GRAPH SEPARATION IN PARAMETERIZED ALGORITHMS

By

M. S. Ramanujan

THE INSTITUTE OF MATHEMATICAL SCIENCES, CHENNAI.

A thesis submitted to the
Board of Studies in Mathematical Sciences

In partial fulfillment of the requirements

For the Degree of

Master of Science

of

HOMI BHABHA NATIONAL INSTITUTE

April 2011

CERTIFICATE

Certified that the work contained in the thesis entitled GRAPH SEPARATION IN PARAMETERIZED PROBLEMS, by M. S. Ramanujan, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Saket Saurabh

Theoretical Computer Science Group,

The Institute of Mathematical Sciences, Chennai

ACKNOWLEDGEMENTS

I am deeply indebted to my thesis advisor Professor Saket Saurabh for his invaluable guidance, not to mention his infectious energy, over the course of the last two years.

A lot of thanks are also due to Professor Venkatesh Raman for sharing his ideas and suggestions throughout the course of my work during this thesis. I am extremely grateful to him and Professors V. Arvind, Kamal Lodaya, C. R. Subramaniam, R. Ramanujam and Meena Mahajan for believing in me, and for granting me an opportunity to learn from their vast knowledge and experience.

I will always be grateful to Professor K. Muralikrishnan for his guidance during my undergraduate days and for introducing me to the exciting world of Parameterized Complexity.

Special thanks go to Geevarghese Philip and Neeldhara Misra for the hours of discussion in which I learnt a lot, and also to Somnath Sikdar, for the discussions we had when I visited him.

I am truly grateful to Aishwarya, Yadu, Karteek, Nitin, Pranabendu, Mrinal, Sreejith, Praveen, and Kunal for helping me out at one time or another in matters relating to science or otherwise.

Finally, the two people I have to thank the most are my parents, who have always encouraged me and supported me in every way possible.

Abstract

Parameterized Complexity is an exact algorithmic approach to deal with intractable computational problems having some small parameters. For decision problems with input size n, and a parameter k, the goal here is to design an algorithm with runtime $f(k)n^{\mathcal{O}(1)}$ where f is an arbitrary function of k, as opposed to (in most cases,) a trivial $n^{k+O(1)}$ algorithm. Such algorithms are called fixed parameter tractable (FPT) and problems with FPT algorithms are said to be fixed parameter tractable (FPT). Such algorithms are practical when small values of the parameters cover practical ranges.

Numerous computational problems have been solved by modelling these problems as problems on graphs and then applying known results in graph theory to extract some structure and use this in designing algorithms. In recent years, a more refined approach has been to model the problems of interest as *separation* problems on graphs and then solve them. This approach has been used to give FPT algorithms for a number of problems including MULTIWAY CUT, DIRECTED FEEDBACK VERTEX SET (DFVS) and ALMOST 2-SAT, with the last two being long standing open questions in the field of parameterized complexity. It is not only the case that a lot of problems have been modelled as separation problems on graphs, but these separation problems themselves appear to have a common underlying combinatorial structure.

The aim of this thesis is to study a combinatorial object called *important separators*, and formally describe it as a tool with which a number of graph separation problems with certain properties can be solved. To this end, we first extend the existing notion of *important separators*, which has already been defined on undirected graphs, to directed graphs as well. Finally, we describe the FPT algorithms for the above three problems using the notion of *important separators* as an explicit tool.

Contents

List of Figures

List of Algorithms

1 Introduction

A fundamental theorems about connectivity in finite undirected graphs is Menger's Theorem [34], which states the following.

Let G be a finite undirected graph and s and t be two nonadjacent vertices. Then, the minimum number of vertices whose removal disconnects s and t is equal to the maximum number of pairwise vertex disjoint paths from s to t.

There is a corresponding edge version of this theorem and also vertex and arc versions on directed graphs.

This *min-max* theorem is an extremely fundamental one in combinatorial optimization and the minimum $s-t$ cut can be computed in polynomial time [1]. Hence, some natural generalizations to the problem of finding the minimum set of vertices disconnecting two vertices include the problem of finding the minimum number of vertices disconnecting vertices of a given set from each other, and the problem of finding the minimum number of vertices disconnecting multiple specified pairs of vertices. It turns out that these problems are computationally intractable in the classical setting, and as a result, it makes sense to approach these problems from the relatively recent field of Parameterized Complexity.

These problems have been studied extensively and have been found to have quite a lot of underlying combinatorial structure. A lot of ways have been devised to exploit these structures, and in this thesis, we will study one such approach. The approach we study uses a combinatorial object called *important separators* to prove the tractability of these problems in the area of parameterized complexity.

1.1 Organization of the Thesis

Chapter 1.2 contains basic Notations and Definitions we will follow in this Thesis. In Chapter 2, we give a high level description of the field of Parameterized Complexity and the basic techniques involved in designing ixed parameter tractable algorithms. In Chapter 3, we first describe the notion of Important Vertex Separators in undirected graphs and give formal Definitions and proofs of some Lemmas and Theorems. We then extend these Definitions and Lemmas to Vertex and Arc Separators in directed graphs and set up the machinery for the application of the notion of Important Separators in parameterized problems which involve Graph Separation. In Subsequent Chapters, we consider the current best algorithms for the MULTIWAY CUT problem, the DIRECTED FEEDBACK VERTEX SET problem, and the ALMOST 2-SAT problem and describe these algorithms using the machinery we set up in the preceding Chapter. In Chapter 4 we describe the part played by Important Separators in the FPT algorithm for the Multiway Cut problem. In Chapter 5 we describe the FPT algorithm for the Feedback Vertex Set problem in Directed Graphs by using the concepts of vertex separators in directed graphs. Finally, in Chapter 6, we consider the Almost 2 SAT problem and describe an FPT algorithm for this problem using the notions of Important Arc Separators in Directed Graphs.

1.2 Notations, Definitions and Conventions

1.2.1 Basic Notations

We assume that the reader is familiar with basic notions like sets, functions, polynomials, relations, integers etc. In particular, for these notions we follow the same notations as that in [43].

1.2.2 Growth of Functions

We employ mainly the big-Oh ($\mathcal{O}$) notation (see [14]) and the big-Oh-star ($\mathcal{O}^*$) notation introduced in [50]. Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ be two functions from Natural numbers to Natural numbers. We say that $f(n) = \mathcal{O}(g(n))$ if there

exist constants c, and n_0 such that for all $n \geq n_0$, $f(n) \leq c.g(n)$. The notation $\mathcal{O}^*$ notation is essentially the big-Oh notation which hides polynomial factors and hence is used only for exponential time algorithms. We use $\mathcal{O}^*(f(n))$ to denote $\mathcal{O}(f(n).n^c)$ where c is some constant. In this thesis, we will use the $\mathcal{O}^*$ notation to hide factors polynomial in input size in order to focus on the function of the parameter. Hence, for us, $\mathcal{O}^*(f(k))$ denotes $\mathcal{O}(f(k).n^c)$ where k is the value of some parameter, n is the size of the input instance, and c is some constant.

1.2.3 Graphs

Undirected Graphs An *undirected* graph G is a pair (V, E) where V and E are unordered sets. The elements of V are called vertices of G. E consists of unordered pairs of vertices and elements of E are called edges of G. A vertex u and a vertex v are said to be adjacent if E contains the pair (u, v). The edge (u, v) is said to be incident on the vertices u and v, while u and v are called the endpoints of the edge (u, v). An undirected graph G is called a *simple undirected* graph if there is no edge in E of the form (v, v) where v is a vertex of G. In this thesis, unless explicitly mentioned otherwise, the graphs we consider are all simple undirected graphs. The open neighborhood or just neighborhood of a vertex v in the graph G is the set $N(v) = \{u|(u, v) \in E\}$ and by closed neighborhood of a vertex v, we mean the set $N[v] = \{v\} \cup N(v)$. Let S be a set of vertices of G. We denote by (open) neighborhood of S, the set $N(S) = (\bigcup_{v \in S} N(v)) \setminus S$ and by closed neighborhood of S we denote the set $N(S) \cup S$.

A *walk* in the graph G is a sequence $W = v_1, \ldots, v_t$ of vertices such that $(v_i, v_{i+1}) \in E$ for every $1 \leq i \leq t - 1$ and it is called a walk from v_i to v_t in G. The *length* of this walk is $t - 1$. A walk in which any vertex occurs at most once is called a *path*. A walk where the first vertex is same as the last vertex and all the other vertices are distinct is called a *cycle*. The walks $v_i, v_{i+1}, \ldots, v_j$, $1 \leq i \leq t$, $i \leq j \leq t$ are called subwalks of the walk W. If W is a path these walks are called subpaths of W. A graph is called acyclic if it does not contain a cycle.

A vertex u is said to be *reachable* from a vertex v in G, if there is a path from u to v in G. The graph G is said to be *connected* if there is a path between every pair of vertices in G.

A set S of vertices (or edges) with some property is said to be a maximal set

with that property if there is no set S' of vertices (respectively edges) such that $S' \supset S$ and S' has the same property.

A set S of vertices (or edges) with some property is said to be a minimal set with that property if there is no set S' of vertices (respectively edges) such that $S' \subset S$ and S' has the same property.

A connected component of G is an induced subgraph $X = G[C]$ of G such that C is a maximal subset of V such that $G[C]$ is connected.

A connected acyclic graph is called a tree and a graph whose connected components are trees is called a forest.

The line graph $L(G)$ of G is a graph where there is a vertex for every edge of G and two vertices of $L(G)$ are adjacent if and only if their corresponding edges are adjacent in G.

Given a graph G, we use $\mu(G)$ and $\beta(G)$ to denote, respectively, the size of a maximum matching and a minimum vertex cover. A graph $G = (V, E)$ is said to be *König* if $\beta(G) = \mu(G)$.

Directed Graphs A directed graph (or digraph) D is a pair (V, A) where V and A are sets. The elements of V are called vertices of D. The set A consists of ordered pairs of vertices and elements of A are called arcs of D. The arc (u, v) is said to be incident on the vertices u and v, while u and v are called the endpoints of the arc (u, v). The undirected graph obtained from D by ignoring the ordering among the pairs constituting the arcs, is called the *underlying* undirected graph of D. A directed graph D is called a *simple directed* graph if there is no edge in A of the form (v, v) where v is a vertex of D. The out neighborhood of a vertex u in the digraph D is the set $N^+(u) = \{v|(u, v) \in A\}$ and the in neighborhood of u is the set $N^-(u) = \{v|(v, u) \in A\}$. Let S be a set of vertices of D. We denote by out neighborhood of S, the set $N^+(S) = (\bigcup_{v \in S} N^+(v)) \setminus S$ and by in neighborhood of S we denote the set $N^-(S) = (\bigcup_{v \in S} N^-(v)) \setminus S$.

A *walk* in the digraph D is a sequence $W = v_1, \ldots, v_t$ of vertices such that $(v_i, v_{i+1}) \in A$ for every $1 \leq i \leq t - 1$ and it is called a walk from v_i to v_t in D. The *length* of this walk is $t-1$. A walk in which any vertex occurs at most once is called a *path*. A walk where the first vertex is same as the last vertex and all the other vertices are distinct is called a *cycle*. The walks $v_i, v_{i+1}, \ldots, v_j$, $1 \leq i \leq t$, $i \leq j \leq t$ are called subwalks of the walk W. If W is a path these walks are called

subpaths of W.

A vertex u is said to be *reachable* from a vertex v in G, if there is a path from u to v in G. The graph G is said to be *strongly connected* if there is a path between every pair of vertices in G.

A graph $D' = (V', A')$ is called a subgraph of D if $V' \subseteq V$ and $A' \subseteq A$. We say D' is a subgraph of D induced by the vertex set $S \subseteq V$ if $V' = S$ and $A' = \{(u, v) \in A | u, v \in S\}$ and denote it by $D[S]$. We say that D' is a subgraph of D induced by the arc set $S \subseteq A$ if $A' = S$ and $V' = \{v \in V |$ there is an arc of M incident on $v\}$ and denote it by $D[S]$.

A set S of vertices (or arcs) with some property is said to be a maximal set with that property if there is no set S' of vertices (respectively arcs) such that $S' \supset S$ and S' has the same property.

A set S of vertices (or arcs) with some property is said to be a minimal set with that property if there is no set S' of vertices (respectively arcs) such that $S' \subset S$ and S' has the same property.

Planar Graphs A *planar* graph is an undirected graph that can be embedded in the plane, that is, it can be drawn on the plane in such a way that its edges intersect only at their endpoints. In other words, it can be drawn in such a way that no edges cross each other. A planar graph already drawn in the plane without edge intersections is called a plane graph or a planar embedding of the graph. A directed graph is called a planar digraph if the underlying undirected graph is a planar graph.

2

Parameterized Complexity

In this Chapter we give a broad overview of the field of Parameterized Complexity.

2.1 Introduction

Definition 2.1.1. ([43]) *A* parameterized problem *is a language* $L \subseteq \Sigma^* \times \mathbb{N}$*, where* Σ *is a finite alphabet. The second component is called the* parameter *of the problem.*

Definition 2.1.2. ([43]) *A parameterized problem* L *is said to be* Fixed Parameter Tractable *(FPT) if it can be determined in time* $f(k).n^{\mathcal{O}(1)}$ *whether or not* $(x,k) \in L$*, where* f *is a computable function depending only on* k *and* $n = |(x,k)|$*. The complexity class containing all fixed parameter tractable problems is called FPT.*

Unless specified otherwise, we will assume that the parameter k is a non negative integer encoded with a unary alphabet.

Definition 2.1.3. *Let* $L_1, L_2 \subseteq \Sigma^* \times \mathbb{N}$ *be two parameterized problems. A parameterized (many one) reduction from* L_1 *is a triple* (f, f_1, f_2) *where* $f_1, f_2 : \mathbb{N} \to \mathbb{N}$ *and* $f : \Sigma^* \times \mathbb{N} \to \Sigma^*$ *such that*

1. $f(x,k)$ *is computable in time* $f_1(k).|(x,k)|^{\mathcal{O}(1)}$ *and*
2. $(x,k) \in L_1$ *if and only if* $(f(x,k), f_2(k)) \in L_2$.

The parameterized reduction (f, f_1, f_2) *is said to be a parameter preserving parameterized reduction if* $f_2(k) = k$.

2.2 Some Standard Techniques for Designing FPT Algorithms

2.2.1 Bounded Search Trees

The method of bounded search trees is one of the earliest known and simplest techniques used in the design of FPT algorithms. The main idea behind this technique is to find in polynomial time, a *small* set of elements such that at least one of them must occur in any feasible solution to the problem. Then we guess an element of this set to be in the solution we are trying to construct, process the input instance so that it reflects our choice of this element, make the necessary changes to the parameter value, and recursively solve the problem on the resulting instance. The small set of elements identified is usually a forbidden structure which needs to be removed and is usually of constant size, although the size of this set can be as large as logarithmic in the size of the input instance. As long as the depth of the recursion tree is bounded by some function of the parameter, since the time taken at each node of the recursion tree is polynomial, this results in an FPT algorithm. There is an easy $\mathcal{O}^*(2^k)$ algorithm for VERTEX COVER [41, 20] using this method.

2.2.2 Iterative Compression

Iterative Compression is a useful technique for designing FPT algorithms for minimization problems. This technique was first introduced in [45] to solve the ODD CYCLE TRANSVERSAL problem, where we are interested in finding a set of at most k vertices such that after removing these vertices, the resulting graph is bipartite. This method was also used in obtaining FPT algorithms for EDGE BIPARTIZATION, CHORDAL DELETION, CLUSTER VERTEX DELETION and FVS on undirected graphs [28, 39, 32, 17, 10]. This technique was also used by Chen et al. [13] to show that the DFVS problem is FPT, and by Razgon and Barry O'Sullivan [44] to show that ALMOST 2-SAT is FPT, two long standing open problems in the area of Parameterized Complexity. This technique is also used to desing exact exponential algorithms (see [22]) and for other results based on this method, we refer the reader to a survey articles [29, 33].

We will give a sketch of this method as applied to a graph problem. The central idea here is to design an FPT algorithm which, when given a $k+1-$sized solution for a problem, either compresses it to a solution of size at most k or proves that there is no solution of size at most k. This is known as the compression step of the algorithm. The method adopted usually is to begin with a subgraph that trivially admits a $(k+1)-$sized solution and then expand it iteratively. In any iteration, we try find a compressed ($k-$sized) solution for the instance corresponding to the current subgraph. If we find such a solution, we use this solution and (usually) the vertex or edge we add to get the next subgraph, to get a $(k+1)-$sized solution for this instance and start the next iteration. We stop when we either get a solution of size k for the entire graph, or if some intermediate instance turns out to be incompressible. In order to stop when some intermediate instance turns out to be incompressible, the problem must have the property that the solution size in any subgraph is at most the solution size in the whole graph.

2.2.3 Kernelization

Kernelization is a polynomial time algorithm that preprocesses instances of problems and returns equivalent instances with a guaranteed upper bound on the size of the output, which is called the kernel of the instance. Kernelization is usually achieved by applying a set of reduction rules that allow us to remove or ignore parts of the instance that are easy to handle.

Kernelization has received increasing interest over the last decade, maturing from a technique to prove fixed parameter tractability, into a stand alone field of research. In recent years there has been many kernelization results for a variety of problems. Some notable examples are the linear vertex kernel for VERTEX COVER by Chen et al. [11], the quadratic kernel for FVS in undirected graphs by Thomasse [49], and a polynomial kernel for MULTICUT in trees due to Bousquet et al. [7]. For further results we refer the reader to the survey articles [30, 6].

2.2.4 Color Coding

The technique of Color Coding was introduced by Alon et al [2] to handle the problem of detecting a $k-$sized subgraph of constant treewidth in an input graph in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$. The color coding technique usually applies when we the objective is to find a small graph (whose size is typically the parameter) in some larger graph as a subgraph. The idea behind this method is to randomly color the entire graph with a set of colors with the number of colors chosen in a way that if the smaller graph does exist in this graph as a subgraph, with high probability it will be colored in a way that we can find it *efficiently*. This randomized algorithm can subsequently be derandomized using hash families.

2.2.5 Dynamic Programming

Many problems that are NP-complete on general graphs are known to be polynomial time solvable on trees. This is mainly because the subtrees rooted at the children of any node in a (rooted) tree are disjoint and hence most types of information collected individually about these subtrees can be merged to obtain the relevant information regarding the subtree rooted at the node under consideration. Hence a standard dynamic programming approach is usually sufficient to solve a lot of these problems on trees. This has served as motivation to devise decompositions of graphs in a tree *like* manner where most of the separation properties involved in trees are replicated on sets of vertices rather than individual vertices. The quantitiy *treewidth* has been defined to formalize and quantify the closeness of a graph to a tree and quite a number of FPT algorithms have been devised with this quantity as the parameter (see [5, 46]).

3

Important Separators

The notion of important separators was formally introduced in [38] to handle the MULTIWAY CUT problem and the same concept was used implicitly in [12] to give an improved algorithm for the same problem. Chen et al. [13] used this idea to resolve the fixed parameter tractability of the DIRECTED FEEDBACK VERTEX SET problem and Razgon and Barry O' Sullivan [44] proved the fixed parameter tractability of the ALMOST 2-SAT problem using this concept.

Organization of the Chapter In Section 3.1 we give a brief description of the intuition behind the concept of important separators. In Section 3.2 we present the definition of important vertex separators in undirected graphs, some basic lemmata stemming from these definitions, and give an upper bound on the number of important separators of bounded size. In Sections 3.3 and 3.4 we extend the definitions and lemmata in Section 3.2 to analogous versions for vertex and arc separators respectively in directed graphs. In Section 3.5 we show that the bound on the number of important separators given in each of the above three Sections is essentially tight. Finally, in Section 3.6, we present an algorithm to enumerate important vertex separators in undirected graphs, and give analogous algorithms to enumerate important vertex and arc separators in directed graphs.

