Parameterized Algorithms for Network Design

by

Pranabendu Misra
MATH10201204006

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the
Board of Studies in Mathematical Sciences
In partial fulfillment of requirements

for the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

May, 2016

Homi Bhabha National Institute

Recommendations of the Viva Voce Board

As members of the Viva Voce Board, we certify that we have read the dissertation prepared by Pranabendu Misra entitled "Parameterized Algorithms for Network Design" and recommend that it maybe accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

_____ Date:

Chair - Prof. Venkatesh Raman

_____ Date:

Guide/Convener - Prof. Saket Saurabh

_____ Date:

Member 1 - Prof. R. Ramanujam

_____ Date:

Member 2 - Prof. Sourav Chakraborty

_____ Date:

Member 3 - Prof. Neeldhara Misra

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to HBNI.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it may be accepted as fulfilling the dissertation requirement.

Date:

Place: Guide

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Pranabendu Misra

DECLARATION

I, hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

Pranabendu Misra

List of publications arising from the thesis

Journal Papers.

1. A polynomial kernel for Feedback Arc Set on bipartite tournaments.
   Pranabendu Misra, Venkatesh Raman, M.S. Ramanujan and Saket Saurabh.
   Theory of Computing Systems, November 2013, Volume 53, Issue 4. Pages 609-620.
   A preliminary version appeared in the proceedings of ISAAC 2011, Yokohama, Japan.

Conferences and Other Papers.

1. *Minimum Equivalent Digraph is Fixed Parameter Tractable*
   Manu Basavaraju, Pranabendu Misra, M.S. Ramanujan and Saket Saurabh.
   Manuscript.

2. *Derandomization of Transversal Matroids and Gammoids in Moderately Exponential Time.*
   Pranabendu Misra Fahad Panolan, M.S. Ramanujan and Saket Saurabh.
   Manuscript.

3. *Fast Exact Algorithms for Survivable Network Design with Uniform Requirements.*
   Akanksha Agarwal, Pranabendu Misra, Fahad Panolan and Saket Saurabh.
   Manuscript.

4. *Deterministic Truncation of Linear Matroids.*
   Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan and Saket Saurabh.
   In Proceedings of ICALP 2015, Kyoto, Japan.

5. *Finding Even Subgraphs Even Faster.*
   Prachi Goyal, Pranabendu Misra, Fahad Panolan, Geevarghese Philip and Saket Saurabh.
   In Proceedings of FSTTCS 2015, Bangalore, India.

6. *Parameterized Algorithms to Preserve Connectivity.*
   Manu Basavaraju, Fedor V. Fomin, Petr A. Golovach, Pranabendu Misra, M. S. Ramanujan and Saket Saurabh.
   In Proceedings of ICALP 2014, Copenhagen, Denmark.

Pranabendu Misra

Dedicated to my parents.

ACKNOWLEDGEMENTS

I have been fortunate to have Dr. Saket Saurabh as my thesis adviser. He provided the initial encouragement to pursue a PhD in Computer Science, and invaluable guidance and support all through it. I am deeply indebted to him for all the knowledge and advice I have received over the years, without which this thesis would not have been possible. I am very grateful to Dr. Venkatesh Raman for being my thesis co-adviser. He was always generous with his time, and access to his extensive knowledge and experience.

I am very grateful to Dr. Sourav Chakraborty for his advice and guidance, especially during my masters degree at CMI. I am grateful to Dr. Meena Manajan, Dr. V Arvind, Dr. C. R. Subramaniam, Dr. Sayan Bhattacharjee and Dr. Prahladh Harsha at IMSc, and Dr. Samir Datta, Dr. Madhavan Mukund, Dr. K. V. Subrahmanyam and Dr. S P Suresh at CMI. They were always accessible, and generous with their knowledge and time, and it was a privilege to learn from them.

I am also very grateful to Dr. Madhadan Mukund for his help with arranging my trip to ISAAC 2011, while at CMI. I am thankful to Dr. Fedor Fomin for arranging my visit to the Algorithms Group at the University of Bergen during May-Oct 2015, which was very enriching. I would also like to thank Dr. Geevarghese Philip for arranging my visit to the Max-Planck Institute for Informatics during August 2013, which was very enjoyable.

