuit is the maximum length of the path from an input node to the output node and the *weft* of a circuit C is the maximum number of large gates on an input-output path in C. The class of the circuits with weft at most t and depth at most d are denoted by $C_{t,d}$

If there exists a parameterized reduction from WCS $[C_{1,d}]$ to some problem, then the problem is W[1]-hard. Hence, by giving a parameterized reduction from WCS $[C_{1,d}]$ to a problem, whose complexity is unknown, we can find out a lower bound of complexity of the problem.

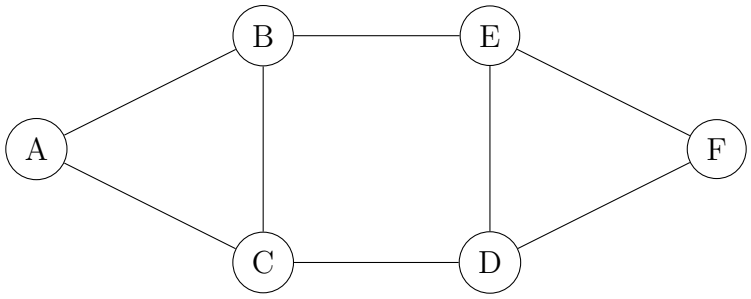
Definition 3.2.8. (W-hierarchy) For $t \geq 1$, a parameterized problem P belongs to the class $W[t]$ if there is a parameterized reduction from P to WCS$[C_{t,d}]$ for some $d \geq 1$.

If a problem is W[1] hard and there exists a other way reduction from the problem to WCS $[C_{1,d}]$ then the problem is W[1]-Complete.

Similar to $P \subseteq NP$, in parameterized complexity $FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[P]$.

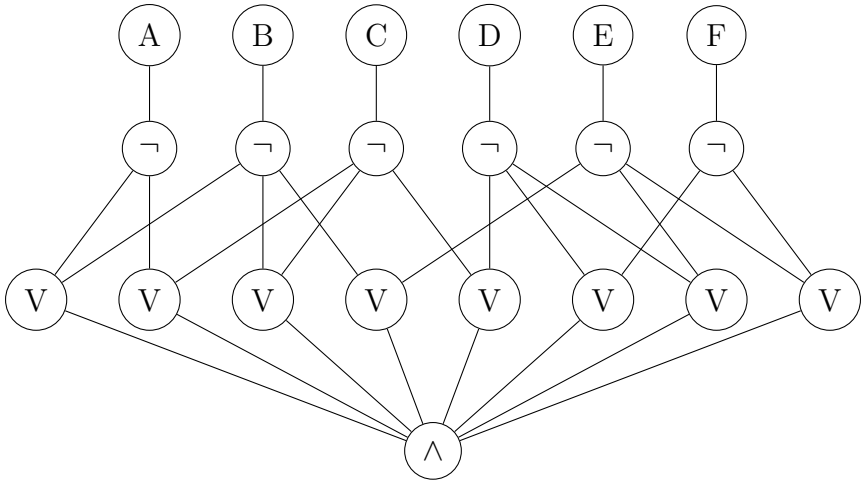
WCS $[C_{1,d}]$, Clique, Independent Set, Partial Vertex Cover are W[1]-complete problems whereas WCS $[C_{2,d}]$, Dominating Set, Set Cover are W[2]-complete problems.

We illustrate the W-hierarchy reduction technique by giving an example of a graph and generating an equivalent circuit diagram to find Independent Set and Dominating Set in the given graph.



Theorem 3.2.9. *[4] Independent Set is in* $W[1]$.