3.1 Motivation

Given a graph $G = (V, E)$ disjoint vertex sets X and Y, suppose we are asked to find a minimum set of vertices in G whose removal disconnects X from Y and

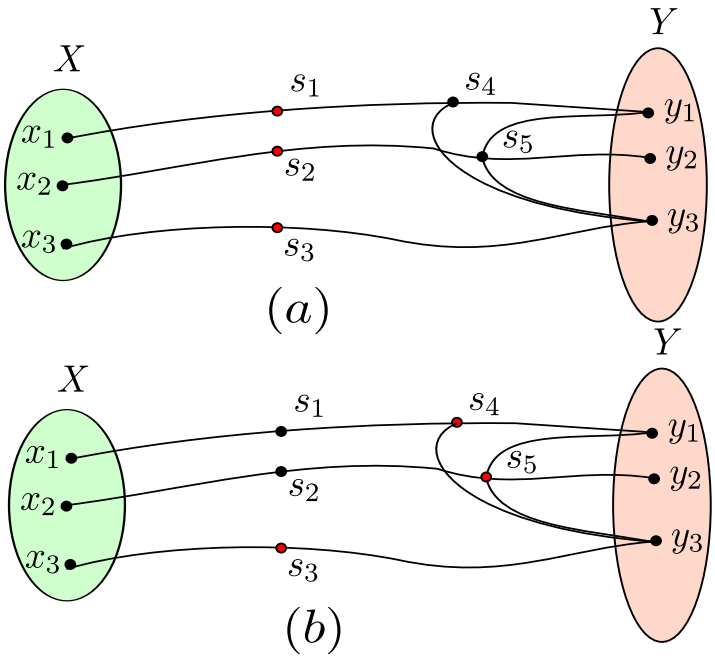

Figure 3.1: $S_1 = \{s_1, s_2, s_3\}$ and $S_2 = \{s_3, s_4, s_5\}$ are two $X - Y$ vertex separators. But S_2 also separates vertices of Y while S_1 does not.

also disconnects every pair of vertices in Y. In such cases, if chosen carefully, the set of vertices which we choose to separate X from Y will also help us in separating some pairs of vertices in Y. Intuitively, "the closer it is to Y, the better is the chance of it separating vertices in Y" and hence some separators seem to be more *important* than others. The sets $S_1 = \{s_1, s_2, s_3\}$ and $S_2 = \{s_3, s_4, s_5\}$ (see Fig. 3.1) are sets of the same size which separate X and Y. But by our intuition, S_2 is more *important* for us since it also separates the vertices in Y. This intuition was formalized in [38] and used to give an FPT algorithm for the MULTIWAY CUT problem.

3.2 Important Vertex Separators in Undirected Graphs

Definition 3.2.1. *Let $G = (V, E)$ be an undirected graph and let $X \subseteq V$. We denote by $\delta(X)$ the vertices of $G \setminus X$ which have a neighbor in X. We define the function $\tilde{f} : 2^V \to \mathbb{N}$ as $\tilde{f}(X) = |\delta(X)|$.*

Definition 3.2.2. *Let $G = (V, E)$ be an undirected graph, let $X \subseteq V$ and $S \subseteq V \setminus X$. We denote by $R_G(X, S)$ the set of vertices of G reachable from X in $G \setminus S$. We drop the subscript G if it is clear from the context.*

Definition 3.2.3. *Let Z be a finite set. A function $f : 2^Z \to \mathbb{R}$ is submodular if for all subsets A and B of Z, $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$.*

Lemma 3.2.4. *Let $G = (V, E)$ be an undirected graph and let $\tilde{f} : 2^V \to \mathbb{N}$ be a function defined as above. Then the function $\tilde{f}$ is submodular.*

Proof. Let $A, B \subseteq V$. We prove the submodularity of $\tilde{f}$ by showing that for every vertex v of the graph the contribution of v to $\tilde{f}(A \cup B) + \tilde{f}(A \cap B)$ is at most its contribution to $\tilde{f}(A) + \tilde{f}(B)$.

(i) $v \notin A \cup B$

(a) v has no neighbors in $A \cup B$. Then v contributes 0 to each of the four terms.

(b) v has a neighbor in A but none in B. In this case, v contributes 1 to $\tilde{f}(A)$, 1 to $\tilde{f}(A \cup B)$ and 0 to the rest.

(c) v has a neighbor in B but none in A. In this case v contributes 1 to $\tilde{f}(B)$, 1 to $\tilde{f}(A \cup B)$ and 0 to the rest.

(d) v has a neighbor in $A \cap B$. In this case v contributes 1 each to $\tilde{f}(A), \tilde{f}(B), \tilde{f}(A \cap B)$, and $\tilde{f}(A \cup B)$.

(ii) $v \in B \setminus A$

(a) v has no neighbors in A. Then v contributes 0 to each of the four terms.

(b) v has a neighbor in A. Then v contributes 1 to $\tilde{f}(A)$, at most 1 to $\tilde{f}(A \cap B)$, and 0 to the rest.

(iii) $v \in A \setminus B$

(a) v has no neighbors in B. Then v contributes 0 to each of the four terms.

(b) v has a neighbor in B. Then v contributes 1 to $\tilde{f}(B)$ and at most 1 to $tilde f(A \cap B)$ and 0 to the rest.

(iv) $v \in A \cap B$. Then, v contributes 0 to each of the four terms. □

Definition 3.2.5. ([38]) *Let $G = (V, E)$ be an undirected graph and let $X, Y \subset V$ be two disjoint vertex sets. A subset $S \subseteq V \setminus (X \cup Y)$ is called an $X - Y$ vertex separator in G if $R_G(X, S) \cap Y = \phi$ or in other words there is no path from X to Y in the graph $G \setminus S$. We denote by $\lambda_G(X, Y)$ the size of the smallest $X - Y$ vertex separator in G. An $X - Y$ separator S_1 is said to **dominate** an $X - Y$ separator S with respect to X if $|S_1| \leq |S|$ and $R(X, S_1) \supset R(X, S)$. If the set X is clear from the context, we just say that S_1 dominates S. An $X - Y$ vertex separator is said to be inclusionwise minimal if none of its proper subsets is an $X - Y$ vertex separator.*

Proposition 3.2.6. *If $R \supseteq X$ is any vertex set disjoint from Y such that $\delta(R) \cap Y = \phi$ then $\delta(R)$ is an $X - Y$ vertex separator.*

Proof. This is because any path from X to Y in G must contain a vertex of $\delta(R)$. Consider a path P from $u \in X$ to $v \in Y$ in G. Since $u \in R$ and $v \notin R$, P must contain a vertex w which is outside R and is neighbor to a vertex in R implying that $w \in \delta(R)$. □

Definition 3.2.7. ([38]) *Let $G = (V, E)$ be an undirected graph, $X, Y \subset V$ be disjoint vertex sets and $S \subseteq V \setminus (X \cup Y)$ be an $X - Y$ vertex separator in G. We say that S is an **important** $X - Y$ vertex separator if it is inclusionwise minimal and there does not exist another $X - Y$ vertex separator S_1 such that S_1 dominates S with respect to X. If $S \subset V$ is an important $X - Y$ vertex separator then the set $R(X, S)$ is called an **important set** and the subgraph $G[R(X, S)]$ is called an **important component** if it is connected.*

Lemma 3.2.8. ([38]) *Let $G = (V, E)$ be an undirected graph, $X, Y \subset V$ be disjoint vertex sets. There exists a unique important $X - Y$ vertex separator S^* of size $\lambda_G(X, Y)$.*

Proof. Consider a minimum size $X - Y$ vertex separator of size $\lambda_G(X, Y)$. Since it is minimal, this separator is either important or there is another that dominates it. Hence, there is at least one important $X - Y$ vertex separator of size $\lambda_G(X, Y)$. Now we show that there cannot be two such important $X - Y$ vertex separators.

Suppose S_1 and S_2 are two important $X-Y$ vertex separators of size $\lambda_G(X,Y)$ where $S_1 \neq S_2$ and let $R_1 = R(X,S_1)$ and $R_2 = R(X,S_2)$. We know that $R_1, R_2 \supset X$, and by the minimality of S_1 and S_2, $\delta(R_1) = S_1$ and $\delta(R_2) = S_2$. But $S_1, S_2 \cap Y = \phi$. Hence by Proposition 3.2.6 the sets $\delta(R_1 \cup R_2)$ and $\delta(R_1 \cap R_2)$ are also $X-Y$ vertex separators and hence $\tilde{f}(R_1 \cup R_2), \tilde{f}(R_1 \cap R_2) \geq \lambda_G(X,Y)$. By the submodularity of $\tilde{f}$ (Lemma 3.2.4), we have that

$$\underbrace{\tilde{f}(R_1)}_{=\lambda_G(X,Y)} + \underbrace{\tilde{f}(R_2)}_{=\lambda_G(X,Y)} \geq \underbrace{\tilde{f}(R_1 \cup R_2)}_{\geq\lambda_G(X,Y)} + \underbrace{\tilde{f}(R_1 \cap R_2)}_{\geq\lambda_G(X,Y)}$$

which implies that $\tilde{f}(R_1 \cup R_2) = \lambda_G(X,Y)$. But this contradicts our assumption that S_1 and S_2 were important $X-Y$ vertex separators since $\delta(R_1 \cup R_2)$ is an $X-Y$ vertex separator which dominates both S_1 and S_2. □

Note: In the future we will continue to refer to the unique smallest important $X-Y$ vertex separator as S^* without explicit reference to Lemma 3.2.8.

Lemma 3.2.9. ([40]) *Let $G = (V,E)$ be an undirected graph, $X, Y \subset V$ be disjoint vertex sets and let S be an important $X-Y$ vertex separator. Then $R(X,S) \supseteq R(X,S^*)$.*

Proof. Suppose that this is not the case and let $R_1 = R(X,S)$ and $R_2 = R(X,S^*)$ where $S \neq S^*$. We know that $R_1, R_2 \supset X$ and the minimality of S and S^* implies that $\delta(R_1) = S$ and $\delta(R_2) = S^*$. But $S, S^* \cap Y = \phi$. Hence, by Proposition 3.2.6, the sets $\delta(R_1 \cup R_2)$ and $\delta(R_1 \cap R_2)$ are also $X-Y$ vertex separators and hence $\tilde{f}(R_1 \cup R_2), \tilde{f}(R_1 \cap R_2) \geq \lambda_G(X,Y)$. By the submodularity of $\tilde{f}$ (Lemma 3.2.4) we have that

$$\tilde{f}(R_1) + \underbrace{\tilde{f}(R_2)}_{=\lambda_G(X,Y)} \geq \tilde{f}(R_1 \cup R_2) + \underbrace{\tilde{f}(R_1 \cap R_2)}_{\geq\lambda_G(X,Y)}$$

which implies that $\tilde{f}(R_1 \cup R_2) \leq \tilde{f}(R_1)$. But this contradicts our assumption that S was an important $X-Y$ vertex separator since $\delta(R_1 \cup R_2)$ is an $X-Y$ vertex separator which dominates S. □

Lemma 3.2.10. *Let $G = (V, E)$ be an undirected graph, $X, Y \subset V$ be disjoint vertex sets and S be an important $X - Y$ vertex separator.*

(a) S is a $\{v\} - Y$ vertex separator for every $v \in R(X, S)$.

(b) For every $v \in S$, $S \setminus \{v\}$ is an important $X - Y$ vertex separator in $G \setminus \{v\}$.

(c) If S is an $X' - Y$ vertex separator for some $X' \supset X$ such that $G[X']$ is connected, then S is also an important $X' - Y$ vertex separator.

Proof. 1. Suppose this were not the case. Since $v \in R(X, S)$ there is a path from X to v in $G \setminus S$. Now since there is also a path from v to Y, this implies the existence of a path from X to Y in $G \setminus S$. But, this is not possible since S is an $X - Y$ vertex separator.

2. Suppose $S' = S \setminus \{v\}$ is not an important $X - Y$ vertex separator in $G' = G \setminus \{v\}$. Then there is an $X - Y$ vertex separator S_1 in G' which dominates S' in G'. Consider the set $S_2 = S_1 \cup \{v\}$. Observe that S_2 is also an $X - Y$ vertex separator in G. This is because any path from X to Y which does not contain v exists in G' and hence must contain a vertex of S_2. Now, since S_1 dominates S' in G', S_2 dominates S in G which contradicts our assumption that S is an important $X - Y$ vertex separator.

3. Assume that this is not the case. Since S is a minimal $X - Y$ vertex separator, S is also a minimal $X' - Y$ vertex separator. Therefore, if S is not an important $X' - Y$ separator it must be the case that there is an $X' - Y$ vertex separator S_1 which dominates S with respect to X'. We will show that S_1 also dominates S with respect to X, which contradicts our assumption that S is an important $X - Y$ vertex separator. Clearly, $|S_1| \leq |S|$. Hence it is enough for us to show that $R(X, S) \subset R(X, S_1)$.

   First we prove that $R(X, S) \subseteq R(X, S_1)$. Consider a vertex v in $R(X, S)$. Clearly $R(X', S) \supseteq R(X, S)$, which means that $v \in R(X', S)$ and since S_1 dominates S with respect to X', v is in $R(X', S_1)$. Since $G[X']$ is connected and contains X, the vertices reachable from X in $G \setminus S_1$ and those reachable from X' in $G \setminus S_1$ are the same implying that $v \in R(X, S_1)$.

Now consider some vertex $u \in S \setminus S_1$. By the minimality of S, u has a neighbor w in $R(X,S)$. But w is also in $R(X,S_1)$ which implies that $u \in R(X,S_1)$ and hence $R(X,S) \subset R(X,S_1)$. □

The following lemma is implicit in [12].

Lemma 3.2.11. ([12]) *Let $G = (V,E)$ be an undirected graph, $X,Y \subset V$ be disjoint vertex sets of G. For every $k \geq 0$ there are at most 4^k important $X-Y$ vertex separators of size at most k.*

Proof. Given $G,X,Y,k \geq 0$ we define a measure $\mu(G,X,Y,k) = 2k - \lambda_G(X,Y)$. We prove by induction on $\mu(G,X,Y,K)$ that there are at most $2^{\mu(G,X,Y,k)}$ important $X-Y$ vertex separators of size at most k.

For the base case, if $2k - \lambda_G(X,Y) < k$ then $\lambda_G(X,Y) > k$ and hence the number of important separators of size at most k is 0. If $\lambda_G(X,Y) = 0$, it means that there is no path from X to Y and hence the empty set alone is the important $X-Y$ vertex separator. For the induction step, consider $G,X,Y,k \geq 0$ such that $\mu = \mu(G,X,Y,k) \geq k, \lambda_G(X,Y) > 0$ and assume that the statement of the Lemma holds for all G',X',Y',k' where $\mu(G',X',Y',k') < \mu$.

By Lemma 3.2.8, there is a unique important $X-Y$ vertex separator S^* of size $\lambda_G(X,Y)$. Since we have assumed $\lambda_G(X,Y)$ to be positive, S^* is non empty. Consider a vertex $v \in S^*$. Any important $X-Y$ vertex separator S either contains v or does not contain v. For any important $X-Y$ vertex separator S which contains v, $S \setminus \{v\}$ is an important $X-Y$ vertex separator in $G \setminus \{v\}$ (Lemma 3.2.10(b)). Hence the number of important $X-Y$ vertex separators containing v, of size at most k in G is at most the number of important $X-Y$ vertex separators of size at most $k-1$ in $G \setminus \{v\}$. Observe that $\lambda_{G\setminus\{v\}}(X,Y) = \lambda_G(X,Y) - 1$ which implies that $\mu(G \setminus \{v\}, X, Y, k-1) < \mu$ and by induction hypothesis, the number of important $X-Y$ vertex separators of size at most $k-1$ in $G \setminus \{v\}$ is bounded by $2^{\mu-1}$ which is also a bound on the number of important $X-Y$ vertex separators in G which have size at most k and contain v.

Now let S be an important $X-Y$ vertex separator of size at most k which does not contain v. By Lemma 3.2.9 we know that $R(X,S) \supseteq R(X,S^*)$ and by the minimality of S^*, v has a neighbor in $R(X,S)$ which implies that $R(X,S) \supseteq R(X,S^*) \cup \{v\}$. We now set $X' = R(X,S^*) \cup \{v\}$. By Lemma 3.2.10(c) we know

that S is a also an important $X' - Y$ vertex separator. Thus a bound on the number of important $X' - Y$ vertex separators of size at most k is also a bound on the number of important $X - Y$ vertex separators of size at most k which do not contain v. First note that $\lambda_G(X', Y) > \lambda_G(X, Y)$ since otherwise we would have an $X - Y$ vertex separator which dominates S^* with respect to X. Now, $\mu(G, X', Y, k) < \mu$ and by induction hypothesis, the number of important $X' - Y$ vertex separators of size at most k is bounded by $2^{\mu-1}$.

Summing up the bounds we get that the number of important $X - Y$ separators of size at most k is bounded by $2 \cdot 2^{\mu-1} = 2^{\mu} \leq 2^{2k}$. □

3.3 Important Vertex Separators in Directed Graphs

In this section, we define the notion of important vertex separators in directed graphs and prove some properties of these separators which are analogous to those seen in the previous section.

Definition 3.3.1. *Let $D = (V, A)$ be a directed graph and let $X \subseteq V$. We denote by $\delta^+(X)$ the out-neighbors of X. We define the function $\hat{f} : 2^V \to \mathbb{N}$ as $\hat{f}(X) = |\delta^+(X)|$.*

Definition 3.3.2. *Let $D = (V, A)$ be a directed graph, let $X \subseteq V$ and $S \subseteq V \setminus X$. We denote by $R_D(X, S)$ the set of vertices of D reachable from X in $D \setminus S$. We drop the explicit reference to D if it is clear from the context.*

Lemma 3.3.3. *Let $D = (V, A)$ be a directed graph and let $\hat{f} : 2^V \to \mathbb{N}$ be a function defined as above. Then the function $\hat{f}$ is submodular.*

Proof. Let $A, B \subseteq V$. We prove the submodularity of $\hat{f}$ by showing that for every vertex v of the graph the contribution of v to $\hat{f}(A \cup B) + \hat{f}(A \cap B)$ is at most its contribution to $\hat{f}(A) + \hat{f}(B)$.

(i) $v \notin A \cup B$

  (a) v has no in-neighbors in $A \cup B$. Then v contributes 0 to each of the four terms.

  (b) v has an in-neighbor in A but none in B. In this case, v contributes 1 to $\hat{f}(A)$, 1 to $\hat{f}(A \cup B)$ and 0 to the rest.

(c) v has an in-neighbor in B but none in A. In this case v contributes 1 to $\hat{f}(B)$, 1 to $\hat{f}(A \cup B)$ and 0 to the rest.

(d) v has an in-neighbor in $A \cap B$. In this case v contributes 1 each to $\hat{f}(A), \hat{f}(B), \hat{f}(A \cap B)$, and $\hat{f}(A \cup B)$.

(ii) $v \in B \setminus A$

(a) v has no in-neighbors in A. Then v contributes 0 to each of the four terms.

(b) v has an in-neighbor in A. Then v contributes 1 to $\hat{f}(A)$, at most 1 to $\hat{f}(A \cap B)$, and 0 to the rest.

(iii) $v \in A \setminus B$

(a) v has no in-neighbors in B. Then v contributes 0 to each of the four terms.

(b) v has an in-neighbor in B. Then v contributes 1 to $\hat{f}(B)$, at most 1 to $\hat{f}(A \cap B)$, and 0 to the rest.

(iv) $v \in A \cap B$. Then v contributes 0 to each of the four terms. □

Definition 3.3.4. *Let $D = (V, A)$ be a directed graph and let $X, Y \subset V$ be two disjoint vertex sets. A subset $S \subseteq V \setminus (X \cup Y)$ is called an $X - Y$ vertex separator in D if $R(X, S) \cap Y = \phi$ or in other words Y is not reachable from X in $D \setminus S$. We denote by $\lambda_D(X, Y)$ the size of the smallest $X - Y$ vertex separator in D. An $X - Y$ separator S_1 is said to **dominate** an $X - Y$ vertex separator S with respect to X if $|S_1| \leq |S|$ and $R(X, S_1) \supset R(X, S)$. If the set X is clear from the context we just say that S_1 dominates S. An $X - Y$ vertex separator is said to be inclusionwise minimal if none of its proper subsets is an $X - Y$ vertex separator.*

Proposition 3.3.5. *If $R \supseteq X$ is any vertex set disjoint from Y such that $\delta^+(R) \cap Y = \phi$ then $\delta^+(R)$ is an $X - Y$ vertex separator.*

Proof. We will prove that any path from X to Y in D must contain a vertex of $\delta^+(R)$. Consider a path P from $u \in X$ to $v \in Y$ in G. Since $u \in R$ and $v \notin R$, P must contain a vertex w which is outside R and is an out-neighbor to a vertex in R implying that $w \in \delta^+(R)$. □

Definition 3.3.6. *Let $D = (V, A)$ be a directed graph, $X, Y \subset V$ be disjoint vertex sets and $S \subseteq V \setminus (X \cup Y)$ be an $X - Y$ vertex separator in G. We say that S is an **important** $X - Y$ vertex separator if it is inclusionwise minimal and there does not exist another $X - Y$ vertex separator S_1 such that S_1 dominates S with respect to X. If $S \subset V$ is an important $X - Y$ vertex separator then the set $R(X, S)$ is called an **important set**.*

Lemma 3.3.7. *Let $D = (V, A)$ be a directed graph, $X, Y \subset V$ be disjoint vertex sets. There exists a unique important $X - Y$ vertex separator S^* of size $\lambda_D(X, Y)$.*

Proof. Consider an $X - Y$ vertex separator of size $\lambda_D(X, Y)$. This separator is either important or there is another that dominates it. Hence, there is at least one important $X - Y$ vertex separator of size $\lambda_D(X, Y)$. Now, we show that there cannot be two such important $X - Y$ vertex separators.