I would like to thank M.S Ramanujan, Manu Basavaraju, Nitin Saurabh, Fahad Panolan and several other current and past members of the computer science group at IMSc for the numerous fun and interesting discussions on many things, science or otherwise. I am grateful to all my co-authors and collaborators, for I have learnt a lot while working with them and I look forward to our future collaborations. Finally, thanks to my friends at CMI and IMSc, for making my time here a lot of fun.

Contents

Synopsis

In this thesis we design algorithms for several network design problems in the framework of parameterized complexity and exact algorithms. Along the way, we also give deterministic algorithms for certain problems in matroid theory. Our results adds to the small list of results on network design problems in the realm of parameterized and exact algorithms

Owing to the intractability of these problems [GJ02, AB09], research has focused on solving these problems in specific settings, called *computational frameworks*, which involve various trade-offs between the resources consumed by an algorithm and the quality of the computed solution. In this thesis, we study the network design problems under the following frameworks. (1) *Exact algorithms* [FK11], where the goal is to optimally solve a given instance, much faster than a typical brute force algorithm. While the running time of the algorithm is still an exponential in the input size, it becomes tractable for inputs of small and even moderate sizes. (2) Parameterized complexity [CFK+15], where the goal is to solve the "structurally simple" instances of the problem quickly. The simplicity of an instance is measured via an associated parameter, which is often the size of an optimum solution. The input instances have the form (x, k), where x denotes the instance and k is a number which denotes parameter. An FPT algorithm solves such instances in time $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$, where n is the size of x and f is a function of k alone. Kernelization is a important sub-area of parameterized complexity, which studies "data reduction" of problem instances. Given an instance of a parameterized problem, in polynomial time, a *kernelization algorithm* produces an equivalent instance, whose size is bounded by a polynomial function of the parameter. Observe that if a problem admits a polynomial kernel, then it is also FPT, although the converse is not true. (3) *Randomized algorithms* [MR10],

the goal is to use random bits in computing a solution. This often allows us to design faster and simpler algorithms as compared to their deterministic counterparts. Furthermore, for many problems, such Polynomial Identity Testing, the only known efficient algorithms are randomized. *Derandomization* of a randomized algorithm, i.e. finding a deterministic version of a randomized algorithm, is an active topic of research. We derandomize FPT algorithms for several problems in this thesis.

Network Design Problems. Network design problems are one of the most well studied class of problems studied in computer science and graph theory. These problems arise from various real world applications and most of them are NP-hard [GJ02, AB09]. Network design problems have been extensively studied in the framework of approximation algorithms [KN10, GK11, Khu97]. These problems are typically of the following flavor. The input is a graph or digraph, and goal is to find a minimum subgraph the input graph which satisfies a given set of network constraints. Such constraints could on the connectivity between pairs of vertices, the distance between pairs of vertices vertices, the degree of vertices and so on. Observe that, in such problems we must ensure the output graph is "well connected", as per the constraints. In this thesis, we consider network design problems with connectivity constraints. A well known example is the Minimum Spanning Tree problem, where the goal is to compute a minimum cost subgraph which connects all vertices. This problem is solvable in polynomial time. This is an example of a *network optimization problem* where the goal is to find a minimum subgraph which meets the connectivity constraints of the problem.

Another variant of network design problems are the *augmentation problems*. Here, the input is a graph or digraph along with a set of links between vertex pairs. The goal is to augment the input graph with a minimum number of links so that it meets the given connectivity constraints. For example, the Minimum Augmentation problem is to augment an input graph or a digraph with a minimum number of links to make it λ-edge connected. Here the set of links is unrestricted and unweighted, i.e. any pair of edges may be added to the graph at a unit cost, and this problems is polynomial time solvable [Fra92a]. However when the set of links is restricted or has weights, the problem becomes NP-hard. The problems

remains hard even if the input graph is guaranteed to be $\lambda - 1$ edge-connected [Fra92a].