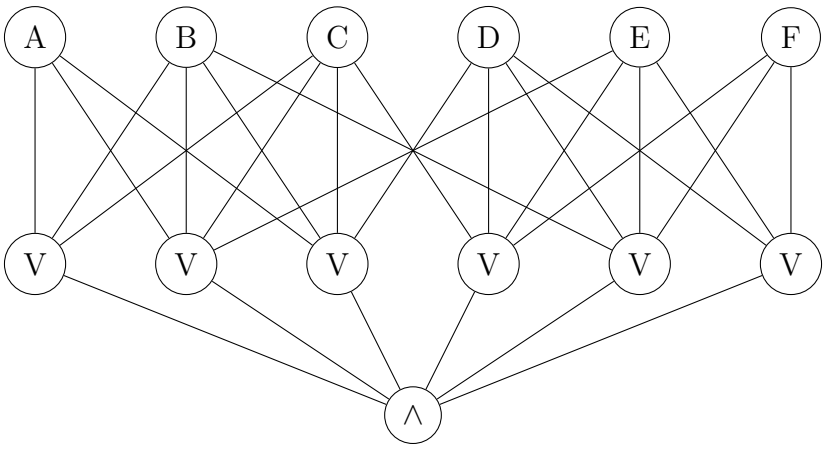
We give an example to illustrate a parameterized reduction from Independent set to $WCS[C_{1,3}]$ problem. Given an instance (G, k) of Independent Set problem we construct an Instance of $WCS[C_{1,3}]$ problem. We construct a depth-3, weft-1 boolean circuit from the graph G shown above. Each or-node corresponds to an edge of G and has in-degree 2. Note that graph G has an independent set of size at most 2, for example $\{A, F\}$ is an independent set in G. We assign value 1 to the nodes correspond to vertices A and F, and 0 to rest of the vertices in the Boolean circuit. This is a input instance of weight 2 for the circuit; and its a satisfiable instance. Thus, given an independent set of size 2 in G, we get a satisfiable instance of weight 2 in the circuit. Similarly, it is easy to observe that if the boolean circuit has a satisfiable instance of size 2, then the graph has an independent set of size 2.



Theorem 3.2.10. *[4] Dominating Set is in* $W[1]$.

Given an instance of Dominating set we will see how to construct an instance of WSC problem, using an example. Consider the graph G shown above as an instance of Dominating

Set problem. We will construct construct a depth-2, weft-2 boolean circuit. Correspond to each node in G there is a or-node in the circuit and it is adjacent to all its closed neighbours. Closed neighbours of A in G are A, B, C. The or-node corresponds to node A is just below the node A in the circuit and it is adjacent to A, B, C. The graph G has a dominating set $\{B, D\}$ of size two. We assign value 1 to the nodes B, D in the circuit and rest of the nodes get value 0. Conversely, assigning values 1 to A and F and 0 to rest of the vertices, gives the output 1 in the graph, therefore $\{A, F\}$ form a dominating set in the given graph. It is easy to verify that G has a dominating set of size 2 if and only if the circuit C has a satisfiable instance of weight 2.



Chapter 4

Main Results

Approximation algorithms and parameterized algorithms are two ways of dealing with NP-hard problems. Approximation algorithms are algorithms that find approximate solutions to NP-hard problems. While designing approximation algorithms, our goal is to design a polynomial time algorithm such that the solution it returns is not necessarily optimal, but probably close to optimal solution. The existance of efficient and deterministic algorithms for NP-hard or NP-complete problems is considered unlikely. However, some NP-hard and NP-complete problems can be solved by algorithms that are exponential in the size of a fixed parameter while polynomial in the size of the input. Such problems are called fixed parameters tractable (FPT). The primary goal of parameterized complexity is to understand the qualitative difference between fixed-parameter tractable problems, and problems that do not admit such efficient algorithms.

4.1 Minimum k Union Problem

We give formal definition of Minimum k-Union problem:

Definition 4.1.1. (Minimum K-Union problem (MkU)) In the MkU problem, we are given a universe $U = \{1, 2, ..., n\}$ of n elements and a collection of sets $\mathcal{S} \subseteq 2^U$, as well as an integer $k \leq |\mathcal{S}|$. The goal is to return a collection $T \subseteq \mathcal{S}$ with $|T| = k$ in order to minimize $\bigcup_{S \in T} S$.