Suppose S_1 and S_2 are two important $X - Y$ vertex separators of size $\lambda_D(X, Y)$ where $S_1 \neq S_2$ and let $R_1 = R(X, S_1)$ and $R_2 = R(X, S_2)$. We know that $R_1, R_2 \supset X$, and by the minimality of S_1 and S_2, $\delta^+(R_1) = S_1$ and $\delta^+(R_2) = S_2$. But $S_1, S_2 \cap Y = \phi$. Hence by Proposition 3.3.5 the sets $\delta^+(R_1 \cup R_2)$ and $\delta^+(R_1 \cap R_2)$ are also $X - Y$ vertex separators and hence $\hat{f}(R_1 \cup R_2), \hat{f}(R_1 \cap R_2) \geq \lambda_D(X, Y)$. By the submodularity of $\hat{f}$ (Lemma 3.3.3) we have that

$$\underbrace{\hat{f}(R_1)}_{=\lambda_D(X,Y)} + \underbrace{\hat{f}(R_2)}_{=\lambda_D(X,Y)} \geq \underbrace{\hat{f}(R_1 \cup R_2)}_{\geq\lambda_D(X,Y)} + \underbrace{\hat{f}(R_1 \cap R_2)}_{\geq\lambda_D(X,Y)}$$

which implies that $\hat{f}(R_1 \cup R_2) = \lambda_D(X, Y)$. This contradicts our assumption that S_1 and S_2 were important $X - Y$ vertex separators as $\delta^+(R_1 \cup R_2)$ is an $X - Y$ vertex separator which dominates both S_1 and S_2. □

Lemma 3.3.8. *Let $D = (V, A)$ be a directed graph, $X, Y \subset V$ be disjoint vertex sets and let S be an important $X - Y$ vertex separator. Then $R(X, S) \supseteq R(X, S^*)$.*

Proof. Suppose that this is not the case and let $R_1 = R(X, S)$ and $R_2 = R(X, S^*)$ where $S \neq S^*$. We know that $R_1, R_2 \supset X$ and the minimality of S and S^* implies that $\delta^+(R_1) = S$ and $\delta^+(R_2) = S^*$. But $S, S^* \cap Y = \phi$. Hence by Proposition 3.3.5 the sets $\delta^+(R_1 \cup R_2)$ and $\delta^+(R_1 \cap R_2)$ are also $X - Y$ vertex separators and hence

$\hat{f}(R_1 \cup R_2), \hat{f}(R_1 \cap R_2) \geq \lambda_D(X,Y)$. By the submodularity of $\hat{f}$ (Lemma 3.2.4) we have that

$$\hat{f}(R_1) + \underbrace{\hat{f}(R_2)}_{=\lambda_D(X,Y)} \geq \hat{f}(R_1 \cup R_2) + \underbrace{\hat{f}(R_1 \cap R_2)}_{\geq \lambda_D(X,Y)}$$

which implies that $\hat{f}(R_1 \cup R_2) \leq \hat{f}(R_1)$. But this contradicts our assumption that S was an important $X - Y$ vertex separator since $\delta^+(R_1 \cup R_2)$ is an $X - Y$ vertex separator which dominates S. □

Lemma 3.3.9. *Let $D = (V, A)$ be a directed graph, $X, Y \subset V$ be disjoint vertex sets and S be an important $X - Y$ vertex separator.*

(a) For every $v \in R(X, S)$, S is a $\{v\} - Y$ vertex separator.

(b) For every $v \in S$, $S \setminus \{v\}$ is an important $X - Y$ vertex separator in $D \setminus \{v\}$.

(c) If S is an $X' - Y$ vertex separator for some $X' \supset X$ such that X' is reachable from X in the induced subgraph $D[X']$, then S is an important $X' - Y$ vertex separator.

Proof. 1. Suppose this were not the case. Since $v \in R(X, S)$ there is a path from X to v in $G \setminus S$. Now since there is also a path from v to Y, this results in a walk from X to Y which implies the existence of a path from X to Y in $G \setminus S$. But, this is not possible since S is an $X - Y$ vertex separator.

2. Suppose $S' = S \setminus \{v\}$ is not an important $X - Y$ vertex separator in $D' = D \setminus \{v\}$. Then, in D', there is an $X - Y$ vertex separator S_1 which dominates S'. Consider the set $S_2 = S_1 \cup \{v\}$. Observe that S_2 is also an $X - Y$ vertex separator in D. This is because any path from X to Y which does not use v exists in D' and hence must contain a vertex of S_2. Now, since S_1 dominates S' in D', S_2 dominates S in G which contradicts our assumption that S is an important $X - Y$ vertex separator.

3. Observe that in order to prove the statement of the Lemma, it is sufficient to prove that any $X' - Y$ vertex separator S_1 which dominates S with respect to X' also dominates S with respect to X. Consider such an $X' - Y$ vertex separator S_1 dominating S with respect to X'. We have that $R(X', S) \subset$

$R(X', S_1)$. Since X' is reachable from X in $D[X']$, $R(X', S) \subseteq R(X, S)$ and $R(X', S_1) \subseteq R(X, S_1)$. But we also know that $R(X', S) \supseteq R(X, S)$ and $R(X', S_1) \supseteq R(X, S_1)$. Hence, $R(X', S) = R(X, S)$ and $R(X', S_1) = R(X, S_1)$. But since S_1 dominates S with respect to X', $R(X', S) \subset R(X', S_1)$. This implies that $R(X, S) \subset R(X, S_1)$ and hence, S_1 also dominates S with respect to X. □

Lemma 3.3.10. *Let $D = (V, A)$ be an undirected graph, $X, Y \subset V$ be disjoint vertex sets of D. For every $k \geq 0$ there are at most 4^k important $X - Y$ vertex separators of size at most k.*

Proof. Given $D, X, Y, k \geq 0$ we define a measure $\mu(D, X, Y, k) = 2k - \lambda_D(X, Y)$. We prove by induction on $\mu(D, X, Y, K)$ that there are at most $2^{\mu(D,X,Y,k)}$ important $X - Y$ vertex separators of size at most k. For the base case, if $2k - \lambda_D(X, Y) < k$ then $\lambda_D(X, Y) > k$ and hence the number of important separators of size at most k is 0. If $\lambda_D(X, Y) = 0$, it means that there is no path from X to Y and hence the empty set alone is the important $X - Y$ vertex separator. Consider $D, X, Y, k \geq 0$ such that $\mu = \mu(D, X, Y, k) \geq k, \lambda_D(X, Y) > 0$ and assume that the statement holds for all D', X', Y', k' where $\mu(D', X', Y', k') < \mu$.

By Lemma 3.3.7 there is a unique important $X - Y$ vertex separator S^* of size $\lambda_D(X, Y)$. Since we have assumed $\lambda_D(X, Y)$ to be positive, S^* is non empty. Consider a vertex $v \in S^*$. Any important $X - Y$ vertex separator S either contains v or does not contain v. For any important $X - Y$ vertex separator S which contains v, $S \setminus \{v\}$ is an important $X - Y$ vertex separator in $D \setminus \{v\}$ by Lemma 3.3.9(b). Hence the number of important $X - Y$ vertex separators of size at most k in D which contain v, is at most the number of important $X - Y$ vertex separators of size at most $k - 1$ in $G \setminus \{v\}$. Observe $\lambda_{G\setminus\{v\}}(X, Y) = \lambda_D(X, Y) - 1$ which implies that $\mu(D \setminus \{v\}, X, Y, k - 1) < \mu$ and by induction hypothesis, the number of important $X - Y$ vertex separators of size at most $k - 1$ in $D \setminus \{v\}$ is bounded by $2^{\mu-1}$ which is also a bound on the number of important $X - Y$ vertex separators of size at most k in D which contain v.

Now let S be an important $X - Y$ vertex separator of size at most k which does not contain v. By Lemma 3.3.8 we know that $R(X, S) \supseteq R(X, S^*)$ and by the minimality of S^*, v has an in-neighbor in $R(X, S)$ which implies that $R(X, S) \supseteq R(X, S^*) \cup \{v\}$. We now set $X' = R(X, S^*) \cup \{v\}$ and by Lemma 3.3.9(c) S is an

important $X' - Y$ vertex separator. Thus a bound on the number of important $X' - Y$ vertex separators of size at most k is also a bound on the number of important $X - Y$ vertex separators of size at most k which do not contain v. Note that $\lambda_D(X', Y) > \lambda_D(X, Y)$, since otherwise we would have an $X - Y$ vertex separator which dominates S^*. Now, $\mu(D, X', Y, k) < \mu$ and by induction hypothesis, the number of important $X' - Y$ vertex separators of size at most k is bounded by $2^{\mu-1}$.

Summing up the bounds we get that the number of important $X - Y$ separators of size at most k is bounded by $2 \cdot 2^{\mu-1} = 2^{\mu} \leq 2^{2k}$. □

3.4 Important Arc Separators

In this section we further extend our current notion of separators to include arc separators in directed graphs.

Definition 3.4.1. *Let $D = (V, A)$ be a directed graph and let $X \subseteq V$. We denote by $\delta^+(X)$ the out-neighborhood of X and by $\partial^+(X)$ the set of arcs from X to $\delta^+(X)$. We define the function $\hat{f}_a : 2^V \to \mathbb{N}$ as $\hat{f}_a(X) = |\partial^+(X)|$.*

Definition 3.4.2. *Let $D = (V, A)$ be a directed graph, let $X \subseteq V$ and $S \subseteq A$. We denote by $R_D(X, S)$ the set of vertices of D reachable from X in $G \setminus S$. We drop the explicit reference to D if it is clear from the context.*

Lemma 3.4.3. *Let $D = (V, A)$ be an undirected graph and let $\hat{f}_a : 2^V \to \mathbb{N}$ be a function defined as above. Then the function $\hat{f}_a$ is submodular.*

Proof. Let $A, B \subseteq V$. We prove the submodularity of $\hat{f}_a$ by showing that for every arc e of the graph the contribution of e to $\hat{f}_a(A \cup B) + \hat{f}_a(A \cap B)$ is at most its contribution to $\hat{f}_a(A) + \hat{f}_a(B)$.

(a) e lies in $D[V \setminus (A \cup B)]$ or in $D[A \cap B]$. In this case, e contributes 0 to each of the four terms.

(b) e is an arc from $A \setminus B$ to $V \setminus (A \cup B)$. In this case, e contributes 1 to $\hat{f}_a(A)$, 1 to $\hat{f}_a(A \cup B)$ and 0 to the rest.

(c) e is an arc from $B \setminus A$ to $V \setminus (A \cup B)$. In this case e contributes 1 to $\hat{f}_a(B)$, 1 to $\hat{f}_a(A \cup B)$ and 0 to the rest.

(d) e is an arc from $A \cap B$ to $V \setminus (A \cup B)$. In this case e contributes 1 each to $\hat{f}_a(A), \hat{f}_a(B), \hat{f}_a(A \cap B)$, and $\hat{f}_a(A \cup B)$.

(e) e is an arc from $A \cap B$ to $A \setminus B$. In this case e contributes 1 each to $\hat{f}_a(B)$ and $\hat{f}_a(A \cap B)$ and 0 to the rest.

(f) e is an arc from $A \cap B$ to $B \setminus A$. In this case e contributes 1 each to $\hat{f}_a(A)$ and $\hat{f}_a(A \cap B)$ and 0 to the rest.

(g) e is an arc from $A \setminus B$ to $B \setminus A$. In this case e contributes 1 to $\hat{f}_a(A)$ and 0 to the rest.

(h) e is an arc from $B \setminus A$ to $A \setminus B$. In this case e contributes 1 to $\hat{f}_a(B)$ and 0 to the rest. □

Definition 3.4.4. *Let $D = (V, A)$ be a directed graph and let $X, Y \subset V$ be two disjoint vertex sets. A subset $S_a \subseteq A$ is called an $X - Y$ arc separator in D if $R(X, S_a) \cap Y = \phi$ or in other words Y is separated from X in $D \setminus S_a$. We denote by $\zeta_D(X, Y)$ the size of the smallest $X - Y$ arc separator in D. An $X - Y$ arc separator S_a^1 is said to **dominate** an $X - Y$ arc separator S_a with respect to X if $|S_a^1| \leq |S_a|$ and $R(X, S_a^1) \supset R(X, S_a)$. If the set X is clear from the context we just say that S_a^1 dominates S_a. An $X - Y$ arc separator is said to be inclusionwise minimal if none of its proper subsets is an $X - Y$ arc separator.*

Observation 3.4.5. *Observe that any inclusionwise minimal $X - Y$ arc separator does not contain an arc with both endpoints in X.*

Proposition 3.4.6. *If $R \supseteq X$ is any vertex set disjoint from Y then $\partial^+(R)$ is an $X - Y$ arc separator.*

Proof. We will prove that any path from X to Y in D must contain an arc in $\partial^+(R)$. Consider a path P from $u \in X$ to $v \in Y$ in G. Since $u \in R$ and $v \notin R$, P must contain a vertex w which is outside R and is an out-neighbor to a vertex z in R implying that the arc $(z, w) \in \partial^+(R)$. □

Definition 3.4.7. *Let $D = (V, A)$ be a directed graph, $X, Y \subset V$ be disjoint vertex sets and $S_a \subseteq A$ be an $X - Y$ arc separator in D. We say that S_a is an **important** $X - Y$ arc separator if it is inclusionwise minimal and there does not exist another*

$X - Y$ arc separator S_a^1 which dominates S_a with respect to X. If $S_a \subset A$ is an important $X - Y$ arc separator then the set $R(X, S_a)$ is called an ***important set****.*

Lemma 3.4.8. *Let $D = (V, A)$ be a directed graph, $X, Y \subset V$ be disjoint vertex sets. There exists a unique important $X - Y$ arc separator S_a^* of size $\zeta_D(X, Y)$.*

Proof. Consider an $X - Y$ arc separator of size $\zeta_D(X, Y)$. This separator is either important or there is another that dominates it. Hence, there is at least one important $X - Y$ arc separator of size $\zeta_D(X, Y)$. Now, let S_a^1 and S_a^2 be two important $X - Y$ arc separators of size $\zeta_D(X, Y)$ where $S_a^1 \neq S_a^2$ and let $R_1 = R(X, S_a^1)$ and $R_2 = R(X, S_a^2)$. We know that $R_1, R_2 \supset X$, and by the minimality of S_a^1 and S_a^2, $\partial^+(R_1) = S_a^1$ and $\partial^+(R_2) = S_a^2$. By Proposition 3.4.6 the sets $\partial^+(R_1 \cup R_2)$ and $\partial^+(R_1 \cap R_2)$ are also $X - Y$ vertex separators and hence $\hat{f}_a(R_1 \cup R_2), \hat{f}_a(R_1 \cap R_2) \geq \zeta_D(X, Y)$. By the submodularity of $\hat{f}_a$ (Lemma 3.4.3) we have that

$$\underbrace{\hat{f}_a(R_1)}_{=\zeta_D(X,Y)} + \underbrace{\hat{f}_a(R_2)}_{=\zeta_D(X,Y)} \geq \underbrace{\hat{f}_a(R_1 \cup R_2)}_{\geq\zeta_D(X,Y)} + \underbrace{\hat{f}_a(R_1 \cap R_2)}_{\geq\zeta_D(X,Y)}$$

which implies that $\hat{f}_a(R_1 \cup R_2) = \zeta_D(X, Y)$. But this contradicts our assumption that S_1^a and S_2^a were important $X - Y$ arc separators as $\partial^+(R_1 \cup R_2)$ is an $X - Y$ arc separator which dominates both S_a^1 and S_a^1. □

Lemma 3.4.9. *Let $D = (V, A)$ be a directed graph, $X, Y \subset V$ be disjoint vertex sets and let S_a be an important $X - Y$ arc separator. Then $R(X, S) \supseteq R(X, S_a^*)$.*

Proof. Suppose that this is not the case and let $R_1 = R(X, S_a)$ and $R_2 = R(X, S_a^*)$ where $S_a \neq S_a^*$. We know that $R_1, R_2 \supset X$ and the minimality of S_a and S_a^* implies that $\partial^+(R_1) = S_a$ and $\partial^+(R_2) = S_a^*$. By Proposition 3.4.6 the sets $\partial^+(R_1 \cup R_2)$ and $\partial^+(R_1 \cap R_2)$ are also $X - Y$ arc separators and hence $\hat{f}_a(R_1 \cup R_2), \hat{f}_a(R_1 \cap R_2) \geq \zeta_D(X, Y)$. By the submodularity of $\hat{f}_a$ (Lemma 3.4.3) we have that

$$\hat{f}_a(R_1) + \underbrace{\hat{f}_a(R_2)}_{=\zeta_D(X,Y)} \geq \hat{f}_a(R_1 \cup R_2) + \underbrace{\hat{f}_a(R_1 \cap R_2)}_{\geq\zeta_D(X,Y)}$$

which implies that $\hat{f}_a(R_1 \cup R_2) \leq \hat{f}_a(R_1)$. But this contradicts our assumption that S_a was an important $X - Y$ arc separator since $\partial^+(R_1 \cup R_2)$ is an $X - Y$ arc separator which dominates S_a. □

Lemma 3.4.10. *Let $D = (V, A)$ be a directed graph, $X, Y \subset V$ be disjoint vertex sets and S_a be an important $X - Y$ arc separator.*

1. *For every $v \in R(X, S)$, S is a $\{v\} - Y$ arc separator.*
2. *For every arc $e \in S_a$, $S_a \setminus \{e\}$ is an important $X - Y$ arc separator in $D \setminus \{e\}$.*
3. *If S_a is an $X' - Y$ arc separator for some $X' \supset X$ such that X' is reachable from X in the induced subgraph $D[X']$, then S_a is an important $X' - Y$ arc separator.*

Proof.

1. Suppose this were not the case. Since $v \in R(X, S)$ there is a path from X to v in $G \setminus S$. Now since there is also a path from v to Y, this results in a walk from X to Y which can be converted to a path from X to Y in $G \setminus S$. But this is not possible since S is an $X - Y$ arc separator.

2. Suppose $S'_a = S_a \setminus \{e\}$ is not an important $X - Y$ arc separator in $D' = D \setminus \{e\}$. Then, in D', there is an $X - Y$ arc separator S^1_a which dominates S'_a. Consider the set $S^2_a = S^1_a \cup \{e\}$. Observe that S^2_a is also an $X - Y$ arc separator in D. This is because any path from X to Y which does not contain the arc e, exists in D' and hence must contain an arc in S^2_a. Now, since S^1_a dominates S'_a in D', S^2_a dominates S_a in G which contradicts our assumption that S_a is an important $X - Y$ arc separator.

3. Assume that this is not the case. Clearly S_a is a minimal $X' - Y$ arc separator and hence there is an $X' - Y$ arc separator S^1_a which dominates S_a with respect to X'. We will prove that S^1_a also dominates S_a with respect to X. Since we already have that $|S^1_a| \leq |S_a|$, it is enough for us to show that $R(X, S_a) \subset R(X, S^1_a)$.

   First we prove that $R(X, S_a) \subseteq R(X, S^1_a)$. Consider a vertex v in $R(X, S_a)$. Clearly $R(X', S_a) \supseteq R(X, S_a)$. But, $R(X', S_a) \subset R(X', S^1_a)$, which means that $v \in R(X', S^1_a)$. Since X' contains X and is reachable from X in $D[X']$,

and S_a^1 does not contain an arc with both end points inside X' (Obs. 3.4.5), X' is reachable from X in $D[X'] \setminus S_a^1$ also, and hence $v \in R(X, S_a^1)$.