Finally, a third variant is the intersection of network design and graph modification problems. Here the goal is to obtain a subgraph of a input graph or digraph, by deleting or editing a minimum number of the vertices and edges, such that the result lies in a given graph class while also satisfying the given connectivity constraints. These problems are of interest as many NP-hard problems become tractable on specific graph classes. Hence we can extend the algorithms for the graph class, to those instances that are "close" to this class. Such problems can also be viewed as a graph modification problem with connectivity constraints.

While these problems have been extensively studied in the framework of approximation algorithms, their parameterized complexity is only recently being explored. Furthermore, no exact algorithms better than the trivial brute-force algorithm is known for many of these problems. The following is a list of network design problems studied in this thesis.

(1) Eulerian Edge Deletion. Eulerian graphs are those graphs which are connected and every vertex has even degree. Many network design problems, are tractable on eulerian graphs [CCM92, PTW01, ZA12, SHSL02]. Hence we consider the problems of remove a minimum number of edges from an input graph or digraph so that the remaining graph is eulerian, which is called the Eulerian Edge Deletion. We also consider another variant of this problem, called Connected Odd Edge Deletion, where the remaining graph must be an odd graphs, i.e. the degree of every vertex is an odd number. For both these problems we give FPT algorithms running in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$, where k is the size of a solution. This improves upon the previous algorithms for Eulerian Edge Deletion whose running time depended on k as $2^{O(k \log k)}$ [CMP+14].

(2) *Exact algorithms for* Survivable Network Design *with uniform requirements.* Survivable network design involves designing a minimum cost network, or augmenting a given network with a set of links of minimum total cost, so that it remains connected after one or more link failures [KN10]. For example, a minimum spanning tree network will becomes disconnected on even a single connection failure. Indeed, any network that must survive $\lambda - 1$ failures must be λ connected. There are two versions of this problem. In

Weighted Network Optimization, we are given a graph or digraph, on n vertices and m edges with edge costs, which is λ connected, The goal is to find a minimum cost subgraph of this graph which is λ-connected. In Weighted Minimum Augmentation, the goal is to augment a given graph or digraph to a λ connected graph by adding a set of new edges of minimum total cost. Note both problems are NP-hard [KN10]. The best known exact algorithm for this problem, until now, was the trivial brute-force algorithm which enumerates and tests all possible solution, and hence takes $2^{\mathcal{O}(m)}$ time. In this thesis, we design fast exact algorithms, which run in time $2^{\mathcal{O}(n)}$, for both the problems.

(3) *Parameterized complexity of* Augmentation by One. Next we consider the parameterized complexity of network augmentation of a $\lambda - 1$ connected graph to λ. The input is a $\lambda - 1$ connected graph G and a set of links L. The goal is to augment G with a minimum number of links in L to make it λ connected. This is a NP-hard problem via a reduction from the classical Hamiltonian Cycle problem. We consider this problem with respect to two parameters. First we parameterize it by the size of a minimum augmenting set and show that it admits a $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ FPT algorithm, even when the links have weights. This improves upon the previous algorithm which runs in time $2^{\mathcal{O}(k \log k)}n^{\mathcal{O}(1)}$ [MV15]. Next we parameterize it by the size of unused links, i.e. the complement of a minimum augmenting set. We show that the problems admits a $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ FPT algorithm and a polynomial kernel with this parameter. Our algorithms are based on an interesting connection of these problems to the Steiner Tree problem.

(4) Minimum Equivalent Digraph *is fixed parameter tractable.* In this problem, we are given a digraph D as input and the goal is to find a minimum spanning subgraph H which has the same reachability relations as D. This problem has been studied in the frameworks of approximation and exact algorithms [BDK09, Vet01, FLS14]. Observe that a reasonable parameter for this problem is the number of arcs which may be safely deleted from the graph, i.e. $A(D) \setminus A(H)$, since any optimum solution has $\mathcal{O}(|V(D)|)$ arcs. We show that with this parameter, the problem admits a polynomial kernel, and hence it is FPT. Our algorithm is based on an interesting combinatorial result, which relates the size of a maximum arc deletion set to the total number of arcs which may be individually removed from the graph.

