2
Preliminaries

In this chapter, we discuss some preliminaries we use in this thesis. We start our discussion with terms related to parameterized complexity. Next we elaborate some graph theoretic terms. Later, we look for some concepts related to matroids. In the end, we discuss about approximation algorithms and related terminology.

2.1 PARAMETERIZED COMPLEXITY

Consider a NP-complete language L'. As the problem whether NP=P or not is still unsolved, we don't know of any polynomial time algorithm for general instance of L'. Thus, all the known algorithms take exponential time in input length. The goal of Parameterized Complexity is to find ways of solving NP-hard problems more efficiently than brute force. Here, we assign a parameter to each instance of the language that is hopefully much smaller than the input size. Our goal is to restrict the time complexity to combinatorial explosion to the parameter instead of input length. Formally,

Definition 2.1.1. *A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is finite alphabet set. The second component is called the parameter of the problem.*

For example, consider the following problem mentioned in Chapter 1 Section 1.3.

IMPOSSIBLE PAIR CONTAINED PATH (IPP)
Input: A program flow graph $G(V,E)$, single source node s and single sink node t. The set V also contains n pairs of nodes $(a_i,b_i), 1 \leq i \leq n$. Thus $V = \{s,t,a_1,b_1,\ldots a_n,b_n\}$
Question: Does there exist a path (with no vertex repetition) in G from s to t such that the path contains exactly one node from each pair.

We may define the parameterized version of the problem as follows.

PARAMETERIZED IMPOSSIBLE PAIR CONTAINED PATH (LIPP)
Input: A program flow graph $G(V,E)$, single source node s and single sink node t. The set V also contains n pairs of nodes $(a_i,b_i), 1 \leq i \leq n$. Thus $V = \{s,t,a_1,b_1,\ldots a_n,b_n\}$. A positive integer k.
Question: Does there exist a k length path (with no vertex repetition) in G from s to t such that the path contains at most one node from each pair.

Here, the path length k is the parameter. We would like to point that most of the definitions in this section can be found in book [21] unless otherwise stated.

Now we discuss the properties of parameterized problem that is *fixed-parameter tractable (*FPT*)*.

Definition 2.1.2. *A parameterized problem L is* fixed-parameter tractable (FPT) *if there is an algorithm that solves the problem in time $f(k) \cdot |I|^{\mathcal{O}(1)}$, where $|I|$ is the size of the input instance, k is parameter associated with input instance and f is an arbitrary computable function depending only on the parameter k. Such an algorithm is called an* FPT *algorithm and such a running time is called* FPT *running time. The complexity class containing all fixed parameter tractable problems is called* FPT*.*

Now let us introduce the notion of parameterized reductions.

Definition 2.1.3. *Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A parameterized reduction from A to B is an algorithm that, given an instance (x,k) of A, outputs an instance (x',k') of B such that*

1. *(x,k) is a* Yes*-instance of A if and only if (x',k') is* Yes*-instance of B.*
2. *$k' \leq g(k)$ for some computable function g.*
3. *the running time of algorithm is $f(k).|x|^{\mathbb{O}(1)}$ for some computable function f.*

We have the following theorem based on parameterized reductions.

Lemma 2.1.1 ([21])**.** *If there is a parameterized reduction from A to B and B is* FPT*, then A is* FPT *as well.*

We next discuss *W hierarchy*. For this, first we define mixed Boolean circuits.

Definition 2.1.4. *A Boolean circuit is of mixed type if it consists of circuits having gates of the following kinds.*

1. *Small gates: NOT gates, AND gates and OR gates with bounded fan-in. We will usually assume that the bound on fan-in is 2 for AND gates and OR gates, and 1 for NOT gates.*
2. *Large gates: AND gates and OR gates with unrestricted fan-in.*

We assume the circuits considered here has single output node which has outdegree 0. Next we define $height$, $weight$ and $weft$ of a circuit.

Definition 2.1.5. *The depth of a circuit C is defined to be the maximum number of gates (small or large), not counting NOT gates, on an input-output path in C. The $weft$ of a circuit C is the maximum number of large gates on an input-output path in C. The weight of an assignment to the circuit is the number of input gates that receives value 1.*

Now we define WEIGHTED CIRCUIT SATISFIABILITY (WCS) problem.

Definition 2.1.6. *In the* WEIGHTED CIRCUIT SATISFIABILITY (WCS) *problem, we are given a circuit C and an integer k, the task is to decide if C has a satisfying assignment of weight exactly k.*

Let $\mathscr{C}$ is a class of circuits. Assume $\mathscr{C}_{t,d}$ to be the class of circuits with weft at most t and depth at most d. We define WCS$[\mathscr{C}]$ to be the problem WCS where the input circuit C belongs to $\mathscr{C}$.