Now, consider some arc $e = (u, w) \in S_a \setminus S_a^1$. By the minimality of S_a, $u \in R(X, S_a)$ and $w \notin R(X, S_a)$. But we have shown that u is also in $R(X, S_a^1)$, which implies that $w \in R(X, S_a^1)$ and hence $R(X, S_a) \subset R(X, S_a^1)$. □

Lemma 3.4.11. *Let $D = (V, A)$ be an undirected graph, $X, Y \subset V$ be disjoint vertex sets of D. For every $k \geq 0$ there are at most 4^k important $X - Y$ arc separators of size at most k.*

Proof. Given $D, X, Y, k \geq 0$ we define a measure $\mu_a(D, X, Y, k) = 2k - \zeta_D(X, Y)$. We prove by induction on $\mu_a(D, X, Y, K)$ that there are at most $2^{\mu_a(D,X,Y,k)}$ important $X-Y$ arc separators of size at most k. For the base case, if $2k-\zeta_D(X, Y) < k$, then $\zeta_D(X, Y) > k$ and hence the number of important $X - Y$ arc separators of size at most k is 0. If $\zeta_D(X, Y) = 0$, it means that there is no path from X to Y and hence the empty set alone is the important $X - Y$ arc separator. Consider $D, X, Y, k \geq 0$ such that $\mu_a = \mu_a(D, X, Y, k) \geq k, \zeta_D(X, Y) > 0$ and assume that the statement holds for all D', X', Y', k' where $\mu_a(D', X', Y', k') < \mu_a$.

By Lemma 3.4.8 there is a unique important $X - Y$ arc separator S_a^* of size $\zeta_D(X, Y)$. Since we have assumed $\zeta_D(X, Y)$ to be positive, S_a^* is non empty. Consider an arc $e = (u, v) \in S^*$. Any important $X - Y$ arc separator S either contains e or does not contain e. For any important $X - Y$ arc separator S_a which contains e, $S_a \setminus \{e\}$ is an important $X - Y$ arc separator in $D \setminus \{e\}$ by Lemma 3.4.10(b). Hence the number of important $X - Y$ arc separators of size at most k in D which contain e, is at most the number of important $X - Y$ arc separators of size at most $k - 1$ in $D \setminus \{e\}$. Observe $\zeta_{D\setminus\{e\}}(X, Y) = \zeta_D(X, Y) - 1$ which implies that $\mu_a(D \setminus \{e\}, X, Y, k - 1) < \mu_a$ and by induction hypothesis, the number of important $X - Y$ arc separators of size at most $k - 1$ in $D \setminus \{e\}$ is bounded by $2^{\mu_a - 1}$ which is also a bound on the number of important $X - Y$ arc separators of size at most k in D which contain e.

Now let S_a be an important $X - Y$ arc separator of size at most k which does not contain e. By Lemma 3.4.9 we know that $R(X, S_a) \supseteq R(X, S_a^*)$ and by the minimality of S_a^*, $u \in R(X, S_a)$ and $v \notin R(X, S_a)$ which implies that $R(X, S_a) \supseteq R(X, S_a^*) \cup \{v\}$. We now set $X' = R(X, S_a^*) \cup \{v\}$. By Proposition 3.4.10(c) we know that S_a is a also an important $X' - Y$ arc separator. Thus a bound on the

number of important $X' - Y$ arc separators of size at most k is also a bound on the number of important $X - Y$ arc separators of size at most k which do not contain v. First note that $\zeta_D(X', Y) > \zeta_D(X, Y)$ since otherwise we would have an $X - Y$ arc separator which dominates S_a^*. Now, $\mu_a(D, X', Y, k) < \mu_a$ and by induction hypothesis, the number of important $X' - Y$ arc separators of size at most k is bounded by $2^{\mu_a - 1}$.

Summing up the bounds we get that the number of important $X - Y$ arc separators of size at most k is bounded by $2 \cdot 2^{\mu_a - 1} = 2^{\mu_a} \leq 2^{2k}$. □

3.5 Tight Instances for the Bound on Number of Important Separators

In this section we demonstrate instances which have a *large* number of important separators, thus proving that the bounds obtained in the previous sections are essentially tight.

Lemma 3.5.1. *The bound of 4^k on the number of important $X-Y$ vertex separators is tight up to a polynomial factor of k even in planar acyclic graphs.*

Proof. In the proof of Lemma 3.2.11 we have shown that the number of important $X - Y$ vertex separators is bounded by the sum of the important separators which contain and those which do not contain some vertex of the smallest important $X - Y$ separator. This is sum is clearly maximized when the measure μ decreases precisely by 1 in either branch. Based on this idea, we demonstrate instances (see Fig. 3.5.2) where the number of important $X - Y$ separators is at least $4^k/poly(k)$ where $poly(k)$ is some polynomial function of k.

For the instance shown in Fig. 3.5.2(a), any minimal $X - Y$ vertex separator is an important $X - Y$ vertex separator. Hence the number of important $X - Y$ separators of size at most k is the number of minimal $X - Y$ separators of size at most k. But any minimal $X - Y$ separator of size say p, corresponds to a subtree of T rooted at X and with p leaves. Hence the number of minimal $X - Y$ separators of size at most k is the number of subtrees of T which are rooted at X and have at most k leaves. But this number is the Catalan number C_{k-1}, which is asymptotically $4^k/poly(k)$.

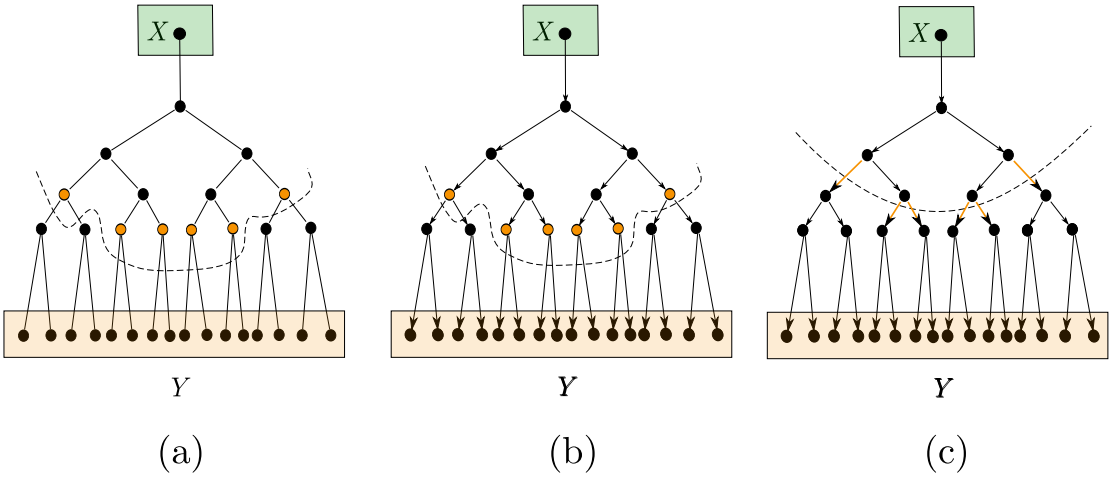

Figure 3.2: Three instances which achieve the given bound upto a polynomial factor of k. (a) Vertex Separators in Undirected Graphs (b) Vertex Separators in Directed Graphs (c) Arc Separators in Directed Graphs.

An analogous argument holds for the instances shown in Fig. 3.5.2(b) and Fig. 3.5.2(c), and hence there are at least $4^k/poly(k)$ vertex and arc separators in the respective instances.

Since the instances in Fig 3.5.2 are all planar and acyclic, we note that this bound is essentially tight even in planar acyclic graphs.

□

Remark 3.5.2. *The example of the tight instances can be found in the slides of Dániel Marx titled* Important separators and spiders.

3.6 Computing Important Separators

In this section we will present the (implicit) algorithm of Chen et al. [12] to enumerate important vertex separators in undirected graphs and extend it to algorithms enumerating important vertex and arc separators in directed graphs.

Lemma 3.6.1. *1. ([38]) Given an undirected graph $G = (V, E)$ and disjoint vertex subsets X, Y, and S, it can be checked in $\mathcal{O}(|V|^4)$ time if S is an important $X - Y$ vertex separator in G. Furthermore, if S is a minimal*

$X - Y$ vertex separator that is not important, then we can find in $\mathcal{O}(|V|^5)$ time an important $X-Y$ vertex separator S_1 that dominates S. In particular, we can find the unique smallest important $X - Y$ vertex separator in time $\mathcal{O}(|V|^5)$.

2. *Given a directed graph $D = (V, A)$ and disjoint vertex subsets X,Y and S, it can be checked in $\mathcal{O}(|V|^4)$ time if S is an important $X - Y$ vertex separator in D. Furthermore, if S is a minimal $X - Y$ vertex separator that is not important, then we can find in $\mathcal{O}(|V|^5)$ time an important $X - Y$ vertex separator S_1 that dominates S. In particular, we can find the unique smallest important $X - Y$ vertex separator in time $\mathcal{O}(|V|^5)$.*

3. *Given a directed graph $D = (V, A)$ and disjoint vertex subsets X and Y, and a set S_a of arcs, it can be checked in $\mathcal{O}(|V|^4)$ time if S_a is an important $X-Y$ arc separator in D. Furthermore, if S_a is a minimal $X-Y$ arc separator that is not important, then we can find in $\mathcal{O}(|V|^5)$ time an important $X - Y$ arc separator S_a^1 that dominates S_a. In particular, we can find the unique smallest important $X - Y$ arc separator in time $\mathcal{O}(|V|^5)$.*

Proof. 1. Testing the minimality of S can be done by checking if $S \setminus \{v\}$ is an $X - Y$ vertex separator for every vertex v in S. If S is indeed minimal then for every v in S we test if there is an $X - Y$ separator S' of size at most $|S|$ that does not contain any vertex of $R(X, S) \cup \{v\}$. This can be done by setting $X' = R(X, S) \cup \{v\}$ and finding a minimum size $X' - Y$ vertex separator by applying standard network flow techniques [1]. This takes $\mathcal{O}(|V|^3)$ time. If we do find such a separator S', it will dominate S and we can conclude that S is not an important $X-Y$ vertex separator. We can repeat this process on S' to either verify that S' is an important $X - Y$ vertex separator or find another $X-Y$ vertex separator dominating S' and so on until we reach an important $X-Y$ vertex separator. This process will terminate in at most $|V|$ steps since each subsequent separator dominates all the previous ones. It is easy to see that if we start this procedure with a minimum size $X-Y$ vertex separator, the important separator returned by this procedure will be the unique smallest important $X-Y$ vertex separator S^*.

2. Testing the minimality of S can be done by checking if $S \setminus \{v\}$ is an $X - Y$ vertex separator for every vertex v in S. If S is indeed minimal then for every v in S we test if there is an $X - Y$ separator S' of size at most $|S|$ that does not contain any vertex of $R(X, S) \cup \{v\}$. This can be done by setting $X' = R(X, S) \cup \{v\}$ and finding a minimum size $X' - Y$ vertex separator by applying standard network flow techniques. This takes $\mathcal{O}(|V|^3)$ time. If we do find such a separator S', it will dominate S and we can conclude that S is not an important $X - Y$ vertex separator. We can repeat this process on S' to either verify that S' is an important $X - Y$ vertex separator or find another $X - Y$ vertex separator dominating S' and so on until we reach an important $X - Y$ vertex separator. This process will terminate in at most $|V|$ steps since each subsequent separator dominates all the previous ones. It is easy to see that if we start this procedure with a minimum size $X - Y$ vertex separator, the important separator returned by this procedure will be the unique smallest important $X - Y$ vertex separator S^*.

3. Testing the minimality of S_a can be done by checking if $S_a \setminus \{e\}$ is an $X - Y$ arc separator for every arc $e = (u, v)$ in S_a. If S_a is indeed minimal then for every e in S_a we test if there is an $X - Y$ arc separator S'_a of size at most $|S_a|$ that does not contain any vertex of $R(X, S_a) \cup \{v\}$. This can be done by setting $X' = R(X, S_a) \cup \{v\}$ and finding a minimum size $X' - Y$ arc separator by applying standard network flow techniques. This takes $\mathcal{O}(|V|^3)$ time. If we do find such a separator S'_a, it will dominate S_a and we can conclude that S_a is not an important $X - Y$ arc separator. We can repeat this process on S'_a to either verify that S'_a is an important $X - Y$ arc separator or find another $X - Y$ arc separator dominating S'_a and so on until we reach an important $X - Y$ arc separator. This process will terminate in at most $|V|$ steps since each subsequent separator dominates all the previous ones. It is easy to see that if we start this procedure with a minimum size $X - Y$ arc separator, the important separator returned by this procedure will be the unique smallest important $X - Y$ arc separator S^*_a.

Lemma 3.6.2. *1. Given an undirected graph $G = (V, E)$, disjoint vertex subsets X and Y and a positive integer k, there is an algorithm which in $\mathcal{O}^*(4^k)$ time*

enumerates all the important $X - Y$ vertex separators of size at most k.

2. *Given a directed graph $D = (V, A)$, disjoint vertex subsets X and Y and a positive integer k, there is an algorithm which in $\mathcal{O}^*(4^k)$ time enumerates all the important $X - Y$ vertex separators of size at most k.*

3. *Given a directed graph $D = (V, A)$, disjoint vertex subsets X and Y and a positive integer k, there is an algorithm which in $\mathcal{O}^*(4^k)$ time enumerates all the important $X - Y$ arc separators of size at most k.*

Input : Graph $G = (V, E)$, disjoint vertex subsets X and Y, positive integer k
Output: Set of all important $X - Y$ vertex separators of size at most k or NO if none exist.

if $k < 0$ **then return** NO
Compute a mimimum size $X - Y$ vertex separator S
if $|S| > k$ **then**
  return NO
end
else *Compute the unique minimum size important $X - Y$ vertex separator S^* and select a vertex $v \in S^*$*
$\mathcal{S}_1 \leftarrow Find - IS_1(G \setminus \{v\}, X, Y, k-1)$
$X' = R(X, S^*) \cup \{v\}$
$\mathcal{S}_2 \leftarrow Find - IS_1(G, X', Y, k)$
if $\mathcal{S}_1$ *is* NO **then return** $\mathcal{S}_2$
if $\mathcal{S}_2$ *is* NO **then return**NO
return $\mathcal{S}_1 \cup \mathcal{S}_2$

Algorithm 3.6.1: Algorithm $Find - IS_1$ to enumerate all important $X - Y$ vertex separators of size at most k

Proof. 1. The algorithm (Algorithm 3.6.1) follows from the proof of Lemma 3.2.11. We find the smallest important $X - Y$ vertex separator S^*, pick a vertex v of S^* and recursively enumerate all the important $X - Y$ vertex separators containing v and those that do not contain v.

2. The algorithm (Algorithm 3.6.2) follows from the proof of Lemma 3.3.10. We find the smallest important $X - Y$ vertex separator S^*, pick a vertex v

Input : Directed graph $D = (V, A)$, disjoint vertex subsets X and Y, positive integer k
Output: Set of all important $X - Y$ vertex separators of size at most k or NO if none exist.

if $k < 0$ **then return** NO
Compute a mimimum size $X - Y$ vertex separator S
if $|S| > k$ **then**
  return NO
end
else *Compute the unique minimum size important $X - Y$ vertex separator S^* and select a vertex $v \in S^*$*
$\mathcal{S}_1 \leftarrow Find - IS_2(D \setminus \{v\}, X, Y, k-1)$
$X' = R(X, S^*) \cup \{v\}$
$\mathcal{S}_2 \leftarrow Find - IS_2(D, X', Y, k)$
if $\mathcal{S}_1$ *is* NO **then return** $\mathcal{S}_2$
if $\mathcal{S}_2$ *is* NO **then return**NO
return $\mathcal{S}_1 \cup \mathcal{S}_2$

Algorithm 3.6.2: Algorithm $Find - IS_2$ to enumerate all important $X - Y$ vertex separators of size at most k

of S^* and recursively enumerate all the important $X - Y$ vertex separators containing v and those that do not contain v.

3. The algorithm (Algorithm 3.6.3) follows from the proof of Lemma 3.4.11. We find the smallest important $X - Y$ arc separator S^*, pick an arc e of S^* and recursively enumerate all the important $X - Y$ arc separators containing e and those that do not contain e. □

3.7 Summary

In this Chapter, we gave the definitions and lemmata used to develop the machinery of important vertex separators in undirected graphs as introduced in [38] and extended these definitions formally to important vertex and arc separators in directed graphs. We then gave a proof of the bound on the number of important separators of bounded size (implicitly) given in [12] and extended it to important vertex/arc separators on directed graphs. We then demonstrated instances

Input : Directed graph $D = (V, A)$, disjoint vertex subsets X and Y, positive integer k
Output: Set of all important $X - Y$ arc separators of size at most k or NO if none exist.

if $k < 0$ **then return** NO
Compute a mimimum size $X - Y$ arc separator S_a
if $|S_a| > k$ **then**
  return NO
end
else *Compute the unique minimum size important $X - Y$ arc separator S_a^* and select a arc $e = (u, v) \in S_a^*$*
$\mathcal{S}_a^1 \leftarrow Find - IS_3(D \setminus \{e\}, X, Y, k - 1)$
$X' = R(X, S_a^*) \cup \{v\}$
$\mathcal{S}_a^2 \leftarrow Find - IS_3(D, X', Y, k)$
if $\mathcal{S}_a^1$ *is* NO **then return** $\mathcal{S}_a^2$
if $\mathcal{S}_a^2$ *is* NO **then return**NO
return $\mathcal{S}_a^1 \cup \mathcal{S}_a^2$

Algorithm 3.6.3: Algorithm $Find - IS_3$ to enumerate all important $X - Y$ arc separators of size at most k

which achieve this bound upto a polynomial factor, thus showing that we cannot hope for better bounds in general. Finally, we presented the algorithm to enumerate all important separators of bounded size in undirected graphs, which is implicit in [12] and gave analogous algorithms to enumerate important vertex and arc separators in directed graphs.

As for further research in this particular direction, it will be interesting to show classes of graphs where the number of important separators might not be as large as the bound given in this Chapter. Furthermore, one can also ask the question if there is a way of classifying the important separators themselves in some manner, so that some important separators turn out to be *more important* than other important separators.

4

Minimum Node Multiway Cut

The MULTIWAY CUT problem is a generalization of the classical $s-t$ cut problem where given an undirected graph, two terminals s, t and an integer k the problem is to find a set of at most k vertices, whose removal disconnects s from t. Such a set of can be found in polynomial time by classical network flow techniques [1]. However for every $l \geq 3$, given l terminals $t_1, \ldots, t_l$, it becomes NP-complete to find a set of at most k vertices such that in the graph obtained by the removal of these vertices no two terminals occur in the same connected component [15]. This problem has applications in areas as varied as Multiprocessor Scheduling [48], and Medical Imaging [8, 9]. An easy $(2-2/l)-$approximation algorithm for the edge variant (where we want to find a set of at most k edges such that in the graph obtained by the removal of these vertices, no two terminals occur in the same connected component) was presented in [15] and in [25], a $(2-2/l)-$approximation algorithm was given for the vertex variant of this problem. This problem has also been studied extensively on planar and tree-like structures [21, 31, 16]. Using the notion of important vertex separators, an FPT algorithm for MULTIWAY CUT was given in [38], and a significantly faster FPT algorithm was presented in [12].

Organization of the Chapter In Section 4.1, we formally define the variants of the Multiway Cut problem we will consider in our discussion. In Section 4.2, we present the algorithm for Multiway Cut after initially discussing the part played by the concept of Important Separators in the algorithm.

4.1 Preliminaries

UNRESTRICTED MULTIWAY CUT	
Input:	An undirected graph $G = (V, E)$, a set T of vertices called terminals, integer k
Parameter:	k
Question:	Does there exist a set $S \subseteq V$ of at most k vertices such that the vertices of T are pairwise disconnected in $G \setminus S$?

RESTRICTED MULTIWAY CUT	
Input:	An undirected graph $G = (V, E)$, a set T of vertices called terminals, integer k
Parameter:	k
Question:	Does there exist a set $S \subseteq V \setminus T$ of at most k vertices such that the vertices of T are pairwise disconnected in $G \setminus S$?

EDGE MULTIWAY CUT	
Input:	An undirected graph $G = (V, E)$, a set T of vertices called terminals, integer k
Parameter:	k
Question:	Does there exist a set $S \subseteq E$ of at most k edges such that the vertices of T are pairwise disconnected in $G \setminus S$?

Lemma 4.1.1. *1. There is a polynomial time parameter preserving reduction from* UNRESTRICTED MULTIWAY CUT *to* RESTRICTED MULTIWAY CUT

2. There is a polynomial time parameter preserving reduction from EDGE MULTIWAY CUT *to* RESTRICTED MULTIWAY CUT

.

Proof. 1. Let $(G = (V, E), T = \{t_1, \ldots, t_r\}, k)$ be an instance of UNRESTRICTED MULTIWAY CUT. We define an instance (G', T', k) of RESTRICTED MULTIWAY CUT as follows. We add r new vertices $t'_1, \ldots, t'_r$ one corresponding to each terminal to G and add edges (t_i, t'_i) for every i. We call this graph G' and set T' as $\{t'_1, \ldots, t'_r\}$. Clearly this reduction can be done in polynomial time and we claim that (G, T, k) is a YES instance of UNRESTRICTED MULTIWAY CUT iff (G', T', k) is a YES instance of RESTRICTED MULTIWAY CUT.