Matroid Algorithms. Matroids have found application in many recent FPT and kernelization algorithms [FLS14, FLPS14, KW12, Mar09]. Matroids and algorithmic tools based on them play an important role in several of the results mentioned above. The connections between linear matroids and graph connectivity are well studied. Co-graphic matroid represents those subsets of edges of a connected graph, which may be removed from the graph without making the graph disconnected [Oxl06]. This property is crucially used in our algorithms for Eulerian Deletion, mentioned above. A key step in this algorithm is to first find a truncation of the co-graphic matroid in *deterministic* polynomial time.

(1) *Deterministic truncation of linear matroids.* We give a deterministic algorithm for truncation of any linear matroid. Our algorithms are based on a connection of matroid truncation to the *Wronskian* matrix of polynomials which is well studied in linear algebra. Representative Sets were a key tool which was used in many recent FPT algorithms and kernels. The previously known algorithms for computing a representative set was randomized, whenever the rank of the matroid is too large [FLS14, KW12]. Using the above algorithm for matroid truncation, we give deterministic algorithms for computing representative sets over any matroid of any rank. This also de-randomizes several algorithms of Marx[Mar09], including FPT algorithms for ℓ-Matroid Parity and ℓ-Matroid Intersection.

(2) *Deterministic representation of gammoids in moderately exponential time.* Gammoids are another class of linear matroids which arise from graph connectivity. They represent the subsets of vertices of a graph which are reachable from a particular subset of vertices, called the *terminals*, by a system of disjoint paths [Oxl06]. Gammoids have recently been used in designing FPT algorithms and kernels for several problems [FLS14, KW12]. Gammoids are closely related to the class of transversal matroids, and one can find a representation of them in randomized polynomial time [Mar09]. Here, we give moderately exponential deterministic algorithms for computing a representation of transversal matroid and gammoids. Our algorithms run in polynomial time, whenever the rank of the matroid is a fixed constant. This gives the first improvement upon the trivial brute force deterministic algorithm for finding a representation.

List of Figures

Introduction

Chapter 1

Organization of the Thesis

This thesis is organized into three parts – (*i*) an introduction to the thesis, computational frameworks and algorithmic techniques, (*ii*) deterministic algorithms for matroids and application, and (*iii*) algorithms for various graph connectivity problems.

In the first part, we introduce computational frameworks and algorithmic techniques used in this thesis. In chapter 2 we introduce the computational frameworks in which the results of this thesis were obtained. In chapter 3, we review the notations and basic definitions of graphs, connectivity and sets which are required in this thesis. In chapter 4, we give a polynomial kernel for Feedback Arc Set on bipartite tournaments. This is an illustration of a kernelization algorithm, and we will design such algorithms for several network design problems in later chapters. In chapter 5, we introduce matroids and representative sets, and give various definitions, examples and algorithmic tools which will be used in the thesis. Then in chapter 6, we give a simple FPT algorithm for the k-path problem. This illustrates the technique of dynamic programming with representative sets over linear matroids, which is a key technique used in this thesis. In chapter 7, we list a few matroids which are related to graph connectivity and their analogues in matroids. Finally in chapter 8, we give a brief history of of network design problems, the currently known algorithmic results, and a list open problems, from the viewpoint of parameterized complexity.

In the second part of the thesis, we give deterministic algorithms for two problems in matroid theory. These algorithms then de-randomize many other algorithms, and will also

be useful in other results in the later parts of the thesis. In chapter 9, we give a deterministic polynomial time algorithm to find the representation of the truncation of a linear matroid. Then we give a deterministic algorithm to compute representative sets over any linear matroid, and deterministic FPT algorithms for ℓ-Matroid Parity. In chapter 10, we give a deterministic algorithm for constructing a representation of transversal matroid and gammoids in moderately exponential time.