Theorem 4.1.2. *[7] The MkU problem is NP-hard.*

The MkU problem is equivalent to SSBVE (Small Set Bipartite Vertex Expansion Problem). In SSBVE, we are asked to select k vertices from one partition of the bipartite graph such that their vertex expansion on other part is minimum. As this problem has received less attention, the only known approximation algorithm is of approximation ratio $O(\sqrt{n})$ where n is the input size due to Chlamtac et. al. APPROX 16. The other problems closest to MkU problems are Set Cover and Maximum Coverage. These two problems have been studied in the realm of parameterized complexity.

We now recall the notion of parameterized reduction. If we can find a parameterized reduction from CLIQUE or some other problem X , then we can say that X cannot have an FPT algorithm unless $FPT = W[1]$.

We now prove that MkU is at least as hard as CLIQUE and INDEPENDENT SET on regular graphs.

Definition 4.1.3. (CLIQUE problem on regular graph) Given a regular graph G and a positive integer k, does the graph G contain a clique of size at least k?

Theorem 4.1.4. *There is a parameterized reduction from CLIQUE on regular graphs to MkU problem.*

Proof: Let (G, k) be an instance of CLIQUE where G be a r-regular graph, $r > k$. We construct an instance of MkU problem, the following way:

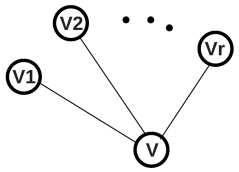
Let $G = (V, E)$ and the set of vertices $V = \{1, 2, ..., n\}$. Set $U = V = \{1, 2, ..., n\}$ and $S_i = N[i]$, the closed neighbourhood of the vertex i and $\mathcal{S} = \{S_1, S_2, ..., S_n\}$. Clearly, $|S_i| = r + 1$.

To prove the correction of the reduction, we prove that G has a k-clique if and only if there is a collection $T \subseteq \mathcal{S}$ with $|T| = k$ such that $|\bigcup_{S \in T} S| \leq rk + 2k - k^2$.

If G has a clique $C = \{v_1, v_2, ..., v_k\}$ of size k, then considering the sets correspond to the vertices in C, we get $|\bigcup_{i=1}^{k} S_{v_i}| \leq rk + 2k - k^2$. Note that C $\subseteq S_{v_i}$ for all i and there are atmost $(r + 1 - k)$ elements in each $S_{v_i} \setminus C$. Therefore, $\bigcup_{i=1}^{k} |(S_{v_i} \setminus C)| \leq k(r + 1 - k)$. Hence, $|\bigcup_{i=1}^{k} S_{v_i}| \leq \bigcup_{i=1}^{k} |(S_{v_i} \setminus C)| + |C| \leq k(r + 1 - k) + k = rk + 2k - k^2$.

Conversely, suppose we have a collection $T \subseteq \mathcal{S}$ with $|T| = k$ and $|\bigcup_{S \in T} S| = rk + 2k - k^2$ then we get a clique of size k in G. Consider a set $S_v = \{v_1, v_2, ..., v_r\}$ in T. The graph

corresponds to S_v, denoted by $G[S_v]$, is shown below:



The graph corresponds to T, denoted by $G[T]$, is union of the graphs $G[S_v]$, $S_v \in T$. Clearly, $G[T]$ is a subgraph of the graph G. The number of vertices of $G[T]$ is $rk + 2k - k^2$ and there are at least k vertices of degree r each. We claim that k of these vertices of degree r form a k-clique.

Definition 4.1.5. (Independent Set on Regular Graph) Given a regular graph G and a positive integer k, does the graph G contain a Independent Set of size at least k?