Suppose (G, T, k) is a YES instance of UNRESTRICTED MULTIWAY CUT and let S be a solution for this instance. Clearly S is a solution for the instance (G', T', k) which does not contain any vertex from T'.

Conversely, any solution for the instance (G', T', k) which does not contain a vertex of T' is also a solution for the instance (G, T, k).

2. Let $(G = (V, E), T = \{t_1, \ldots, t_r\}, k)$ be an instance of EDGE MULTIWAY CUT. We define an instance (G', T', k) of RESTRICTED MULTIWAY CUT as follows. We first add r new vertices $t'_1, \ldots, t'_r$ one corresponding to each terminal to G and add edges (t_i, t'_i) for every i. We define the graph $L(G)$ as the line graph of this modified graph and define the set of terminals T' as the vertices of $L(G)$ which correspond to the edges $(t_1, t'_1), \ldots, (t_r, t'_r)$. It is easy to see that this reduction can be done in polynomial time and we claim that (G, T, k) is a YES instance of EDGE MULTIWAY CUT iff $(L(G), T', k)$ is a YES instance of RESTRICTED MULTIWAY CUT.

Suppose (G, T, k) is a YES instance of EDGE MULTIWAY CUT and let S be a solution for this instance. Clearly the set of vertices corresponding to the edges in S is a solution for the instance $(L(G), T', k)$ which does not contain any vertex from T'.

Conversely, let S' be a solution for the instance (G', T', k) which does not contain a vertex of T'. Then the set of edges corresponding to the vertices of S' in the original instance is clearly a solution for this instance. □

Due to the above Lemma, in the rest of this Chapter, we will concentrate on the RESTRICTED MULTIWAY CUT problem.

4.2 Multiway Cut, Important Separators and the Algorithm

Lemma 4.2.1. ([38]) *Let $(G = (V, E), T, k)$ be an instance of* Restricted Multiway Cut. *If (G, T, k) is a* Yes *instance then it has a solution $\hat{S}$ such that a minimal subset of $\hat{S}$ separating t_1 from $T \setminus \{t_1\}$ is an important $t_1 - T \setminus \{t_1\}$ vertex separator.*

Proof. Let $S \subseteq V$ be a minimal solution and let S_1 be a minimal subset of S such that $G \setminus S$ has no $t_1 - T \setminus \{t_1\}$ path. If S_1 is the empty set, it must be the case that there is no path from t_1 to $T \setminus \{t_1\}$ in G. By definiton, S_1 is an important $t_1 - T \setminus \{t_1\}$ vertex separator and we are done by setting $\hat{S} = S$. Hence, we will assume that S_1 is non empty. If S_1 is an important $t_1 - T \setminus \{t_1\}$ vertex separator, we are done by setting $\hat{S} = S$. Suppose that this is not the case.

Since S_1 is a minimal $t_1 - T \setminus \{t_1\}$ vertex separator which is not important, there is a $t_1 - T \setminus \{t_1\}$ vertex separator S_2 which dominates S_1. Set $\hat{S} = (S \setminus S_1) \cup S_2$. We claim that $\hat{S}$ is a solution of this instance which satisfies the statement of the Lemma. Clearly $|\hat{S}| \leq |S|$ and the minimal part of $\hat{S}$ separating t_1 from $T \setminus \{t_1\}$ is S_2 which by our assumption is an important $t_1 - T \setminus \{t_1\}$ vertex separator.

It remains for us to prove that $\hat{S}$ is a multiway cut of T. Suppose this not so and let there be a path P from t_i to t_j in $G \setminus \hat{S}$. Since S was a multiway cut of T, P contains a vertex $v \in S_1 \setminus S_2$ (see Fig. 4.1) and hence there is a path P' from v to T. Since S_1 was minimal, v has a neighbor u in $R(t_1, S_1)$. Now, $R(t_1, S_1) \subset R(t_1, S_2)$ and $v \notin S_2$. Hence, it must be the case that $v \in R(t_1, S_2)$. But then, S_2 is a $t_1 - T \setminus \{t_1\}$ vertex separator and there is a path from $v \in R(t_1, S_2)$ to T which is not possible by Lemma 3.2.10(a). This concludes the proof of the Lemma. □

4.2.1 Algorithm for Multiway Cut

Theorem 4.2.2. ([12]) Restricted Multiway Cut *can be solved in $\mathcal{O}^*(4^k)$ time.*

Proof. The idea of the algorithm is as follows. Let T be the set of terminals and let t_1 be a terminal not disconnected from $T \setminus \{t_1\}$. By Lemma 4.2.1 we know that the solution, if there exists one, contains an important $t_1 - T \setminus \{t_1\}$

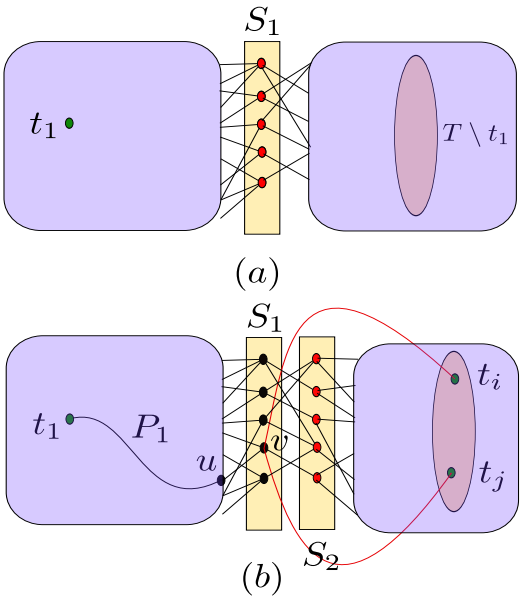

Figure 4.1: An illustration of the instance with (a) the minimal part S_1, of the solution separating t_1 from the rest (b) S_1 replaced with S_2

separator. Hence we guess an important $t_1 - T \setminus \{t_1\}$ vertex separator, remove these vertices from the graph, and continue. In order to simplify the analysis of the algorithm, we embed the algorithm for enumerating important vertex separators (Algorithm 3.6.1) into the algorithm for MULTIWAY CUT.

To analyze the algorithm MWC (Algorithm 4.2.1) we define the search tree $\mathbb{T}(G, T, k)$ resulting from a call $MWC(G, T, k)$ inductively as follows. The tree $\mathbb{T}(G, T, k)$ is a rooted tree whose root node corresponds to the instance (G, T, k). If $MWC(G, T, k)$ does not make a recursive call then (G, T, k) is said to be the only node of this tree. If $MWC(G, T, k)$ does make recursive calls, the children of (G, T, k) correspond to the instances given as input to the calls made inside the call $MWC(G, T, k)$. The subtree rooted at a child node (G', T', k') corresponds to the search tree $\mathbb{T}(G', T', k')$.

We define a measure for the input (G, T, k) of the algorithm MWC as $\mu(G, T, k) = 2k - \lambda_G(t_1, T \setminus \{t_1\})$.

Lemma 4.2.3. *Let (G, T, k) be an instance of* RESTRICTED MULTIWAY CUT *and let (G', T', k') be a child node of (G, T, k) in $\mathbb{T}(G, T, k)$. Then $\mu(G, T, k) < \mu(G', T', k')$.*

Input : An instance $(G, T = \{t_1, \ldots, t_r\}, k)$ of RESTRICTED MULTIWAY CUT
Output: A multiway cut of size at most k for the instance (G, T, k) if it exists and NO otherwise

1 **if** $k < 0$ **then return** NO
2 **while** *there is no path from* t_1 *to* $T \setminus \{t_1\}$ **do**
3 $\quad T \leftarrow T \setminus \{t_1\}$, $r \leftarrow r - 1$, and rename T as $t_1, \ldots, t_r$
4 **end**
5 *Compute a minimum size* $t_1 - T \setminus \{t_1\}$ *vertex separator* S
6 **if** $|S| > k$ **then return** NO
7 **else** *Compute the unique minimum size important* $t_1 - T \setminus \{t_1\}$ *vertex separator* S^* *and select a vertex* $v \in S^*$
8 $S_1 \leftarrow MWC(G \setminus \{v\}, T, k-1)$
9 **if** S_1 *is not* NO **then return** $S_1 \cup \{v\}$
10 *identify all the vertices of* $R(t_1, S^*) \cup \{v\}$ *into a single vertex, name this new vertex* t_1'*, and name the resulting graph* G'.
11 $S_2 \leftarrow MWC(G', T' = \{t_1', t_2, t_3, \ldots, t_r\}, k)$
12 **return** S_2

Algorithm 4.2.1: Algorithm MWC for RESTRICTED MULTIWAY CUT

Proof. (a) Suppose the child node (G', T', k') corresponds to a call made in Step 8. Then $G' = G \setminus \{v\}$, $T' = T$ and $k' = k - 1$ where v is some vertex in S^*, the unique smallest important $t_1 - T \setminus \{t_1\}$ vertex separator in G. As seen in the proof of Lemma 3.2.11, $2k - \lambda_G(t_1, T \setminus \{t_1\}) > 2k' - \lambda'_G(t_1, T' \setminus \{t_1\})$ and hence $\mu(G, T, k) > \mu(G', T', k')$.

(b) Suppose the child node (G', T', k') corresponds to a call made in Step 11. It follows from the proof of Lemma 3.2.11 that $2k - \lambda_G(t_1, T \setminus \{t_1\}) > 2k - \lambda_G(t_1', T \setminus t_1')$ and hence $\mu(G, T, k) > 2k - \lambda_G(R(t_1, S^*) \cup \{v\})$, which implies that $\mu(G, T, k) > \mu(G', T', k')$. □

Correctness Proof

Lemma 4.2.4. *The algorithm* MWC *correctly solves the* RESTRICTED MULTIWAY CUT *problem.*

Proof. Correctness of Step 1 is obvious and Steps 2-4 only remove terminals which are already separated from the rest of the terminals. Step 6 is correct because the size of the minimum $t_1 - T \setminus \{t_1\}$ vertex separator is a lower bound on the solution size. The correctness of steps 8 and 11 follows from Lemma 4.2.1 since these steps merely branch on the choice of an important $t_1 - T \setminus \{t_1\}$ vertex separator. □

Running time Analysis

Lemma 4.2.5. *The number of leaves of the search tree $\mathbb{T}(G, T, k)$ is bounded by $2^{\mu(G,T,k)}$.*

Proof. We will prove the statement of the Lemma by induction on $\mu(G, T, k)$. For the base case, when $\lambda_G(t_1, T \setminus \{t_1\}) > k$ or when $\lambda_G(t_1, T \setminus \{t_1\}) = 0$ there is no branching involved and hence the number of leaves is 1 which agrees with the statement of the Lemma. Now assume that $\mu(G, T, k) > k$ and $\lambda_G(t_1, T \setminus \{t_1\}) > 0$ and that the statement of the lemma holds for all instances (G', T', k') such that $\mu(G', T', k') < \mu(G, T, k)$. Let (G_1, T_1, k_1) and (G_2, T_2, k_2) be the children of (G, T, k). By Lemma 4.2.3, $\mu(G_1, T_1, k_1), \mu(G_2, T_2, k_2) < \mu(G, T, k)$. Hence by the induction hypothesis, the number of leaves in the subtree rooted at (G_1, T_1, k_1) is bounded by $2^{\mu(G_1,T_1,k_1)}$ and the number of leaves in the sub tree rooted at (G_2, T_2, k_2) is bounded by $2^{\mu(G_2,T_2,k_2)}$. Therefore, the number of leaves in the sub tree rooted at (G, T, k) is bounded by $2^{\mu(G_1,T_1,k_1)} + 2^{\mu(G_2,T_2,k_2)}$ which is at most $2.2^{\mu(G,T,k)-1} \leq 2^{\mu(G,T,k)}$. □

Along every root to leaf path of $\mathbb{T}(G, T, k)$, we either pick a solution vertex (Step 8) or identify at least two vertices (Step 11), and hence the length of any root to leaf path in the tree is bounded by $\mathcal{O}(n)$. We now bound the time taken at each node of the tree before making further recursive calls. This time is bounded by the time required to compute the unique smallest important $t_1 - T \setminus \{t_1\}$ vertex separator which by Lemma 3.6.1 is $\mathcal{O}(n^5)$. Hence the running time of the algorithm is bounded by $\mathcal{O}(4^k n^6)$. □

Theorem 4.2.2 along with Lemma 4.1.1 gives us the following Corollaries.

Corollary 4.2.6. Unrestricted Multiway Cut *can be solved in time $\mathcal{O}^*(4^k)$.*

Corollary 4.2.7. EDGE MULTIWAY CUT *can be solved in time* $\mathcal{O}^*(4^k)$.

Furthermore, Theorem 4.2.2 also results in the following Corollary.

Corollary 4.2.8. ([*Lemma 3.9,* ([38])]) *The* MULTICUT *problem (see Appendix for definition) is FPT with parameters* k *and* l *where* ℓ *is the number of* (s_i, t_i) *pairs we are required to disconnect.*

4.3 Summary

In this Chapter, we applied the concept of important separators to present the FPT algorithm of Chen et al. [12] for RESTRICTED MULTIWAY CUT and using Lemma 4.1.1, also proved two other variants of the Multiway Cut problem to be FPT. We showed that the part of the solution isolating a terminal from the rest of the terminals may be assumed to be an important separator, and the algorithm worked by repeatedly selecting a terminal and isolating it from the rest of the terminals by branching on an important vertex separator separating this terminal from the rest.

Other than considering edge variants, another natural generalization is to consider the MULTIWAY CUT problem on directed graphs. We note that Lemma 4.2.1 breaks down in the directed case and hence this algorithm does not extend naturally to the directed case.

Open Questions and Future Directions. Further questions in this direction include looking for a faster FPT algorithm and also proving upper or lower bounds on kernels of this problem.

5

Directed Feedback Vertex Set

5.1 Introduction

In the FEEDBACK VERTEX SET (FVS) problem, we are given a graph $G = (V, E)$ and asked to find a subset F in the graph such that $G \setminus F$ is acyclic. We call such a set F a feedback vertex set (FVS) of G. The graph G can be undirected or directed. The problem is NP-complete in both directed and undirected graphs [24].

The feedback vertex set problem, especially in directed graph, has important applications in Database Systems [23] and Operating Systems [47] to solve many problems, with one such problem being deadlock recovery. A deadlock is represented by a directed cycle in the system resource-allocation graph D. Therefore, in order to recover from deadlocks, we need to abort a set of processes in the system which equates to finding a feedback vertex set in D.

In the parameterized version of this problem, we want an FPT algorithm parameterized by the size of the set F. The parameterized version of this problem on undirected graphs was proved to be FPT [19, 4] and fixed parameter tractability of the same problem on directed graphs was open for a long time. This problem has been extensively studied on subclasses of directed graphs, for instance in tournaments [18].

For the case of general digraphs, Chen et al. [13] gave an FPT algorithm for this problem that runs in time $\mathcal{O}^*(4^k k!)$ where k is the size of the feedback set asked for. This algorithm reduces the DFVS problem to a multicut problem, which is then solved in FPT time. Their algorithm utilizes the notions of important separators implicitly and in this Chapter, we will present this algorithm with

explicit use of the properties surrounding the notion of important separators.

Organization of the Chapter In Section 5.2, we set up a few definitions which will help us in presenting the rest of the Chapter. In Section 5.3 we describe the method of iterative compression as applied to the DIRECTED FEEDBACK VERTEX SET (DFVS) problem and reduce it to instances of a cut problem in directed graphs. In Section 5.4 we discuss the part played by important separators in solving this cut problem and present an FPT algorithm for this problem using important separators.

5.2 Preliminaries

Given a directed graph $D = (V, A)$ and a partition of V into V_1 and V_2 such that the induced sub graphs $D[V_1]$ and $D[V_2]$ are acyclic, we call the pair (V_1, V_2) a *DAG Bipartition* of the graph D. Given a DAG bipartition (V_1, V_2) and a topological ordering $\pi : V_2 \rightarrow [|V_2|]$ of the induced sub graph $D[V_2]$ we call the triple (V_1, V_2, π) an *ordered DAG Bipartition* of D.

5.3 Iterative Compression for DFVS

Given an instance $(D = (V, A), k)$ of DFVS we will apply the standard technique of iterative compression introduced in . We fix an arbitrary ordering of the vertices of D as $v_1, \ldots, v_n$, define the sub graph D_i as $D[\{v_1, \ldots, v_i\}]$ and define the instances $I_1, \ldots, I_n$ as $I_i = (D_i, k)$. One by one, we iterate through the instances I_i starting with $i = k + 1$ and with the help of a known (bigger) solution try to find a solution $\hat{S}_i$ of size at most k for the i^{th} instance. Formally we define the COMPRESSION VERSION of our problem as follows.

DIRECTED FEEDBACK VERTEX SET COMPRESSION	
Input:	$(D = (V, A), S, k)$, where D is a directed graph and S a set of vertices of size at most $k + 1$ such that $D \setminus S$ is acyclic, k a positive integer
Parameter:	k
Question:	Does there exist a set $\hat{S}$ containing at most k vertices such that $D \setminus \hat{S}$ is acyclic?

We will reduce the DFVS problem to at most n instances of the DIRECTED FEEDBACK VERTEX SET COMPRESSION problem as follows. We use the fact that the set $S_{k+1} = \{v_1, \ldots, v_{k+1}\}$ is a solution of size at most $k + 1$ for I_{k+1} and $\hat{S}_{i-1} \cup \{v_i\}$ is a solution of size at most $k + 1$ for instance I_i. We then check if (D_{k+2}, S_{k+1}, k) is a YES instance of DIRECTED FEEDBACK VERTEX SET COMPRESSION and if so get a solution $\hat{S}_{k+2}$ of size at most k, set this as S_{k+3} and move on to the iteration. If any of the intermediate instances of DIRECTED FEEDBACK VERTEX SET COMPRESSION is a NO instance, then clearly the input instance of DFVS is also a NO instance. Finally the solution for the original input instance will be $\hat{S}_n$. Since there can be at most n iterations, the total time taken is bounded by n times the time required to solve the DIRECTED FEEDBACK VERTEX SET COMPRESSION problem.

Now we present the algorithm for the DIRECTED FEEDBACK VERTEX SET COMPRESSION problem as follows. Let the input instance be $I = (D = (V, A), S, k)$. We guess a subset Y of S of size at most k to be in our new solution and set $N := S \setminus Y$. Now, if the induced sub graph $D[N]$ is not acyclic it is easy to see that we have to reject this particular guess of Y. Since S was a solution in the first place, $D[V \setminus S]$ is acyclic. We have thus reduced this problem to checking if the instance $(D \setminus Y, V \setminus S, N, k)$ has a feedback vertex set S' of size atmost k such that $S' \subseteq V \setminus S$. We will use the fact that $(V \setminus S, N)$ is a bipartition of $D \setminus Y$ to solve this problem. In order to do that we first formally define the following variants of the DFVSproblem.

DAG BIPARTITION FVS	
Input:	$(D = (V, A), V_1, V_2, k)$, where D is a directed graph, (V_1, V_2) is a DAG bipartition of D, k is a positive integer
Parameter:	k
Question:	Does there exist a feedback vertex set S of size at most k for D, such that $S \subseteq V_1$?

ORDERED DAG BIPARTITION FVS	
Input:	$(D = (V, A), V_1, V_2, \pi, k)$, where D is a directed graph, (V_1, V_2) is a DAG bipartition of D, π is a topological ordering of $D[V_2]$, and k is a positive integer
Parameter:	k
Question:	Does there exist a set $S \subseteq V_1$ of size at most k, such that in the graph $D \setminus S$ there are no paths from v_1 to v_2 for any $v_1, v_2 \in V_2$ with $\pi(v_1) \geq \pi(v_2)$.

Lemma 5.3.1. *Given an instance $I_1 = (D = (V, A), V_1, V_2, k)$ of the* DAG BIPARTITION FVS *problem, I is a* YES *instance iff there is an instance $I_2 = (D, V_1, V_2, \pi, k)$ of the* ORDERED DAG BIPARTITION FVS *problem which is also a* YES *instance.*

Proof. Suppose the instance I_1 is a YES instance and let F be an FVS of size at most k where $F \subseteq V_1$. Then, $D \setminus F$ is acyclic and hence has a topological ordering $\tilde{\pi}$. Since $F \cap V_2 = \phi$, $\tilde{\pi}$ induces an ordering among the vertices of V_2. Let this induced ordering be π. We show that the instance (D, V_1, V_2, π, k) is a YES instance of ORDERED DAG BIPARTITION FVS by showing that F is a solution for this instance. If it is not, then there are vertices v_i and v_j in V_2 such that $i \geq j$ and there is a path from v_i to v_j in $D \setminus F$. But this cannot happen since $\tilde{\pi}$ was a topological ordering of $D \setminus F$ where v_i occurs before v_j.