In the third part of the thesis, we give algorithms for various network design problems. In chapter 11, we give a single exponential FPT algorithm for the Eulerian Deletion problem in graphs and digraphs, parameterized by the size of the deletion edge set. In chapter 12, we give single exponential exact algorithms for the problems of finding a minimum cost λ-edge connected subgraph of an input graph or digraph. Our algorithms also imply a single exponential algorithm for computing a minimum cost augmenting set to augment the edge connectivity of a graph or digraph to λ. In chapter 13, we give a single exponential FPT algorithm for the problem of augmenting the connectivity of a graph from λ to $\lambda + 1$ by a given set of links parameterized by the size of the augmenting set. In chapter 14, we consider the above problem parameterized by the size of the complement of the augmenting set in the given set of links. We show that, with respect to this parameter the problem has a single exponential FPT algorithm and a polynomial kernel. Our algorithms are based on a connection of these problems to the Steiner Tree problem. And finally, in chapter 15 we give a kernelization algorithm for the Minimum Equivalent Digraph problem, parameterized by the size of the number of arcs which may be removed from the digraph. We also show that Minimum Strongly Connected Spanning Subgraph has a linear time FPT algorithm.

Chapter 2

Computational Framework

Computer Science grew out of a need to compute solutions to real world problems as efficiently as possible. A computational problem is often to compute a maximum (or a minimum) value solution of the given problem instance, which is called an optimal solution to the instance. In computer science, such problems are often modeled as abstract problems on mathematical structures such as graphs, set systems, linear equations etc, which captures all the relevant details of the problem. The primary goal is to design efficient algorithms to find a solution to these abstract problems. With the advent of computers, computer science grew in popularity, and many problems arising from the real world were investigated. It was soon discovered that only a few of these problems seem to be efficiently solvable and for some inexplicable reason, a vast majority are not. This mystery took on a more concrete form, when Karp and Levin, independently, discovered that these "difficult" problems were all related to each other, as in, if one of them could be solved efficiently then all of them could be solved efficiently. It led to the widely held belief that these difficult problems are indeed intractable, and this was the starting point of research in Computational Complexity which investigates the difficulty of computational problems. A proof of the intractability of these problems is the biggest open problem in computer science and perhaps all of mathematics.

But this still leaves us with the question of solving these problems, as many of these problems have important real world applications. Researchers have attempted to tackle the

intractability of these problems, within acceptable limits, by designing algorithms for them in various computational frameworks. When we are mainly concerned with how much time it takes to solve a problem, the following are the prominent algorithmic frameworks.

- Exact algorithms, where we try to design algorithms to find an optimal solution as quickly as possible.
- Approximation algorithms, where we try to efficiently compute a solution which is not too worse compared to an optimal solution.
- Parameterized Complexity, which tries to exploit structural properties of the problem, to find an optimum solution efficiently. In particular, the goal is to efficiently solve those instances of the problem which are "structurally simple", as is the case with real world instances of many problems.

In this thesis, we design algorithms mainly in the framework of parameterized complexity and exact algorithms. In the following, we briefly review some relevant notions and terms. For a detailed introduction to the topic of classical complexity theory we refer to [AB09] for parameterized complexity we refer to [CFK+15] and finally for exact algorithm we refer to [FK11].

Often a computation problem is recast as a decision problem, where the objective is to determine if the given problem instance has a solution whose value is greater (or smaller) than a given target value. Generally, the computational problem and the decision problem are "equivalent", i.e. an efficient algorithm that answers the decision problem can be converted into an efficient algorithm for the computational problem, and vice versa. Let us state these formally. Let Σ be finite set, called the *alphabet* and let Σ^* be the set of strings over this alphabet. A *language* L is a subset of Σ^* and the *decision problem* Π associated with L is the following. Given a string $x \in \Sigma^*$, decide if $x \in L$ or not. In many contexts, the terms Π and L can be used interchangeably. An algorithm A for the problem Π is a finite sequence of instruction, that given x can correctly decide the membership of x. Often, algorithms are modeled as *Turing Machines* which carries out the steps of the algorithm. The complexity of an algorithm is measured in terms of the resources it consumes, such