Theorem 4.1.6. *There is a parameterized reduction from* `Independent Set` *on regular graphs to MkU problem.*

Proof: Let (G, k) be an instance of `Independent Set`, where G is an r-regular graph. We construct the instance of MkU problem the following way: Let $G = (V, E)$ where $V = \{1, 2, .., n\}$ and $E = \{e_1, e_2, ..., e_m\}$. We set $U = \{e_1, e_2, ..., e_m\}$ and $S_i = \{e_l | e_l$ is incident on the vertex i $\}$, the set of edges incident on the vertex i; and $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$. Clearly, $|S_i| = r$ for all i as G is a r-regular graph. We claim that G has an independent set of size k if and only if $(U, \mathcal{S}, k, rk)$ is a yes-instance of Minimum k Union.

If G has an independent set $L = \{v_1, v_2, ..., v_k\}$ of size k, then every edge of G appears in at most one of the sets $S_{v_1}, S_{v_2}, ..., S_{v_k}$. That is, the sets $S_{v_1}, S_{v_2}, ..., S_{v_k}$ are pairwise disjoint. Hence $|\bigcup_{i=1}^{k} S_{v_i}| = rk$.

Conversely, suppose there is a collection of k sets $S_{u_1}, S_{u_2}, ..., S_{u_k} \in \mathcal{S}$ such that $|\bigcup_{i=1}^{k} S_{u_i}| = rk$. As $|S_{u_i}| = r$ for all i and $|\bigcup_{i=1}^{k} S_{u_i}| = rk$, this implies S_{u_i} are pairwise disjoint. Thus, the set $\{u_1, u_2, ..., u_k\}$ form an independent set of size k in G.

The following theorem follows from Theorem 4.1.6.

Theorem 4.1.7. *The MkU problem is* $W[1]$*-hard.*

4.2 Hall Set on C_4-free bipartite graph

Let G be an undirected graph. Then G is said to be C_4 free if it does not contain cycles of length 4. Here we study the Hall Set on C_4 free bipartite graphs and prove that the problem is also W[1]-hard similar to Hall Set on general bipartite graph. We recall the definitions of Hall Set problem and Clique problem.

Definition 4.2.1. (Hall Set) In a bipartite graph G with bipartite classes (A, B), set S is a Hall set in G if $S \subseteq A$ and $|N(S)| < |S|$.

Definition 4.2.2. (Hall Set on C_4-free bipartite graphs) Given a C_4-free bipartite graph G with bipartite classes (A, B) and a positive integer k, does there exists a set $S \subseteq A$ of size at most k such that $|N(S)| < |S|$?

We now prove that Hall Set problem on C_4-free bipartite graphs is at least as hard as Clique problem.

Theorem 4.2.3. *There is a parameterized reduction from Clique to Hall Set on C_4-free bipartite graph.*

Proof. Let (G, k) be an instance of CLIQUE where $G = (V, E)$. We construct an instance of Hall Set problem on C_4-free bipartite graph, the following way:

Given a graph $G = (V, E)$, we construct a bipartite graph H with bipartition A, B, where $A = A_1 \cup A_2$ and $B = B_1 \cup B_2$. The vertices in A_1 are the edges of G; so A_1 has $|E(G)|$ vertices. The vertices in B_1 are the vertices of G, so B_1 contains $|V(G)|$ vertices. Vertex $uv \in A_1$ is adjacent to vertices u and v in B_1. Additionally, corresponds to each vertex $uv \in A_1$, we add a vertex $u'v'$ in B_2. That is,

$$B_2 = \{u'v' \ : \ uv \in A_1\} \text{ and } |B_2| = |A_1| = |E(G)|.$$

We make $uv \in A_1$ adjacent to $u'v' \in B_2$. Finally we add $k+1$ isolated vertices in A_2. Thus, $|A| = |E(G)| + k + 1$ and $|B| = |V(G)| + |E(G)|$. Note that H is C_4 free.

We claim that there exists a Hall set of size $\binom{k}{2} + k + 1$ in H if and only if there exists a k-clique in G.

Suppose H has a Hall Set S of size $\binom{k}{2} + k + 1$. The size of neighbourhood of the Hall Set is less than the size of the Hall Set. Hence, $\binom{k}{2} + k + 1 > |N(S)|$.