Conversely, suppose there is an instance $I_2 = (D, V_1, V_2, \pi, k)$ of the ORDERED DAG BIPARTITION FVS problem which has a solution F of size at most k. We claim that the set F is indeed a feedback vertex set for the instance (D, V_1, V_2, k) contained in V_1. By the definition of the ORDERED DAG BIPARTITION FVS problem, F is contained in V_2. It remains to show that $D \setminus F$ is acyclic. Suppose this

is not so. Consider a cycle C in the graph $D \setminus F$. Since $D[V_1]$ is acyclic C must contain at least one vertex of V_1. If it contained exactly one vertex v of V_1, clearly there is a path from v to v in the graph $D \setminus F$ which contradicts our assumption that F was a solution for the instance I_2. If C contained two vertices u and v we have two directed paths one from u to v and the other from v to u, one of which must contradict our assumption that F was a solution for the instance I_2. Hence F is indeed a feedback vertex of D. □

We will use Lemma 5.3.1 to solve the DAG BIPARTITION FVS problem as follows. Given an instance (D, V_1, V_2, k), we guess the topological ordering of the vertices of V_2 and for each guess π, solve the corresponding instance (D, V_1, V_2, π, k) of the ORDERED DAG BIPARTITION FVS problem. Our approach to solve this problem will be to first transform it into an instance of a separation problem which we solve using the notion of important separators.

Given an instance $I = (D, V_1, V_2, \pi, k)$ of the ORDERED DAG BIPARTITION FVS problem where $V_2 = \{v_1 \ldots, v_\ell\}$ and $\pi(v_i) = i$, we replace every $v_i \in V_2$ with two vertices s_i and t_i, make the in-neighbors of v_i the in-neighbors of t_i and make the out-neighbors of v_i the out-neighbors of s_i. Clearly this modified graph D_{skew} is acylic since any cycle in D uses a vertex of V_2 and hence must now be broken by our construction.

SKEW MULTICUT	
Input:	$(D = (V, A), \mathcal{S} = \{S_1, \ldots, S_\ell\}, \mathcal{T} = \{T_1, \ldots, T_\ell\}, k)$ where D is a directed acyclic graph, $\mathcal{S} \cup \mathcal{T}$ is a collection of disjoint subsets of V, called the terminal sets, a positive integer k. Additionally, no vertex in S_i for $1 \leq i < \ell$ has an incoming arc incident on it, and no vertex in T_i for $1 \leq i \leq \ell$ has an outgoing arc incident on it.
Parameter:	k
Question:	Does there exist a set $S \subseteq V_1$ of size at most k such that in the graph $D \setminus S$ there is no path from S_i to T_j for any $i \geq j$?

Given an instance $I = (D, V_1, V_2, \pi, k)$ of the ORDERED DAG BIPARTITION

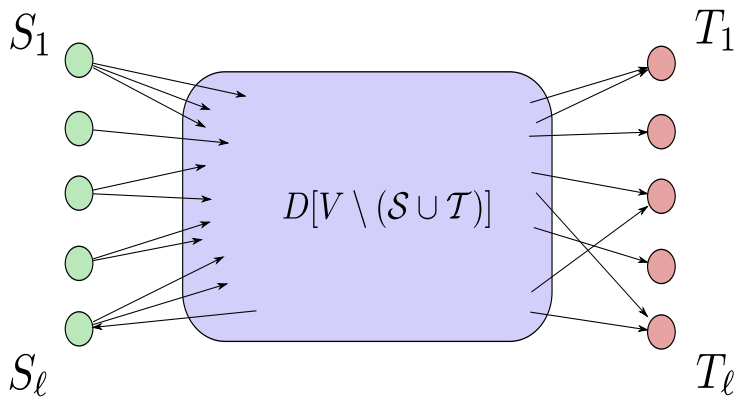

Figure 5.1: Example of an instance of SKEW MULTICUT.

FVS problem, let D_{skew} be the graph defined as before. It is not very hard to see that $I' = (D_{skew}, \{s_1, \ldots, s_l\}, \{t_1, \ldots, t_l\}, k)$ is now an instance of the SKEW MULTICUTproblem. Now, any path from v_i to v_j in D corresponds to a path from s_i to t_j in D_{skew} and vice versa. Therefore I is a YES instance of ORDERED DAG BIPARTITION FVS iff I' is a YES instance of SKEW MULTICUT. Hence, in order to solve the ORDERED DAG BIPARTITION FVS problem, it is sufficient to solve the SKEW MULTICUTproblem.

Theorem 5.3.2. ([13]) *The* SKEW MULTICUT *problem can be solved in time* $\mathcal{O}^*(4^k)$.

Given Theorem 5.3.2, since we run the algorithm for SKEW MULTICUT for every permutation of V_2, the time required to solve DAG BIPARTITION FVS is $\mathcal{O}^*(4^k.\ell!)$ where ℓ is the size of the partition V_2. The number of choices of size i for Y is $\binom{k+1}{i}$ and for each choice of Y, we run the algorithm for DAG BIPARTITION FVS with parameter $k - i$. Hence the time required to solve the DFVS problem is

$$\mathcal{O}^*(\Sigma_{i=0}^{k}\tbinom{k+1}{i}4^{k-i}(k+1-i)!) = \mathcal{O}^*(4^k k!).$$

In the rest of the chapter, we will give a proof of Theorem 5.3.2.

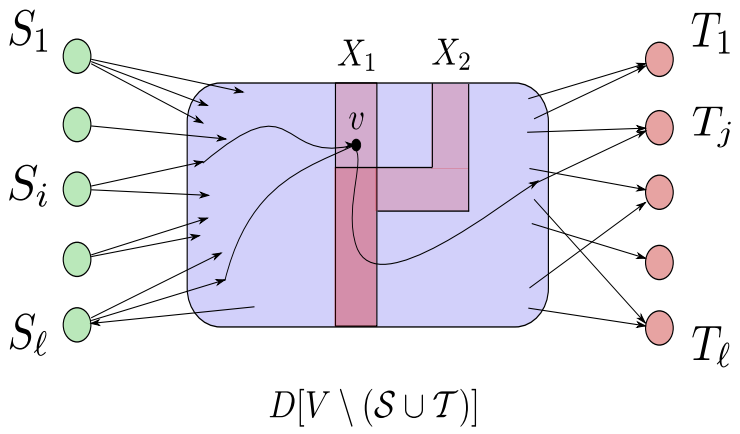

Figure 5.2: Illustration of the path from the vertex $v \in X_1$ to $\mathcal{T}$ in the graph $D \setminus X_2$.

5.4 Skew Multicut, Imporant Separators and the Algorithm

Lemma 5.4.1. *Let $(D = (V, A), \mathcal{S} = \{S_1, \ldots, S_\ell\}, \mathcal{T} = \{T_1, \ldots, T_\ell\}, k)$ be an instance of* Skew Multicut. *If it is a* Yes *instance then it has a solution $\hat{X}$ which contains an important $S_\ell - \mathcal{T}$ vertex separator.*

Proof. Let X be a solution for the given instance and let X_1 be a minimal part of X such that in $D \setminus X_1$ there are no $S_\ell - \mathcal{T}$ paths. If X_1 is an important $S_\ell - \mathcal{T}$ vertex separator, we are done. Suppose that this is not the case. Then, there is an $S_\ell - \mathcal{T}$ vertex separator X_2 which dominates X_1. Define $\hat{X} = (S \setminus X_1) \cup X_2$. We claim that $\hat{X}$ is a solution for the given instance. Clearly, $|\hat{X}| \leq |X|$ and hence $|\hat{X}| \leq k$. It remains to prove that $\hat{X}$ is a skew multicut for this instance.

Suppose that this is not the case and let P be a path from S_i to T_j in $D \setminus \hat{X}$ where $i \geq j$. Clearly such a path must contain a vertex v in $X_1 \setminus X_2$ (see Fig. 5.2). By the minimality of X_1, v has a neighbor in $R_D(S_\ell, X_1)$. Since $R_D(S_\ell, X_2)$ contains $R_D(S_\ell, X_1)$, and v is not in X_2, it must be the case that v is in $R_D(S_\ell, X_2)$. The existence of the path P implies the existence of a path from v to $\mathcal{T}$. But X_2 is an $S_\ell - \mathcal{T}$ vertex separator and by Lemma 3.3.9(a), there cannot be a path from

$v \in R_D(S_\ell, X_2)$ to $\mathcal{T}$. This concludes the proof of the Lemma. □

5.4.1 The Algorithm

Input : An instance $(D = (V, A), \mathcal{S} = \{S_1, \ldots, S_\ell\}, \mathcal{T} = \{T_1, \ldots, T_\ell\}, k)$ of SKEW MULTICUT
Output: A skew multicut of size at most k for the input instance if it exists and NO otherwise

1 **if** $k < 0$ **then return** NO
2 $i \rightarrow$ *largest such that* $\{T_1, \ldots, T_i\}$ *is reachable from* S_i.
3 *remove all vertices* S_j *and* T_j *where* $j \geq i$ *from the terminal set and rename the remaining terminals and update* ℓ *appropriately.*
4 *Compute a minimum size* $S_\ell - \mathcal{T}$ *vertex separator* X.
5 **if** $|X| > k$ **then return** NO
6 **else** *Compute the unique minimum size important* $S_\ell - \mathcal{T}$ *vertex separator* X^* *and select a vertex* $v \in X^*$
7 $Q_1 \leftarrow SKEW - MC(D \setminus \{v\}, \mathcal{S}, \mathcal{T}, k-1)$
8 **if** Q_1 *is not* NO **then return** $Q_1 \cup \{v\}$
9 $S_\ell \leftarrow R(S_\ell, X^*) \cup \{v\}$
10 $Q_2 \leftarrow SKEW - MC(D, \mathcal{S}, \mathcal{T}, k)$
11 **return** Q_2

Algorithm 5.4.1: Algorithm $SKEW - MC$ for RESTRICTED MULTIWAY CUT

The idea of the algorithm is as follows. Given an instance $(D, \mathcal{S}, \mathcal{T}, k)$ of SKEW MULTICUT, if $\mathcal{T}$ cannot be reached from S_ℓ, we can ignore the terminal sets S_ℓ and T_ℓ. Hence we find the largest i such that S_i has a path to the set $\{T_1, \ldots, T_i\}$. By Lemma 5.4.1 we know that we can assume that the minimal subset of the solution separating $\mathcal{T}$ from S_ℓ is an important $S_\ell - \mathcal{T}$ vertex separator. Hence, we simply guess the important $S_\ell - \mathcal{T}$ vertex separator using the algorithm $Find - IS_2$ (Algorithm 3.6.2), remove these vertices from the instance and recurse. For ease of presentation and better analysis of the algorithm, we embed the part where we guess the important separator, into the main algorithm instead of merely using it as a subroutine.

Correctness The Correctness of Step 1 is obvious. Step 5 is correct because the size of the minimum $S_\ell - \mathcal{T}$ vertex separator is a lower bound on the solution size. Steps 7 and 10 are merely part of guessing the important $S_1 - \mathcal{T}$ vertex separator (Lemma 3.3.10, Algorithm 3.6.2). Since we have embedded the algorithm enumerating all important $S_\ell - \mathcal{T}$ vertex separators into our algorithm for SKEW MULTICUT, we need to show that the instances on which the recursive calls of Steps 7 and 10 are made are valid instances of SKEW MULTICUT. The instance $(D \setminus \{v\}, \mathcal{S}, \mathcal{T}, k-1)$ is clearly a valid instance of SKEW MULTICUT since the input instance $(D, \mathcal{S}, \mathcal{T}, k)$ was a valid instance. Now, when we add the vertices of $R(S_\ell, X^*) \cup \{v\}$ to S_ℓ, we will not violate any conditions on the input since S_ℓ is allowed to have incoming arcs by the definition of the SKEW MULTICUT problem. We know from Lemma 5.4.1 that, if there is a solution, there is one which contains an important $S_\ell - \mathcal{T}$ vertex separator and hence guessing the important $S_\ell - \mathcal{T}$ vertex separator in steps 7 and 10 is also correct.

Running Time. To analyze the algorithm we define the search tree $\mathbb{T}(D, \mathcal{S}, \mathcal{T}, k)$ resulting from a call to $SKEW - MC(D, \mathcal{S}, \mathcal{T}, k)$ inductively as follows. The tree $\mathbb{T}(D, \mathcal{S}, \mathcal{T}, k)$ is a rooted tree whose root node corresponds to the instance $(D, \mathcal{S}, \mathcal{T}, k)$. If $SKEW - MC(D, \mathcal{S}, \mathcal{T}, k)$ does not make a recursive call then $(D, \mathcal{S}, \mathcal{T}, k)$ is said to be the only node of this tree. If it does make recursive calls, the children of $(D, \mathcal{S}, \mathcal{T}, k)$ correspond to the instances given as input to the recursive calls made inside the current procedure call. The subtree of $\mathbb{T}(D, \mathcal{S}, \mathcal{T}, k)$ rooted at a child node $(D', \mathcal{S}', \mathcal{T}', k')$ is the search tree $\mathbb{T}(D', \mathcal{S}', \mathcal{T}', k')$.

Given an instance $I = (D, \mathcal{S}, \mathcal{T}, k)$, we prove by induction on $\mu(I) = 2k - \lambda_D(S_1, \mathcal{T})$ that the number of leaves of the tree $\mathbb{T}(I)$ is bounded by $max\{2^{\mu(I)}, 1\}$. In the base case, if $\mu(I) < k$, then $\lambda(S_1, \mathcal{T}) > k$, in which case we return NO and the number of leaves is 1. We can assume without loss of generality that $\lambda(S_1, \mathcal{T}) > 0$, since Steps 2 and 3 of the algorithm will ensure that this happens. Now, assume that $\mu(I) \geq k$ and our claim holds for all instances I' such that $\mu(I') < \mu(I)$.

The children I_1 and I_2 of this node correspond to the recursive calls made in Steps 7 and 10. But in these two cases, as seen in the proof of Lemma 3.3.10, $\mu(I_1), \mu(I_2) < \mu(I)$ and hence applying induction hypothesis on the two child nodes and summing up the number of leaves in the subtrees rooted at each, we

can bound the number of leaves in the sub tree of I by $2^{\mu(I)}$.

The time spent at a node I is bounded by the time required to compute the unique smallest $S_1-\mathcal{T}$ vertex separator in D which takes $\mathcal{O}(n^5)$ time (Lemma 3.6.1(b)). Along any path from the root to a leaf, at any internal node, the size of the set S_1 increases or a vertex is removed from the graph. Hence the length of any root to leaf path is at most n. Therefore the running time of this algorithm is $\mathcal{O}^*(4^k)$.

5.5 Summary

In this Chapter, we applied the notion of important separators to present the algorithm of Chen et al. [12] for DFVS. We described an application of the method of iterative compression to this problem and showed that it can be reduced to the SKEW MULTICUT problem where we are required to separate some vertices in a particular way.

We then described how one may explicitly use the notion of important separators to solve this problem. We showed that the part of the solution disconnecting the rest of the terminals from one may be assumed to be an important separator, and the algorithm worked by repeatedly selecting a terminal and isolating the rest of the terminals from this terminal by branching on an important vertex separator separating the rest of the terminals from this one.

Open Questions and Future Directions. The main open questions regarding the DFVS problem are coming up with a faster FPT algorithm (or prove that it is unlikely to exist) and proving upper or lower bounds on polynomial kernels for this problem.

6

Almost 2-SAT

6.1 Introduction

An algorithm for testing whether a given 2-sat formula is satisfiable, is one of the fundamental polynomial time algorithms [3] and has been used crucially as a subroutine in several polynomial time algorithms [26, 27]. Similarly, a parameterized analogue of 2-SAT – ALMOST 2-SAT, is turning out to be extremely useful in the context of designing parameterized algorithms. In ALMOST 2-SAT (or 2-ASAT), we are given a 2-SAT formula F, a positive integer k and the objective is to check whether there exists at a set of most k clauses whose deletion from ϕ makes the resulting formula satisfiable. The ALMOST 2-SAT problem was introduced in [37] and the fixed parameter tractability of this problem was a long standing open problem. In [35], this problem was shown to be a generalization of the parameterized ODD CYCLE TRANSVERSAL problem. An algorithm running in time $\mathcal{O}^*(15^k)$ was given for this problem in [44], proving that it is FPT parameterized by the solution size. This algorithm implicitly used the notion of important separators. In this Chapter, we present this algorithm by explicitly using the machinery of important separators.

Organization of the Chapter In Section 6.2 we introduce some definitions and notations we will be following in our discussion. In Section 6.3 we describe the method of iterative compression as applied to the 2-ASAT problem and introduce an annotated variant of the 2-ASAT problem which will be our main focus. In Section 6.4, we characterize the instances of this annotated variant as graphs

with some separation property, which allows us to tap in to the machinery of important separators. In Section 6.5 we discuss the part played by important separators in solving this problem and present an FPT algorithm for this problem using important separators.

6.2 Preliminaries

6.2.1 2-CNF formulas

A CNF formula F is called a 2-CNF formula if every clause in F has at most 2 literals. In this chapter, we assume that every clause is of length exactly 2 since any clause with a single literal l can be represented as $(l \vee l)$ and also that every clause is distinct. In [44] it is shown that this assumption is without loss of generality. We also assume that every clause $C = (l_1 \vee l_2)$ has a fixed ordering among its literals and without loss of generality we will refer to l_1 as the first literal of C and l_2 as the second literal of C.

Given a set S_C of clauses of F, we denote by $F \setminus S_C$ the formula obtained from F by deleting the clauses of S_C.

Given a set S_V of variables of F, we denote by $F \setminus S_V$ the formula obtained from F by deleting every clause of F which contains a literal involving a variable from S_V.

Let F be a 2-CNF formula, S_C be a set of clauses of F, C be a clause of F, L be a set of literals and l be a single literal of F. Then we denote by $Clause(F)$ the set of clauses of F and $Var(F), Var(S_C), Var(C), Var(L)$ and $Var(l)$ denote the set of variables whose literal appears in F, S_C, L and l respectively.

A set L of literals of F is called non-contradictory if L does not contain both a literal l and its negation $\bar{l}$. The set $\bar{L}$ is the set of negations of the literals in L. A literal l is said to satisfy a clause $C = (l_1 \vee l_2)$ if $l = l_1$ or $l = l_2$. For example, if $C = (x_1 \vee \bar{x}_2)$ where x_1 and x_2 are variables of the formula, we say that C is satisfied by x_1 or $\bar{x}_2$. To help the reader in understanding this notation, we note that when we say that a literal satisfies a clause, we imply that any assignment (in the classical sense) for the formula F which sets this literal to true, satisfies this clause.

A set L of literals is said to satisfy a clause C if there is a literal $l \in L$ such that l satisfies C.

Given a 2-CNF formula F, a non-contradictory set of literals L such that $Var(L) = Var(F)$, is said to be an assignment for F. An assignment for F which satisfies every clause of F is said to be a satisfying assignment for F. A 2-CNF formula F is said to be satisfiable if there exists a satisfying assignment for F.

Let F be a 2-CNF formula, L be a set of literals. We say that F is satisfiable with respect to L iff $F \wedge \bigwedge_{l \in L} l$ is satisfiable. In other words, there is a satisfying assignment for F which contains L.

We say that $SWRT(F, L)$ is true (false) if F is satisfiable(not satisfiable) with respect to L. For ease of presentation, we abuse notation by representing a set of literals L which consists of a single literal l as l instead of $\{l\}$.

6.2.2 Implication Graphs, Walks and Paths

Given a 2-CNF formula F we associate a canonical directed graph $D(F)$ with this formula which we will call the *implication graph* of F. The graph $D(F)$ has $2|Var(F)|$ vertices, one for each possible literal involved in the formula. For the sake of convenience the vertices will be named $x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$ where $x_1, \ldots, x_n$ are the variables of F. For every clause $C = (l_1 \vee l_2) \in Clause(F)$, $D(F)$ has two arcs $e_C = (\bar{l}_1, l_2)$ and $\hat{e}_C = (\bar{l}_2, l_1)$. The implied meaning of these arcs is the following. Consider any assignment which satisfies this clause. Suppose this assignment contains $\bar{l}_1$, then it must be the case that it contains l_2. Similarly, if this assignment contains $\bar{l}_2$, then it must be the case that it contains l_1.

Since there are no duplicate clauses in F, $D(F)$ is a simple directed graph. Given a set S of clauses, we denote by $A(S)$ the set of arcs of $D(F)$ which correspond to a clause in S. Given a set S_a of arcs of $D(F)$ we denote by $Clause(S_a)$ the set of clauses of F which have a corresponding arc in S_a.