Definition 2.1.7. *For $t \geq 1$, a parameterized language L belongs to the class $W[t]$ if there is a parameterized reduction from L to $WCS[\mathscr{C}_{t,d}]$ for some $d \geq 1$.*

For more literature on *W hierachy* and various *W[1]-complete* and *W[2]-complete* problems, refer [21]. Next we define *kernel* of parameterized problem.

Definition 2.1.8. *A parameterized problem L, is said to admit an $f(k)$ kernel if there is an* FPT *time algorithm which given an input (x,k) outputs (x',k') such that $|x| \leq f(k)$ and $k' \leq g(k)$ for some computable functions f and g of k. The function $f(k)$ is called the size of the kernel.*

We have following theorem.

Theorem 2.1.1. *A parameterized problem L has an* FPT *algorithm with a running if and only if it admits a kernel.*

Next we define the notion of parameterized minimization problem as defined in [60].

Definition 2.1.9 ([60])**.** *A parameterized minimization problem Π is a computable function*

$$\Pi : \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}.$$

The *instances* of a parameterized minimization problem Π are pairs $(I,k) \in \Sigma^* \times \mathbb{N}$, and a *solution* to (I,k) is simply a string $s \in \Sigma^*$, such that $|s| \leq |I| + k$. The *value* of the solution s is $\Pi(I,k,s)$. Just as for "classical" optimization problems the instances of Π are given as input, and the algorithmic task is to find a solution with the best possible value, where best means minimum for minimization problems.

Definition 2.1.10 ([60])**.** *For a parameterized minimization problem Π, the* optimum value *of an instance $(I,k) \in \Sigma^* \times \mathbb{N}$ is $OPT_\Pi(I,k) = \min_{\substack{s\in\Sigma^* \\ |s|\leq|I|+k}} \Pi(I,k,s)$. For an instance (I,k) of a parameterized minimization problem Π, an* optimal solution *is a solution s such that $\Pi(I,k,s) = OPT_\Pi(I,k)$.*

Definition 2.1.11 ([60])**.** *Let $\alpha \geq 1$ be constant. A fixed parameter tractable α-approximation algorithm for a parameterized minimization problem Π is an algorithm that takes as input an instance (I,k), runs in time $f(k)|I|^{\mathcal{O}(1)}$, and outputs a solution s such that $\Pi(I,k,s) \leq \alpha \cdot OPT(I,k)$.*

Note that Definition 2.1.11 only defines constant factor FPT-approximation algorithms. The definition can in a natural way, be extended to approximation algorithms whose approximation ratio depends on the parameter k, on the instance I, or on both. The parameterized minimization version of Set Cover (SC)[1] can be defined as follows.

$$\mathsf{SC}(((U,\mathscr{F}),k),\mathscr{F}') = \begin{cases} \infty & \text{if } \mathscr{F}' \text{ is not a set cover of } (U,\mathscr{F}) \\ \min\{|\mathscr{F}'|, k+1\} & \text{otherwise} \end{cases}$$

One can similarly define parameterized minimization version of other problems such as Graphical CF-SC and Matroidal CF-SC. For more background, the reader is referred to the following books [31, 24, 21].

2.2 GRAPHS

Given a graph $G(V,E)$, $V(G)$ and $E(G)$ denote its vertex-set and edge-set, respectively. For a graph $G(V,E)$, we define $G[V']$ to be the graph induced on vertex set $V' \subseteq V$. We use standard notation and terminology from the book of Diestel [23] for graph-related terms which are not explicitly defined here.

Now we define some graph classes that we use later in thesis.

Low Density Graphs and Cluster Graphs. A graph G is a called a cluster graph, if each connected component of G is a clique.

Definition 2.2.1 ([43])**.** *A set of objects $\mathscr{A}$ in $\mathbb{R}^d$ (not necessarily convex or connected) has density ρ if any object f (not necessarily in $\mathscr{A}$) intersects at most ρ objects in $\mathscr{A}$ with diameter greater than or equal to the diameter of f. The minimum such quantity is called the density of $\mathscr{A}$. If ρ is a constant, then $\mathscr{A}$ has low density.*

[1] We refer readers to page number 15, paragraph titled "**Capping the objective function at $k+1$**" in [60] for the explanation of capping the objective function to $k+1$ in the parameterized approximation.

Definition 2.2.2 ([43]). *For $\alpha > 0$, an object $g \subseteq \mathbb{R}^d$ is α-fat if for any ball b with a center inside g, that does not contain g in its interior, we have $vol(b \cap g) \geq \alpha \cdot vol(b)$. A set $\mathscr{A}$ of objects is fat if all its members are α-fat for some constant α.*

Now we define the arboricity of undirected graph.