as the number of steps required (i.e. time), or amount of space used. In this thesis, we are mainly concerned with time complexity of algorithms, which is also called the running time. Before moving ahead, let us briefly discuss how such things are measured. For a resource, such as time, we measure the running time as a function of n, where n is the length of the input string x. For example, say that an algorithm is polynomial time if the running time of the algorithm is a polynomial function of of n. Usually we don't state the function explicitly and instead use the "Big-O" notation. For two functions, $f(n)$ and $g(n)$ we write $f(n) = \mathcal{O}(g(n))$ if there are constants C and N such that $f(n) \leq Cg(n)$ for all $n > N$. For example, when the running time of an algorithm is a function which is a polynomial in n of degree c, we write the running time as $\mathcal{O}(n^c)$, and such algorithms are called polynomial time algorithms. For an exponential function $f(n)$, we also use the notation $\mathcal{O}^*(f(n))$ to mean $\mathcal{O}(f(n) \cdot n^c)$ where c is a constant.

Problems are often sorted into classes, as per the properties of the best known algorithms for the problem. Such classes of problems is called a *complexity class*. Two important complexity classes are P and NP. The class P contains all those problems for which there are polynomial time algorithms. To define NP we need notion of a polynomial time *verifier*. A problem Π is said to have a verifier A, which is an algorithm that given a string x of length n, and a claimed proof y of membership of x in Π such that $|y|$ is some polynomial in n, decides the validity of the proof in polynomial time. In other words $x \in \Pi$, if and only if there is a string y of length polynomial in n, such that the verifier A accepts the string (x, y) in polynomial time. The class NP contains all those languages that have a polynomial time verifier. It is clear that P is contained in NP. Problems in P are considered to be "easy" as they can be decided in polynomial time. The class NP includes the "difficult" problems, for which there are no known polynomial time algorithms, but a solution to an NP-problem can be verified in polynomial time.

A *polynomial time many one reduction* from a problem A to a problem B is a polynomial time algorithm, that given an instance x of A, produces an instance y such at $|y|$ is a polynomial function of $|x|$, and $x \in A$ if and only if $y \in B$. Therefore, if there is a polynomial time algorithm for solving B, then there is also a polynomial time algorithm for solving A. A problem B is called NP-hard if there is a polynomial time many one

reduction from every problem in NP to B. Reduction between problems in NP was first investigated by Karp, and independently by Levin, who showed that a number of problems in NP, which have no known polynomial time algorithm, are all reducible to each other. This led to the definition of *NP-complete* problems which are those problems in NP which are also NP-hard. Since then thousands of problems arising from real world applications have been shown to be NP complete, and it led credence to the belief that these problems are indeed intractable. See [AB09, GJ02] for more details.

Let us now look at the computational frameworks used in this thesis.

2.1 Exact Algorithms

The goal of an exact algorithm is to compute an optimum solution to an instance of a computational problem in as little time as possible. For problems whose decision versions are NP-complete, such an algorithm is expected to take an exponential time. Observe that we always have a "brute-force" algorithm for this class of problems, which enumerates all potential solutions, verifies each one and finally returns an optimal solution. Hence, the goal is to design an exact algorithm which is provably faster than the brute-force algorithm. While the running time of the algorithm is still an exponential in the input size, it becomes tractable for inputs of small and even moderate sizes. See [FK11] for an introduction to this framework. In this thesis, we will design fast exact algorithms for several network design problems.

2.2 Parameterized Complexity

In parameterized complexity we investigate problems with respect to their structural parameters in order to design algorithms which solve them almost efficiently. Let us define the terms formally. A *parameterized language* L is a subset of $\Sigma^* \times N$. A parameterized problem is to decide the membership of a pair (x, k) where x is a string and k is a number in a parameterized language. Here x is called an instance, while k is called the parameter of the instance. A parameterized problem is *fixed parameter tractable* if there is an algorithm