Let $w = l_1, \ldots, l_t$ be a walk in the graph $D(F)$. Then we say that w is a walk from l_1 to l_t and we represent by $Rev(w)$ the walk $\bar{l}_t, \ldots, \bar{l}_1$. Given a set of literals L such that $l_1 \in L$ we say that w is a walk from L. Consider another walk $w_1 = l_t, \ldots, l_{t'}$. Then we represent by $w + w_1$ the concatenated walk $l_1, \ldots, l_t, \ldots, l_{t'}$.

Let $w = l_1, \ldots, l_t$ be a walk in the graph $D(F)$. We define a prefix walk of w as the sub walk of w from l_1 to some l_i, $1 \leq i \leq t$. We define a suffix walk of w as

the sub walk of w from some l_i to l_t, $1 \leq i \leq t$. Note that w is both a prefix walk and suffix walk of itself.

A walk w in $D(F)$ is called non trivial if it contains more than one vertex and trivial otherwise.

6.3 Iterative Compression for 2-ASAT

Given an instance (F, k) of 2-ASAT, we fix an arbitrary ordering of the clauses of F as $C_1, \ldots, C_m$, define the 2-CNF formula F_i as $\bigwedge_{j=1}^{i} C_j$ and define the instances $I_1, \ldots, I_m$ where $I_i = (F_i, k)$. One by one, we iterate through the instances I_i starting with $i = k + 1$ and with the help of a known (bigger) solution try to find a solution $\hat{S}_i$ of size at most k for the i^{th} instance. Formally we define the *compression version* of our problem as follows.

PROBLEM I2	
Input:	(F, S, k), where F is a 2-CNF formula and S a set of clauses of size at most $k+1$ such that $F \setminus S$ is satisfiable, k a positive integer
Parameter:	k
Question:	Does there exist a set $\hat{S}$ of at most k clauses such that $F \setminus \hat{S}$ is satisfiable?

We will reduce the 2-ASAT problem to at most m instances of PROBLEM I2 as follows.

Clearly, the set of clauses $S_{k+1} = \{C_1, \ldots, C_{k+1}\}$ is a solution of size at most $k + 1$ for the instance I_{k+1}. It is also easy to see that if $\hat{S}_{i-1}$ is a solution of size at most k for instance I_{i-1}, then the set $S_{i-1} \cup C_i$ is a solution of size at most $k+1$ for the instance I_i. Hence, we start off the iteration with the instance (F_{k+1}, S_{k+1}, k) of PROBLEM I2 and find a solution of size at most k for this instance. If there is such a solution $\hat{S}_{k+1}$, we set $S_{k+2} = \hat{S}_{k+1} \cup \{C_{k+2}\}$ and look for a solution of size at most k for the instance I_{k+2} and so on. If during any intermediate iteration, the corresponding instance does not have a solution of size k, it implies that the original instance of 2-ASAT is also a NO instance. Finally the solution for the

original input instance will be $\hat{S}_m$. Since there can be at most m iterations, the total time taken is bounded by m times the time required to solve PROBLEM I2.

Now we present the algorithm for PROBLEM I2 as follows. Let the input instance be $I = (F, S, k)$. We guess a subset Y of S of size at most k to be in our new solution and set $N := S \setminus Y$. Now, for every clause in N, we guess the literal which is going to satisfy this clause. Let the set of literals thus guessed be L. Since $F \setminus S$ is satisfiable it has a satisfying assignment P. Define $L_1 = L \cap P$ and $L_2 = L \setminus L_1$. We have thus reduced this problem to checking if the instance $(F' = F \setminus Y, k)$ has a set of S' of at most k clauses such that $F' \setminus S'$ has a satisfying assignment containing $L_1 \uplus L_2$. The reason for partitioning L into L_1 and L_2 is so that we can propose an annotated variant of this problem with some additional structure imposed on the input. We formally define this annotated variant as follows.

PROBLEM I1	
Input:	(F, L_1, L_2, k), where F is a 2-CNF formula, L_1 and L_2 are sets of literals, F has a satisfying assignment containing L_1, k a positive integer
Parameter:	k
Question:	Does there exist a set S' of size at most k such that $F \setminus S'$ has a satisfying assignment containing $L_1 \cup L_2$?

2-ASLASAT	
Input:	(F, L, l, k), where F is a 2-CNF formula, L is a set of literals, l is a single literal, F has a satisfying assignment containing L, k a positive integer
Parameter:	k
Question:	Does there exist a set S' of size at most k such that $F \setminus S'$ has a satisfying assignment containing $L \cup \{l\}$?

Lemma 6.3.1. ([44]) *There is a polynomial time parameter preserving reduction from* PROBLEM I1 *to* 2-ASLASAT

Theorem 6.3.2. ([44]) 2-ASLASAT*can be solved in time* $\mathcal{O}^*(5^k)$.

The number of possibilities for the pair (Y, L) is 3^{k+1} since any clause $C = (l_1 \vee l_2)$ in S can either be part of the new solution, or be satisfied by l_1 or be satisfied by l_2. Hence, given Theorem 6.3.2 and Lemma 6.3.1, it is easy to see that we can solve 2-ASAT in time $\mathcal{O}^*(5^k \cdot 3^k) = \mathcal{O}^*(15^k)$. In this Chapter, our analysis of the algorithm will be less intricate than that seen in [44]. This is in order to focus our attention more on the role played by important separators in our algorithm. As a result, we will prove the following theorem in this chapter.

Theorem 6.3.3. 2-ASLASAT *can be solved in time* $\mathcal{O}^*(9^k)$.

Given Theorem 6.3.3 and Lemma 6.3.1, it is easy to see that we can solve 2-ASAT in time $\mathcal{O}^*(9^k \cdot 3^k) = \mathcal{O}^*(27^k)$.

6.4 2-ASLASAT as a Directed Graph Separation Problem

In this section, we model the 2-ASLASAT problem as a problem of separating certain kinds of vertices from certain other kinds of vertices in a directed graph. We begin with some observations that follow easily from the definitions.

Observation 6.4.1. *Let F be a 2-CNF formula and L be a set of literals such that $SWRT(F, L)$ is false. Then, for any $L' \supseteq L$, $SWRT(F, L')$ is also false.*

Observation 6.4.2. *Let F be a 2-CNF formula and L be a set of literals such that $SWRT(F, L)$ is true. Then, for any $L' \subseteq L$, $SWRT(F, L')$ is also true.*

Lemma 6.4.3. ([44]) *Let F be a 2-CNF formula and $w = l_1, \ldots, l_t$ be a non trivial walk in $D(F)$. Then $SWRT(F, \{l_1, \bar{l}_t\})$ is false. In particular, if $l_1 = \bar{l}_t$ then $SWRT(F, l_1)$ is false.*

Proof. We prove the statment of the lemma by induction on t. For the base case, $t = 2$. In this case $w = l_1, l_2$. The clause corresponding to this edge is $(\bar{l}_1 \vee l_t)$ which cannot be satisfied by $\{l_1, \bar{l}_2\}$. We now assume that $t > 2$ and the statement of the Lemma holds for all walks $l_1, \ldots, l_{t'}$ where $t' < t$. Then any walk $w = l_1, \ldots, l_t$ can be written as $w' + (l_{t-1}, l_t)$ where $w' = l_1, \ldots, l_{t-1}$ and by the induction hypothesis, $SWRT(F, \{l_1, l_{t-1}\})$ is false. This implies that any satisfying assignment for F which contains l_1 must contain l_{t-1} and such an

assignment cannot satisfy the clause $(\bar{l}_{t-1} \vee l_t)$ unless it contains l_t. Thus we have that $SWRT(F, \{l_1, \bar{l}_t\})$ is false. □

Lemma 6.4.4. ([44]) *Let F be a 2-CNF formula, L be a set of literals such that $SWRT(F, L)$ is true, w be a walk of $D(F)$ from L and $C = (l_1 \vee l_2)$ be a clause in $Clause(F)$ such that the walk w contains the arc e_C ($\hat{e}_C$). Then for any walk w_1 of $D(F)$ from L, w_1 cannot contain the arc $\hat{e}_C$ (respectively e_C). In particular, any walk from L such that $SWRT(F, L)$ is true, can contain at most one of the two edges $e_C, \hat{e}_C$ for every clause C.*

Proof. Suppose w contains the arc e_C. Suppose that there is also a walk w_1, which contains the arc $\hat{e}_C$. Now w has a prefix walk $w' + e_C$ where w' is a walk from $l_x \in L$ to $\bar{l}_1$. Also w_1 has a prefix walk $w'_1 + \hat{e}_C$ which is a walk from $l_y \in L$ to $\bar{l}_2$. Hence the walk $w' + e_C + Rev(w'_1)$ is a walk from l_x to $\bar{l}_y$ and by Lemma 6.4.3 $SWRT(F, \{l_x, l_y\})$ is false which contradicts Observation 6.4.2. The case when w uses the arc $\hat{e}_C$ can be handled analogously. □

6.4.1 Characterization of Extendable Satisfying Assignments

Theorem 6.4.5. ([44]) *Let (F, L, l) be an instance of the* 2-ASLASAT *problem. Then $SWRT(F, L \cup \{l\})$ is false if and only if $D(F)$ has a walk from $L \cup \{l\}$ to $\bar{l}$.*

Proof. For the if direction, consider a walk w from l_1 to $\bar{l}$ in $D(F)$. By Lemma 6.4.3, $SWRT(F, \{l_1, l\})$ is false. Hence if $l_1 \in L \cup \{l\}$, applying Observation 6.4.1, $SWRT(F, L \cup \{l\})$ is false.

For the converse, assume that $SWRT(F, L \cup \{l\})$ is false. Define the set I to be the set of vertices reachable from l in $D(F)$. Note that I is also a set of literals.

(a) I is contradictory. Then there are literals l_1 and $\bar{l}_1$ which are both reachable from l. Let w_1 be a walk from l to l_1 and w_2 be a walk from l to $\bar{l}_1$. Then $w_1 + Rev(w_2)$ is a walk from l to $\bar{l}$.

(b) I intersects $\bar{L}$. Then there is a walk w from l to $\bar{l}_x$ where $l_x \in L$. Then, $Rev(w)$ is a walk from L to $\bar{l}$.

(c) I is neither contradictory nor intersects $\bar{L}$. In this case we will give a satisfying assignment for F which contains $L \cup \{l\}$ which contradicts the assumption that $SWRT(F, L \cup \{l\})$ is false. Let P be a satisfying assignment of F which contains L. We know that such an assignment exists since (F, L, l) is a valid instance of 2-ASLASAT. We let P' be the restriction of P to $Var(F) \setminus Var(I)$ and we claim that $P_1 = P' \cup I$ is a satisfying assignment for F containing $L \cup \{l\}$. It is clear that P_1 is a non contradictory assignment for F and by definition, P_1 contains l. Also P_1 cannot contradict L since otherwise it would imply that $L \cap I \neq \phi$ which contradicts our assumption. Hence P_1 indeed contains $L \cup \{l\}$.

Now it remains to prove that for every clause C of F, P_1 satisfies C. Consider a clause $C = (l_1 \vee l_2)$. If C is satisfied by I, it is clearly satisfied by P_1. Hence we assume that C is not satisfied by I. First we consider the case when $Var(C) \cap Var(I) \neq \phi$. In particular, suppose $\bar{l}_1 \in I$. Then, l_2 is reachable from $\bar{l}_1$ due to the arc $(\bar{l}_1, l_2)$ and hence reachable from l, which implies that $l_2 \in I$ thus satisfying C which is a contradiction. Similarly if $\bar{l}_2 \in I$, then $l_1 \in I$ which implies that I satisfies C, a contradiction. Now we consider the case when $Var(C) \cap Var(I) = \phi$. In this case the restriction of P to $Var(C)$ is the same as the restriction of P' to $Var(C)$ which implies that P' satisfies C and hence P_1 satisfies C. □

Given Theorem 6.4.5, we may be tempted to just find a minimum size $(L \cup \{l\}) - \bar{l}$ cut in the graph $D(F)$ and claim that the corresponding clauses form an optimal solution for the given instance of 2-ASLASAT. But in this graph, every clause of F has two corresponding arcs. Hence, the relation between the size of the "solution" for the instance of 2-ASLASAT obtained from a minimum size $(L \cup \{l\}) - \bar{l}$ cut in the graph $D(F)$ and the size of the optimal solution for the instance may differ by as much as a factor of 2 and hence this direct approach will not lead to an algorithm.

Lemma 6.4.6. ([44]) *Let (F, L, l, k) be an instance of* 2-ASLASAT *such that in the graph $D(F)$ there is no walk from L to $\bar{l}$. If there is a walk w from l to $\bar{l}$ in $D(F)$, then there is a clause $C = (l_1 \vee l_2)$ of F such that $SWRT(F, L \cup \{l_1\})$ is true, $SWRT(F, L \cup \{l_2\})$ is true, and there exist walks w_1 and w_2 from l_1 to $\bar{l}$ and l_2 to $\bar{l}$.*

Proof. Let P be a satisfying assignment of F containing L. We know that such an assignment exists by the definiton of the 2-ASLASAT problem and let P' be the restriction of P to $Var(w)$. Define F' to be the restriction of F to $Clause(w)$. Clearly P' is a satisfying assignment for F'. But by Lemma 6.4.3, $SWRT(F, l)$ is false and hence P' cannot contain l. Now, $\bar{P}'$ does contain l. But since $SWRT(F, l)$ is false, it must be the case that $\bar{P}'$ leaves some clause $C = (l_1 \vee l_2)$ of F' unsatisfied. This can happen only if $\bar{l}_1, \bar{l}_2 \in \bar{P}'$ which implies that $l_1, l_2 \in P'$ which is contained in P. Thus P is a satisfying assignment for F containing $L \cup \{l_1, l_2\}$.

Now, if w uses the arc $e_C = (\bar{l}_1, l_2)$ then w has a suffix walk w' from l_2 to $\bar{l}$ and a prefix walk w'' from l to $\bar{l}_1$ which implies that $Rev(w'')$ is a walk from l_1 to $\bar{l}$. □

6.5 2-ASLASAT and Important Separators

Given a solution S for an instance (F, L, l, k) of 2-ASLASAT, by Lemma 6.4.5 the graph $D(F) \setminus A(S)$ has no walk from $L \cup \{l\}$ to $\bar{l}$. Let $A_{L-\bar{l}}(S)$ be a minimal subset of $A(S)$ such that $D(F) \setminus A_{L-\bar{l}}(S)$ has no walk from L to $\bar{l}$.

Lemma 6.5.1. *Let (F, L, l, k) be an instance of* 2-ASLASAT*). If it is a* YES *instance then it has a solution $\hat{S}$ which contains an important $L - \bar{l}$ arc separator in $D(F)$.*

Proof. Consider a solution S for this instance. By Lemma 6.4.5 the graph $D(F) \setminus A(S)$ has no walk from $L \cup \{l\}$ to $\bar{l}$. Let $A_{L-\bar{l}}(S)$ be a minimal subset of $A(S)$ such that $D(F) \setminus A_{L-\bar{l}}(S)$ has no walk from L to $\bar{l}$, and let $S^1_a = A_{L-\bar{l}}(S)$. If S^1_a is an important $L - \bar{l}$ arc separator in $D(F)$, we are done. Suppose that this is not the case. Then, since S^1_a is a minimal $L-\bar{l}$ arc separator which is not important, there is another minimal $L-\bar{l}$ arc separator S^2_a such that $|S^2_a| \leq |S^1_a|$ and $R_{D(F)}(L, S^1_a) \subset R_{D(F)}(L, S^2_a)$. We will prove that replacing the clauses corresponding to S^1_a with those corresponding to S^2_a results in a solution which satisfies the statement of the Lemma. We define $\hat{S} = (S \setminus Clause(S^1_a)) \cup Clause(S^2_a)$ and claim that $\hat{S}$ is a solution for this instance which satisifes the conclusions of the Lemma. By the definition of $\hat{S}$, it is true that it contains an important $L-\bar{l}$ arc separator in $D(F)$. It remains to prove that size of $\hat{S}$ is at most k and that $SWRT(F \setminus \hat{S}, L \cup \{l\})$ is true.

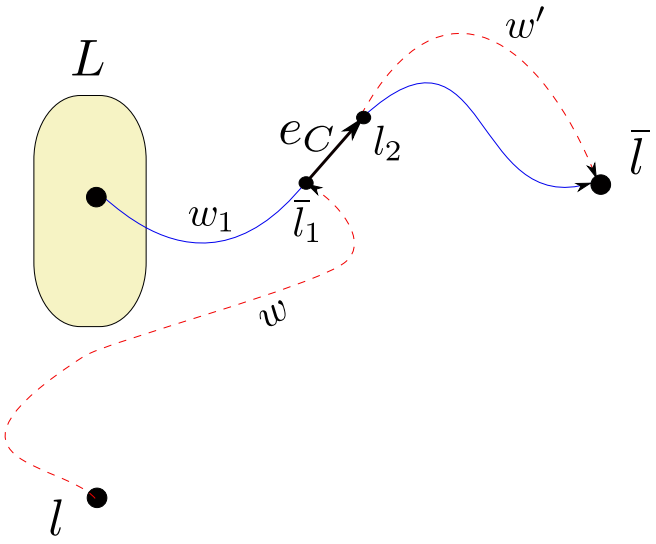

Figure 6.1: Illustration of the walks w, w_1 and w'.

We will first prove that $|Clause(S_a^2)| \leq |Clause(S_a^1)|$. In order to prove this, it is sufficient to prove that $S_a^2 = |Clause(S_a^2)|$ and $S_a^1 = |Clause(S_a^1)|$. In other words, we will show that S_a^2 and S_a^1 both contain at most one arc corresponding to any clause. By minimality of S_a^2 and S_a^1, every arc in S_a^2 or S_a^1 is reachable from L in $D(F)$ and by Lemma 6.4.4 it cannot be the case that there are two arcs in S_a^2 or S_a^1 which correspond to the same clause of F. Hence $S_a^2 = |Clause(S_a^2)|$ and $S_a^1 = |Clause(S_a^1)|$ which implies that $|\hat{S}| \leq |S| \leq k$.

We now show that $SWRT(F \setminus \hat{S}, L \cup \{l\})$ is true. By Lemma 6.4.5 it is enough for us to show that $D(F) \setminus A(\hat{S})$ does not have a walk from $L \cup \{l\}$ to $\bar{l}$. Suppose there is a walk $w = l_1, \ldots, l_t$ from $L \cup \{l\}$ to $\bar{l}$ in $D(F) \setminus A(\hat{S})$. Then w must be a walk from l to $\bar{l}$ since $A(\hat{S})$ contains S_a^2 which is an $L - \bar{l}$ arc separator. Since S was a solution and $\hat{S}$ is not, w must use one of the two arcs corresponding to a clause $C = (l_x \vee l_y) \in S \setminus \hat{S}$. It is clear that such an arc is contained in S_a^1 but not in S_a^2.

We know that S_a^1 contains at most one arc corresponding to C and so without loss of generality let this arc be e_C. Now w may use the arc e_C or $\hat{e}_C$ or both. If w uses the arc $\hat{e}_C$ but not the arc e_C, then $Rev(w)$ is a walk from l to $\bar{l}$ which uses the arc e_C and we set $w := Rev(w)$. Hence we assume without loss of generality that w uses the arc e_C.