Definition 2.2.3. *The arboricity of an undirected graph is the minimum number of forests into which its edges can be partitioned. Thus a graph G is said to have* arboricity d *if the edges of G can be partitioned into at most d forests.*

Let $\mathscr{G}_d$ denote the family of graphs of arboricity d. This family includes the family of intersection graphs of low density objects in low dimensional Euclidean space as explained in [42, 43]. Specifically, this includes planar graphs, graphs excluding a fixed graph as a minor, graphs of bounded expansion, and graphs of bounded degeneracy. In most applications, conflict graphs themselves belong to a family of geometric graphs. Har-Peled and Quanrud [42, 43] showed that low-density geometric objects form a subclass of the class of graphs that have polynomial expansion, which in turn, is contained in the class of graphs of bounded arboricity.

2.3 APPROXIMATION ALGORITHMS

As the problem whether NP=P or not is still unresolved, we don't yet have polynomial time algorithms for NP-complete problems. One of the approaches to resolve this issue is relaxing the need to optimally solve these problems, and instead solve them efficiently with provable guarantees on the distance of the returned solution to the optimal one. This leads us to the concept of approximation. Before introducing approximation, let us first formally define NP optimization problems. Most of the definitions in this section are reffered from [19] and [20].
An NP optimization problem Π is a quadruple $(I, f, m, type)$ where

1. I is a set of instances and is recognizable in polynomial time.
2. Given an instance $x \in I$, $f(x)$ is the set of feasible solutions such that for all $y \in f(x)$, size of y is polynomially bounded in size of x. Also, for any y with size bounded by polynomial in size of $x \in I$, whether $y \in f(x)$ can be determined in polynomial time.
3. Given an instance x and a feasible solution $y \in f(x)$, $m(x,y)$ denotes the measure of y, which is usually a positive real. Also m is polynomially computable function.
4. $type \in \{max, min\}$.

The goal of an NP optimization problem with respect to an instance $x \in I$ is to find an optimum solution, that is, a feasible solution y such that

$$m(x,y) = type\{m(x,y') | y' \in f(x)\}.$$

For example, consider the following problem

GRAPHICAL CONFLICT FREE SET COVER (GRAPHICAL CF-SC)
Input: A universe U of size n, a family $\mathscr{F}$ of size m of subsets of U and a conflict graph $CG_{\mathscr{F}}$.
Question: Find a set a set cover (if exists) $\mathscr{F}' \subseteq \mathscr{F}$ of minimum size such that $CG_{\mathscr{F}}[\mathscr{F}']$ is an independent set?

Here $type = min$ and m is the size of set cover $C \subseteq \mathscr{F}$ such that $CG_{\mathscr{F}}[C]$ is an independent set. For any $x \in I_{\Pi}$, The class NPO is the set of all NP optimization problems.

Let Π be an NPO problem. Given an instance x and a feasible solution y of x and $opt(x)$ is measure of optimal solution of x, we define the performance ratio of y with respect to x as

$$R(x,y) = \max\left\{\frac{m(x,y)}{opt(x)}, \frac{opt(x)}{m(x,y)}\right\}$$

The performance ratio is always a number greater than or equal to 1 and is as close to 1 as the value of the feasible solution is close to the optimum value.

We now define approximation algorithm.

Definition 2.3.1. *Let Π be an NPO problem and let $\mathscr{A}$ be an algorithm that, for any instance x of Π, returns a feasible solution $\mathscr{A}(x)$ in polynomial time. Given a rational $r > 1$, we say that $\mathscr{A}$ is an r-approximation algorithm for Π if the performance ratio of the feasible solution $\mathscr{A}(x)$ with respect to x verifies the following inequality:*

$$R(x, \mathscr{A}(x)) \leq r.$$

Now let us define the class APX.

Definition 2.3.2. *An NPO problem Π belongs to the class APX if exists an r-approximation algorithm $\mathscr{A}$ for Π, for some real $r > 1$.*

One can also define a family of complexity classes $f(n)$-APX, where $f(n)$-APX contains problems with a polynomial time approximation algorithm having $\mathscr{O}(f(n))$ approximation ratio.

Another important class in approximation is *polynomial time approximation scheme* (PTAS) which is defined as follows.

Definition 2.3.3. *A* polynomial time approximation scheme *(PTAS) for a minimization problem is an algorithm $\mathscr{A}$ that takes an input instance, a constant $\varepsilon > 0$, and returns a solution* SOL *such that* $\text{SOL} \leq (1+\varepsilon)\text{OPT}$*, where* OPT *is the optimal value, and the running time of $\mathscr{A}$ is $n^{\mathscr{O}(f(\frac{1}{\varepsilon}))}$, for some computable function f depending only on ε.*

Now let us look at approximation-preserving reductions.