We choose $k+1$ isolated vertices of A_2 (as they do not have any neighbours) and remaining $\binom{k}{2}$ vertices from A_1 in the Hall Set S. That is,

$$S = A_2 \cup S_1 \qquad \text{where } S_1 \subseteq A_1 \text{ and } |S_1| = \binom{k}{2}.$$

Note that S_1 has exactly $\binom{k}{2}$ neighbours in B_2 and at least k neighbours in B_1; so S_1 has at least $\binom{k}{2} + k$ neighbours in B. Hence $|N(S)| \geq \binom{k}{2} + k$. As $\binom{k}{2} + k + 1 > |N(S)| \geq \binom{k}{2} + k$, only possible value of $|N(S)|$ is $\binom{k}{2} + k$. Thus, S_1 has exactly k neighbours $\{v_1, v_2, \ldots, v_k\}$ in B_1. These k vertices $\{v_1, v_2, \ldots, v_k\}$ are adjacent to $\binom{k}{2}$ vertices in A_1. Therefore, these k vertices $\{v_1, v_2, \ldots, v_k\}$ forms a k-clique in graph G.

Conversely, suppose there is a k-Clique in G. We construct a Hall set S of size $\binom{k}{2} + k + 1$ as follows. We include in S, $\binom{k}{2}$ vertices of A_1 correspond to $\binom{k}{2}$ edges of the k-clique. We also include in S another r vertices from A_1 and remaining $k + 1 - r$ vertices from A_2 for some positive integer r. It is easy to verify that S has $\binom{k}{2} + k + r$ neighbours. For S to be Hall Set, $\binom{k}{2} + k + 1 > \binom{k}{2} + k + r$. This implies $r = 0$. Thus $\binom{k}{2}$ vertices of A_1 correspond to $\binom{k}{2}$ edges of the k-clique and $k + 1$ vertices of A_2 for a Hall Set of size $\binom{k}{2} + k + 1$ in H.

The following theorem follows from Theorem 4.2.3.

Theorem 4.2.4. *The Hall Set problem in C_4 free graph is $W[1]$-hard.*

4.3 Minimum Neighbourhood Problem on General Graph

Definition 4.3.1. (Minimum Neighbourhood Problem) We are given a graph $G = (V, E)$ with n vertices and two positive integers $k \leq n$ and l. The goal is to choose a set $S \subseteq V$ of size k from the graph G such that $|N_G[S]| \leq l$.

We now prove that Minimum Neighbourhood Problem is at least as hard as Minimum-k-Union Problem.

Definition 4.3.2. (Minimum-k-Union Problem) Given a family of sets $\mathcal{S}$ over a Universe U and positive integers k and l, the goal is to find out whether there exists k sets in $\mathcal{S}$ such that their union is at most l

Theorem 4.3.3. *There is a parameterized reduction from Minimum-k-Union to Minimum Neighbourhood Problem in general graph.*

Proof. Let $(U, \mathcal{S}, k, l)$ be an instance of Minimum-k-Union problem. We construct an instance $(G, k, k+l)$ of Minimum Neighbourhood problem, the following way: Let $U = \{u_1, u_2, \ldots, u_n\}$ and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$. In graph G, we

- add a vertex s_i corresponds to each set S_i of $\mathcal{S}$; set $V_1 = \{s_1, s_2, \ldots, s_m\}$.
- add a vertex u_j corresponds to each element u_j of U; set $V_2 = \{u_1, u_2, \ldots, u_n\}$
- if an element $u_j \in S_i$, then make the vertex s_i adjacent to u_j.
- additionally, add $l+k$ vertices to the graph G and make them adjacent to each u_j; set $V_3 = \{x_1, x_2, \ldots, x_{k+l}\}$.

The vertex set of G, $V = V_1 \cup V_2 \cup V_3$. We claim that there exists a Minimum-k-Union of size l in (U, F, k, l) if and only if there exists a Minimum-k-Neighbourhood of size at most $l+k$ in G.