We know that there is an i such that $w = l_1, \ldots, l_i, l_{i+1}, \ldots, l_t$ where $l_i = \bar{l}_x$ and $l_{i+1} = l_y$ and let w' be the suffix walk $l_i, l_{i+1}, \ldots, l_t$ (see Fig. 6.1). Now, since S_a^1 was minimal, e_C is reachable from L and hence there is a walk w_1 from L to $\bar{l}_x$ which lies entirely inside the graph induced on $R_{D(F)}(L, S_a^1)$ and hence also lies entirely inside the graph induced on $R_{D(F)}(L, S_a^2)$ implying that it does not use an arc in S_a^2. But now, $w_1 + w'$ is a walk from L to $\bar{l}$ in the graph $D(F \setminus \hat{S})$ which is a contradiction. This completes the proof of the Lemma. □

6.5.1 The Algorithm

Input : An instance (F, L, l, k) of 2-ASLASAT
Output: A solution of size at most k for the instance (F, L, l, k) if it exists and NO otherwise

1 **if** $k < 0$ **then return** NO
2 *Compute a mimimum size* $L - \bar{l}$ *arc separator* S_a *in the directed graph* $D(F)$
3 **if** $|S_a| = 0$ **then**
4 **if** *there an* l *to* $\bar{l}$ *walk* **then**
5 *find the clause* $C = (l_1 \vee l_2)$ *given by Lemma 6.4.6*
6 $S_1 \leftarrow Solve - ASLASAT(F \setminus C, L, l, k-1)$
7 **if** S_1 *is not* NO **then return** $S_1 \cup \{C\}$
8 $S_2 \leftarrow Solve - ASLASAT(F, L \cup \{l_1\}, l, k)$
9 **if** S_2 *is not* NO **then return** S_2
10 $S_3 \leftarrow Solve - ASLASAT(F, L \cup \{l_2\}, l, k)$
11 **return** S_3
12 **end**
13 **else return** ϕ
14 **end**
15 **if** $|S| > k$ **then return** NO
16 **else** *Compute the unique minimum size important* $L - \bar{l}$ *arc separator* S_a^* *in* $D(F)$ *and select an arc* $e = (w, z) \in S_a^*$
17 $S_4 \leftarrow Solve - ASLASAT(F \setminus Clause(e), L, l, k-1)$
18 **if** S_4 *is not* NO **then return** $S_4 \cup \{Clause(e)\}$
19 $S_5 \leftarrow Solve - ASLASAT(F, R_{D(F)}(L, S_a^*) \cup \{z\}, l, k)$
20 **return** S_5

Algorithm 6.5.1: Algorithm $Solve - ASLASAT$ for 2-ASLASAT

The idea of the algorithm is as follows. Given an instance (F, L, l, k) of 2-

ASLASAT, by Lemma 6.4.5 we know that it is enough for us to find a set S_a of arcs of $D(F)$ such that $|Clause(S_a)| \leq k$ and $D(F) \setminus S_a$ has not paths from $L \cup \{l\}$ to $\bar{l}$. If there is no path from $L \cup \{l\}$ to $\bar{l}$ in $D(F)$, the empty set is the solution. Suppose there is a path from L to $\bar{l}$ in $D(F)$. By Lemma 6.5.1 we can assume that $A_{L-\bar{l}}(S)$ is an important $L - \bar{l}$ arc separator in $D(F)$ where S is a solution for the input instance. Hence we guess an important $L - \bar{l}$ arc separator and remove the corresponding clauses from the formula, reduce k accordingly, and continue. Suppose there are no paths from L to $\bar{l}$ in $D(F)$ but a path from l to $\bar{l}$, we know by Lemma 6.4.6 that there is a clause $(l_1 \vee l_2)$ such that there is a satisfying assignment for F containing $L \cup \{l_1, l_2\}$. Hence we branch into the following three exhaustive cases. Either we delete the clause $(l_1 \vee l_2)$ from the instance or pick l_1 as part of the satisfying assignment or pick l_2 as part of the satisfying assignment. For ease of presentation and better analysis of the algorithm, we embed the part where we guess the important separator, into the main algorithm instead of merely using it as a subroutine.

Correctness The Correctness of Step 1 is obvious. Steps 3-14 are correct due to Lemma 6.4.6 and the fact that the three cases are exhaustive. Step 15 is correct because the size of the minimum $L - \bar{l}$ arc separator in $D(F)$ is a lower bound on the solution size. Steps 17 and 19 are merely part of guessing the important $L - \bar{l}$ arc separator (Lemma 3.4.11, Algorithm 3.6.3). Since we have embedded the algorithm enumerating all important $L - \bar{l}$ arc separators into our algorithm, we need to show that the instances on which the recursive calls of Steps 17 and 19 are made are valid instances of 2-ASLASAT. The instance $(F \setminus Clause(e), L, l, k-1)$ is a valid instance of 2-ASLASATsince (F, L, l, k) was a valid instance. Now, consider a satisfying assignment P of F which contains L. By Lemma 6.4.3 it must be the case that P also contains any literal reachable from L in $D(F)$. Hence the instance $(F, R_{D(F)}(L, S_a^*) \cup \{z\}, l, k)$ is also a valid instance of 2-ASLASAT. We know from Lemma 6.5.1 that if there is a solution, there is one which contains an important $L - \bar{l}$ arc separator in $D(F)$. Hence guessing the important $L - \bar{l}$ arc separator in steps 17 and 19 is also correct.

Running Time. To analyze the algorithm we define the search tree $\mathbb{T}(F, L, l, k)$ resulting from a call to $Solve - ASLASAT(F, L, l, k)$ inductively as follows. The

tree $\mathbb{T}(F, L, l, k)$ is a rooted tree whose root node corresponds to the instance (F, L, l, k). If $Solve - ASLASAT(F, L, l, k)$ does not make a recursive call then (F, L, l, k) is said to be the only node of this tree. If it does make recursive calls, the children of (F, L, l, k) correspond to the instances given as input to the recursive calls made inside the current procedure call. The subtree of $\mathbb{T}(F, L, l, k)$ rooted at a child node (F', L', l, k') is the search tree $\mathbb{T}(F', L', l, k')$.

Given an instance $I = (F, L, l, k)$, we prove by induction on $\mu(I) = 2k - \zeta_{D(F)}(L, \bar{l})$ that the number of leaves of the tree $\mathbb{T}(I)$ is bounded by $max\{3^{\mu(I)}, 1\}$. In the base case, if $\mu(I) < k$, then $\zeta(L, \bar{l}) > k$ in which case the number of leaves is 1. Assume that $\mu(I) \geq k$ and our claim holds for all instances I' such that $\mu(I') < \mu(I)$.

Suppose $\zeta(L, \bar{l}) = 0$. In this case, the children I_1, I_2 and I_3 of this node correspond to the recursive calls made in Steps 6, 8 and 10 respectively. It is easy to see that $\mu(I_1) < \mu(I)$. By Lemma 6.4.6 there are paths from l_1 to $\bar{l}$ and from l_2 to $\bar{l}$. Hence, $\zeta(L \cup \{l_1\}, \bar{l}) > 0$ and $\zeta(L \cup \{l_2\}, \bar{l}) > 0$. This implies that $\mu(I_2), \mu(I_3) < \mu(I)$. By the induction hypothesis, the number of leaves in the search trees rooted at I_1, I_2 and I_3 are at most $3^{\mu(I_1)}, 3^{\mu(I_2)}$ and $3^{\mu(I_3)}$ respectively. Hence the number of leaves in the search tree rooted at I is at most $3 \cdot 3^{\mu(I)-1} = 3^{\mu(I)}$.

Suppose $\zeta(L, \bar{l}) > 0$. In this case, the children I_1 and I_2 of this node correspond to the recursive calls made in Steps 17 and 19. But in these two cases, as seen in the proof of Lemma 3.4.11, $\mu(I_1), \mu(I_2) < \mu(I)$ and hence applying induction hypothesis on the two child nodes and summing up the number of leaves in the sub trees rooted at each, we can bound the number of leaves in the sub tree of I by $3^{\mu(I)}$.

Hence the number of leaves of the search tree $\mathbb{T}$ rooted at the input instance $I = (F, L, l, k)$ is $3^{\mu(I)} \leq 3^{2k}$. The time spent at a node I is bounded by the time required to compute the unique smallest $L - \bar{l}$ arc separator in $D(F)$ which takes $\mathcal{O}(n^5)$ time (Lemma 3.6.1(c)). Along any path from the root to a leaf, at any internal node, the size of the set L increases or a clause is removed from the formula. Hence the length of any root to leaf path is at most $max\{m, n\}$. Therefore the running time of this algorithm is $\mathcal{O}^*(9^k)$.

A more intricate analysis of the algorithm in [44] shows that it runs in time $\mathcal{O}^*(5^k)$.

6.6 Summary

In this Chapter, we described an $\mathcal{O}^*(C^k)$ algorithm where C is a constant, for the Almost 2 Sat problem using important separators. We first applied iterative compression and showed that it is sufficient to give a fast FPT algorithm for an annotated variant, 2-ASLASAT. In order to solve this problem, we first modelled it as a separation problem on directed graphs and proved a characterization of instances which have an extendable satisfying assignment. We then showed that we can use important separators to achieve the required separation properties in the auxiliary directed graph and finally gave an algorithm which used important separators to solve the 2-ASLASAT problem.

Open Questions and Future Directions. It is well known that the ABOVE GUARANTEE VERTEX COVER problem and the KÖNIG VERTEX DELETION problem (see Appendix for the definition) are equivalent to ALMOST 2-SAT [42]. The FPT algorithm for ALMOST 2-SAT implies FPT algorithms for both these and numerous other problems. The fastest known FPT algorithms for these problems all run in $\mathcal{O}^*(15^k)$ time and it will be interesting to design FPT algorithms with improved running times for all of these problems. Finally, it remains an open question if there exist a polynomial kernel for any of the above mentioned problems. It would be interesting to either design a kernelization algorithm, or prove that these problems are not likely to admit a polynomial kernel.

7 Conclusion

In this thesis, we studied a combinatorial object called *important separators*, and formally described it as a tool with which the parameterized version of a number of graph separation problems with certain properties can be solved. We presented the existing definitions of important separators in undirected graphs, and extended it to directed graphs. Having done this, we gave the FPT algorithms for the MULTIWAY CUT problem, DFVS problem, and the ALMOST 2-SAT problem, with explicit use of the machinery of important separators we set up in Chapter 3.

We note that the notion of important separators was used purely as a branching algorithm in Chapter 4. It was also combined with iterative compression in Chapters 5 and 6. More recently, this notion has been combined with randomization [40] to yield an FPT algorithm for the MULTICUT problem which was open for quite some time. Subsequently, the combination of important separators with randomization has also been used in [36]. Thus, we note that the notion of important separators has already proven to be an extremely useful and versatile tool to handle separation problems in graphs. We conclude this thesis by pointing out two things. The first, is that there are numerous problems on graph separation which are still unresolved and could possibly be resolved by approaches not too dissimilar to the one we have studied in this thesis. Finally, we point out that the kernelization complexity of a lot of problems of this kind is open on general graphs.

Appendix: Problem Definitions

1. ODD CYCLE TRANSVERSAL
   Input: Undirected graph $G = (V, E)$, positive integer k
   Parameter: k
   Question: Does there exist a set $S \subseteq V$ of at size at most k such that the graph $G \setminus S$ is bipartite?

2. EDGE BIPARTIZATION
   Input: Undirected graph $G = (V, E)$, positive integer k
   Parameter: k
   Question: Does there exist a set $S \subseteq E$ of at size at most k such that the graph $G \setminus S$ is bipartite?

3. CHORDAL DELETION
   Input: Undirected graph $G = (V, E)$, positive integer k
   Parameter: k
   Question: Does there exist a set $S \subseteq V$ such that the subgraph of G induced on $V \setminus S$ does not have an induced cycle of length greater than 3?

4. CLUSTER VERTEX DELETION
   Input: Undirected graph $G = (V, E)$, positive integer k
   Parameter: k
   Question: Does there exist a set $S \subseteq V$ such that the subgraph of G induced on $V \setminus S$ is a union of disjoint cliques?

5. MULTICUT
   Input: Undirected graph $G = (V, E)$, a set $R = \{(s_1, t_1), \ldots, (s_\ell, t_\ell)\}$ of ℓ pairs of vertices, a positive integer k
   Parameter: k

Question: Does there exist a set $S \subseteq V$ of size at most k such that there is not path from s_i to t_i in the graph $G \setminus S$, for every $1 \leq i \leq \ell$?

6. ABOVE GUARANTEE VERTEX COVER

Input: Undirected graph $G = (V, E)$, positive integer k

Parameter: k

Question: Does there exist a vertex cover of G of size at most $m + k$ where m is the size of the maximum matching in G?

7. KÖNIG VERTEX DELETION

Input: Undirected graph $G = (V, E)$, positive integer k

Parameter: k

Question: Does there exist a set $S \subseteq V$ of size at most k such that the graph $G \setminus S$ is a König graph?

Bibliography

[1] R. K. AHUJA, T. L. MAGNANTI, AND J. B. ORLIN, *Network flows*, Prentice Hall Inc., 1993. Theory, algorithms, and applications. 1, 29, 34

[2] N. ALON, R. YUSTER, AND U. ZWICK, *Color-coding*, J. ACM, 42 (1995), pp. 844–856. 9

[3] B. ASPVALL, M. F. PLASS, AND R. E. TARJAN, *A linear-time algorithm for testing the truth of certain quantified boolean formulas*, Inf. Process. Lett., 8 (1979), pp. 121–123. 52

[4] H. L. BODLAENDER, *On disjoint cycles*, Int. J. Found. Comput. Sci., 5 (1994), pp. 59–68. 42

[5] ——, *Discovering treewidth*, 2005. 9

[6] ——, *Kernelization: New upper and lower bound techniques*, in IWPEC, 2009, pp. 17–37. 8

[7] N. BOUSQUET, J. DALIGAULT, S. THOMASSÉ, AND A. YEO, *A polynomial kernel for multicut in trees*, in STACS, 2009, pp. 183–194. 8

[8] Y. BOYKOV AND M.-P. JOLLY, *Interactive organ segmentation using graph cuts*, in MICCAI, 2000, pp. 276–286. 34

[9] Y. BOYKOV, O. VEKSLER, AND R. ZABIH, *Fast approximate energy minimization via graph cuts*, IEEE Trans. Pattern Anal. Mach. Intell., 23 (2001), pp. 1222–1239. 34

[10] J. CHEN, F. V. FOMIN, Y. LIU, S. LU, AND Y. VILLANGER, *Improved algorithms for the feedback vertex set problems*, in WADS, 2007, pp. 422–433. 7

[11] J. CHEN, I. A. KANJ, AND W. JIA, *Vertex cover: Further observations and further improvements*, J. Algorithms, 41 (2001), pp. 280–301. 8

[12] J. CHEN, Y. LIU, AND S. LU, *An improved parameterized algorithm for the minimum node multiway cut problem*, Algorithmica, 55 (2009), pp. 1–13. 10, 16, 28, 32, 33, 34, 37, 41, 51

[13] J. CHEN, Y. LIU, S. LU, B. O'SULLIVAN, AND I. RAZGON, *A fixed-parameter algorithm for the directed feedback vertex set problem*, J. ACM, 55 (2008). 7, 10, 42, 47

[14] T. H. CORMEN, C. E. LEISERSON, R. L. RIVEST, AND C. STEIN, *Introduction to algorithms, second edition*, 2001. 2

[15] E. DAHLHAUS, D. S. JOHNSON, C. H. PAPADIMITRIOU, P. D. SEYMOUR, AND M. YANNAKAKIS, *The complexity of multiterminal cuts*, SIAM Journal on Computing, 23 (1994), pp. 864–894. 34

[16] E. DAHLHAUS, D. S. JOHNSON, C. H. PAPADIMITRIOU, P. D. SEYMOUR, AND M. YANNAKAKIS, *The complexity of multiterminal cuts*, SIAM J. Comput., 23 (1994), pp. 864–894. 34

[17] F. DEHNE, M. FELLOWS, M. LANGSTON, F. ROSAMOND, AND K. STEVENS, *An $o(2^{O(k)}n^3)$ fpt algorithm for the undirected feedback vertex set problem*, 2007. 7

[18] M. DOM, J. GUO, F. HÜFFNER, R. NIEDERMEIER, AND A. TRUSS, *Fixed-parameter tractability results for feedback set problems in tournaments*, in CIAC, 2006, pp. 320–331. 42

[19] R. G. DOWNEY AND M. R. FELLOWS, *Fixed parameter tractability and completeness*, in Complexity Theory: Current Research, 1992, pp. 191–225. 42

[20] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized Complexity*, Springer-Verlag, New York, 1999. 7

[21] P. L. ERDÖS AND L. A. SZÉKELY, *On weighted multiway cuts in trees*, Math. Program., 65 (1994), pp. 93–105. 34

[22] F. V. FOMIN, S. GASPERS, D. KRATSCH, M. LIEDLOFF, AND S. SAURABH, *Iterative compression and exact algorithms*, Theor. Comput. Sci., 411 (2010), pp. 1045–1053. 7

[23] G. GARDARIN AND S. SPACCAPIETRA, *Integrity of data bases: A general lock-out algorithm with deadlock avoidance*, in IFIP Working Conference on Modelling in Data Base Management Systems, 1976, pp. 395–412. 42

[24] M. R. GAREY AND D. JOHNSON, *Computers and intractability: A guide to the theory of np-completeness*, in Computer Animation Conference, 1979. 42

[25] N. GARG, V. V. VAZIRANI, AND M. YANNAKAKIS, *Multiway cuts in node weighted graphs*, J. Algorithms, 50 (2004), pp. 49–61. 34

[26] F. GAVRIL, *Testing for equality between maximum matching and minimum node covering*, Inf. Process. Lett., 6 (1977), pp. 199–202. 52

[27] ——, *An efficiently solvable graph partition problem to which many problems are reducible*, Inf. Process. Lett., 45 (1993), pp. 285–290. 52

[28] J. GUO, J. GRAMM, F. HÜFFNER, R. NIEDERMEIER, AND S. WERNICKE, *Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization*, J. Comput. Syst. Sci., 72 (2006), pp. 1386–1396. 7

[29] J. GUO, H. MOSER, AND R. NIEDERMEIER, *Iterative compression for exactly solving np-hard minimization problems*, in Algorithmics of Large and Complex Networks, 2009, pp. 65–80. 7

[30] J. GUO AND R. NIEDERMEIER, *Invitation to data reduction and problem kernelization*, SIGACT News, 38 (2007), pp. 31–45. 8

[31] D. HARTVIGSEN, *The planar multiterminal cut problem*, Discrete Applied Mathematics, 85 (1998), pp. 203–222. 34

[32] F. HÜFFNER, C. KOMUSIEWICZ, H. MOSER, AND R. NIEDERMEIER, *Fixed-parameter algorithms for cluster vertex deletion*, Theory Comput. Syst., 47 (2010), pp. 196–217. 7

[33] F. HÜFFNER, R. NIEDERMEIER, AND S. WERNICKE, *Techniques for practical fixed-parameter algorithms*, Comput. J., 51 (2008), pp. 7–25. 7

[34] M. KARL, *Zur allgemeinen kurventheorie*, Fund. Math., 10 (1927), pp. 96–115. 1

[35] S. KHOT AND V. RAMAN, *Parameterized complexity of finding subgraphs with hereditary properties*, Theor. Comput. Sci., 289 (2002), pp. 997–1008. 52

[36] D. LOKSHTANOV AND D. MARX, *Clustering with local restrictions*, ICALP, (2011, to appear). 66

[37] M. MAHAJAN AND V. RAMAN, *Parameterizing above guaranteed values: Maxsat and maxcut*, J. Algorithms, 31 (1999), pp. 335–354. 52

[38] D. MARX, *Parameterized graph separation problems*, Theoret. Comput. Sci., 351 (2006), pp. 394–406. 10, 11, 13, 28, 32, 34, 37, 41

[39] D. MARX, *Chordal deletion is fixed-parameter tractable*, Algorithmica, 57 (2010), pp. 747–768. 7

[40] D. MARX AND I. RAZGON, *Fixed-parameter tractability of multicut parameterized by the size of the cutset*, STOC, (2011, to appear). 14, 66

[41] K. MEHLHORN, *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*, vol. 2 of Monographs in Theoretical Computer Science. An EATCS Series, Springer, 1984. 7

[42] S. MISHRA, V. RAMAN, S. SAURABH, S. SIKDAR, AND C. R. SUBRAMANIAN, *The complexity of König subgraph problems and above-guarantee vertex cover*, Algorithmica, 58 (2010). 65

[43] R. NIEDERMEIER, *Invitation to Fixed-Parameter Algorithms*, vol. 31 of Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, Oxford, 2006. 2, 6

[44] I. RAZGON AND B. O'SULLIVAN, *Almost 2-sat is fixed-parameter tractable.*, J. Comput. Syst. Sci., 75 (2009), pp. 435–450. 7, 10, 52, 53, 56, 57, 58, 59, 64

[45] B. A. REED, K. SMITH, AND A. VETTA, *Finding odd cycle transversals*, Oper. Res. Lett., 32 (2004), pp. 299–301. 7

[46] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors. ii. algorithmic aspects of tree-width*, J. Algorithms, 7 (1986), pp. 309–322. 9

[47] A. SILBERSCHATZ, P. B. GALVIN, AND G. GAGNE, *Operating system concepts (7. ed.)*, Wiley, 2005. 42

[48] H. S. STONE, *Multiprocessor scheduling with the aid of network flow algorithms*, IEEE Trans. Software Eng., 3 (1977), pp. 85–93. 34

[49] S. THOMASSÉ, *A quadratic kernel for feedback vertex set*, in SODA, 2009, pp. 115–119. 8

[50] G. J. WOEGINGER, *Exact algorithms for np-hard problems: A survey*, Combinatorial Optimization - Eureka, You Shrink!, LNCS, (2003), pp. 185–207. 2