Approximation-Preserving Reductions

Given two optimization problems Π and Π', an approximation-preserving reduction is a pair of functions (f,g), such that:

1. f maps an instance x of Π to an instance x' of Π'. Also, f is polynomial time computable.
2. g maps a solution y' of Π' to a solution y of Π. Here, g is polynomial time computable.
3. g preserves approximation ratio.

There are various approximation reductions, few of which we discuss next. Let Π and Π' be two optimization problems. Also let x be an instance of Π. The function f maps x to an instance of Π' denoted as $f(x)$ in polynomial time in length length of x. Assume $sol(x)$ and $sol(f(x))$ denotes the set of feasible solution to x and $f(x)$, respectively. Assume $y \in sol(f(x))$. We have m and m' as measure function of Π and Π', respectively. and y be an instance of Π'. For the general scheme, refer Figure 2.1.

Using these notations, we define L-reduction as follows.

Definition 2.3.4. *A reduction (f,g) is said to be L-reduction from $\Pi(I_\Pi, f_\Pi, m, type)$ to $\Pi'(I_{\Pi'}, f_{\Pi'}, m', type)$ if there exists two positive constants α and β such that*

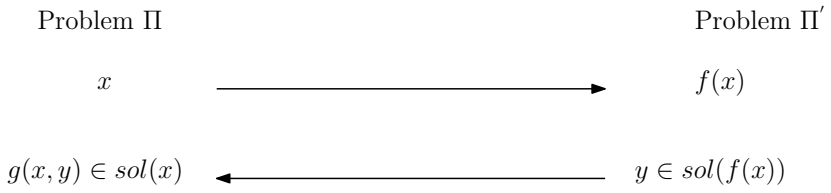

Figure 2.1 : Approximation reduction scheme

1. *For any $x \in I_\Pi$, $opt(f(x)) \leq \alpha opt(x)$ where $opt()$ gives the optimal solution to the instance.*

2. *For any $x \in I_\Pi$ and for any $y \in sol(f(x))$, $|opt(x) - m(x, g(x,y))| \leq \beta |opt(f(x)) - m'(f(x), y)|$.*

Now let us define PTAS-reduction. Here, assume $R_P(,)$ to be the performance ratio corresponding to to any NP optimization problem P.

Definition 2.3.5. *A reduction (f, g, δ) is said to be PTAS-reduction from $\Pi(I_\Pi, f_\Pi, m, type)$ to $\Pi'(I_{\Pi'}, f_{\Pi'}, m', type)$ for three computable functions f, g, δ if*

1. *For any $x \in I_\Pi$ and for any $\varepsilon \in (0,1)$, $f(x, \varepsilon) \in I_{\Pi'}$ is computable in time polynomial with respect to $|x|$.*

2. *For any $x \in I_\Pi$, for any $\varepsilon \in (0,1)$ and for any $y \in sol(f(x, \varepsilon))$, $g(x, y, \varepsilon) \in sol(x)$ is computable in time polynomial in $|x|$ and $|y|$.*

3. *For any $x \in_\Pi$, for any $\varepsilon \in (0,1)$ and for any $y \in sol(f(x, \varepsilon))$*

$$R_{\Pi'}(f(x,\varepsilon), y) \geq \delta(\varepsilon) \text{ implies } R_\Pi(x, g(x, y, \varepsilon)) \geq \varepsilon.$$

We use PTAS-reduction to define class APX-complete as follows.

Definition 2.3.6. *An NPO problem Π in APX is APX-complete if, for any other problem Π' in APX, Π' is PTAS-reducible to Π.*

We introduce few more terms that we use in this thesis in next section.

2.4 FEW OTHER TERMINOLOGIES

For a positive integer t, we use $[t]$ as a shorthand for $\{1, 2, \ldots, t\}$. Given a function $f : D \to R$ and a subset $D' \subseteq D$, let $f|_{D'}$ denote the restriction of the function f to the domain D'. and we say that $f|_{D'}$ is injective if for any $x, y \in D'$, $f(x) \neq f(y)$. A family of sets $\mathscr{A}$ is called a p-family, if the cardinality of all the sets in $\mathscr{A}$ is p. Given two families of sets $\mathscr{A}$ and $\mathscr{B}$, we define $\mathscr{A} \bullet \mathscr{B} = \{X \cup Y \mid X \in \mathscr{A} \text{ and } Y \in \mathscr{B} \text{ and } X \cap Y = \emptyset\}$. Throughout the thesis we use ω to denote the exponent in the running time of matrix multiplication, the current best known bound for ω is < 2.373 [79]. We use e to denote the base of natural logarithm.