Suppose there is a collection of k sets $S_{u_1}, S_{u_2}, ..., S_{u_k} \in \mathcal{S}$ such that $|\bigcup_{i=1}^{k} S_{u_i}| = l$. We choose the vertices $s_{u_1}, s_{u_2}, ..., s_{u_k}$ correspond to $S_{u_1}, S_{u_2}, ..., S_{u_k}$. As the size of the union of these k sets is l, the closed neighbourhood of $s_{u_1}, s_{u_2}, ..., s_{u_k}$ will contain $s_{u_1}, s_{u_2}, ..., s_{u_k}$ and l vertices u, where $u \in \bigcup_{i=1}^{k} S_{u_i}$. Hence, the size of the closed neighbourhood of $\{s_{u_1}, s_{u_2}, ..., s_{u_k}\}$ in G is $k+l$.

Conversely, suppose there is a collection $L \subseteq V(G)$ of k many vertices in G that have a closed neighbourhood of size at most $k+l$. L cannot contain any vertex from V_2 as each vertex in V_2 has $k+l+1$ closed neighbours. L cannot contain any vertex from V_3 as each vertex of V_3 has $|U|+1$ closed neighbours. Thus $L \subseteq V_1$ and let $L = \{s_{u_1}, \ldots, s_{u_k}\}$. We consider the k sets $S_{u_1}, S_{u_2}, ..., S_{u_k}$ correspond to these k vertices. As L has $k+l$ closed neighbours, $|\bigcup_{i=1}^{k} S_{u_i}| = l$.

We proved that there is a parameterized reduction from CLIQUE on regular graphs to MkU problem and there is a parameterized reduction from MkU to Minimum Neighbourhood Problem in general graph. Thus, there is a parameterized reduction from CLIQUE on regular graphs to Minimum Neighbourhood Problem in general graph. Therefore, we have the following theorem:

Theorem 4.3.4. *The Minimum Neighbourhood Problem in general graph is W[1]-hard.*

Chapter 5

Conclusions

This dissertation looks into three different variants of the Minimum k Union Problem in the realm of Parameterized Complexity. In Section 4.1 we showed that the Minimum k Union Problem in general graph is W[1]-hard by giving parameterized reductions from Clique and Independent Set, and hence we do not expect an FPT algorithm for the same (unless FPT=W[1]). We also showed that the Hall Set Problem in C_4-free graph is W[1]-hard by giving a reduction from the Clique and hence no FPT algorithm is possible in this case too. Moreover, in Minimum Neighbourhood Problem, we give a reduction from MkU Problem and show that it is too W[1]-hard.

Bibliography

[1] R. Diestel, *Graph Theory* 5th ed., Springer-Verlag Berlin Heidelberg, 2017.

[2] J.A. Bondy and U.S.R. Murthy, *Graph Theory*, Springer, 2017.

[3] R.G. Downey and M.R. Fellows, *Parameterized Complexity*, Springer, 1997.

[4] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dniel Marx, Marcin Pilipczuk, Micha Pilipczuk and Saket Saurabh, *Parameterized Algorithms*, Springer, 2016.

[5] Markus Blser, Computing small partial coverings, *Information Processing Letters*, 85, 327-331, 2003.

[6] H. L. Bodlaender, Dynamic programming algorithms on graphs with bounded tree-width. In the proceedings of the 15th International Colloquium on Automata, Languages and Programming (ICALP), Lecture Notes in Computer Science, 317, 105-117, 1988.

[7] Eden Chlamtac, Michael Dinitz, Yury Makarychev, Minimizing the Union: Tight Approximations for Small Set Bipartite Vertex Expansion. *SODA*, 2017.

[8] Omid Amini, Fedor V. Fomin, Saket Saurabh, Parameterized Algorithms for Partial Cover Problems arXiv:0802.1722 [cs.DS]