for this problem which decides the membership of (x, k) pairs in time $\mathcal{O}(f(k)n^c)$ where is f is an arbitrary function of k alone and c is a constant. FPT algorithms are considered to be "efficiently solvable" in this framework. The key insight here is that when the parameter is small, the problem instances are"structurally simple" even if $|x|$ is large. Hence, the instance can be solved efficiently. The most commonly used parameter is the solution size. Other examples of parameters are the treewidth of the input graph, rank of an input matrix, the chromatic number of the input graph etc. Example of problems which admit FPT algorithms include k-PATH, VERTEX COVER, FEEDBACK VERTEX SET, MULTI-CUT etc. When stating the running time of an FPT algorithm, we often omit the polynomial terms in the running time and write $\mathcal{O}^*(f(k))$, where the algorithm has a running time of $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$. When the FPT algorithm has a running time $\mathcal{O}(f(k)n)$, it is called a *linear time* FPT algorithm.

Not all parameterized problems are known to be fixed parameter tractable (FPT). Examples include problems such as INDEPENDENT SET, DOMINATING SET, SET COVER etc. There is a theory of intractability of parameterized problems which sorts the problems into a hierarchy of complexity classes called the W-hierarchy. It is organized as $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \ldots \subseteq \mathsf{XP}$, where each $W[i]$ is a superset of all $W[j]$ for $j < i$, and it is conjectured that they are all distinct. The class XP consists of those parameterized problems which have an algorithm, that given $(x, k) \in \Sigma^* \times \mathbb{N}$ and decides the membership in time $\mathcal{O}(n^{f(k)})$ where f is an arbitrary function of k alone. Note that such an algorithm is often feasible for small values of k. We refer to [CFK+15, FG06] for an introduction to parameterized complexity.

2.2.1 Kernelization

Kernelization is an important sub-topic of parameterized complexity. A parameterized problem Π, is said to admit an $f(k)$ kernel,where f is a function of k, if there is a polynomial time algorithm which given an input (x, k) outputs (x', k') such that $|x'|, |k'| \leq f(k)$ and $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$. The function $f(k)$ is often called the size of the kernel. It is easy to observe that if a parameterized problem has an FPT algorithm with a running time of $\mathcal{O}(f(k)n^c)$, then it also admits a $\mathcal{O}(f(k))$ kernel.

A more interesting notion is that of a *polynomial kernel*, where the goal is to find a kernel of size polynomial in the parameter. It is known that the parameterized version of many NP-complete problems admit polynomial kernels. This include problems such as Vertex Cover, Feedback Vertex Set etc. On the other hand, there are several problems which have an FPT algorithm but are not known admit polynomial kernels. This includes problems such as Multicut and k-path. There is also a corresponding theory of hardness of kernelization which is being actively developed, based on deep connections to classical complexity theory. Note that if a problem admits a polynomial kernel, it is also FPT, as even a bruteforce algorithm to solve the kernel will run in FPT time. In this thesis, we shall shall show that some important network design problems admit polynomial kernels.

2.3 Randomized Algorithms

As the name suggests, randomized algorithms, and the complexity classes related to them, are obtained by allowing the algorithms to use random bits in their computation. This often allows us to design faster and simpler algorithms as compared to their deterministic counterparts. While we won't go into the details, let us briefly mention a few things, which are relevant to this thesis. See [MR10] for an introduction to this topic.

The class BPP contains all those problems which are decidable by a randomized polynomial time algorithm whose probability of error bounded by a constant. Note that it is a counterparts of the class P. It is conjectured that $\mathsf{BPP} = \mathsf{P}$, even though we don't know of a proof of $\mathsf{BPP} \subset \mathsf{NP}$. A proof of these conjectures is a major open problem in theoretical computer science. There are randomized counterparts of many other complexity classes as well, such as FPT and XP, and it is conjectured that they are equivalent to their deterministic counterpart. Observe that any problems in P is contained in BPP, but there are several problems such as Polynomial Identity Testing which are in BPP, i.e. they have randomized polynomial time algorithms, but no deterministic polynomial time algorithms have been found yet. Similarly there are several problems for which only randomized FPT and kernelization algorithms are known. The *derandomization* of these randomized algorithms, i.e. constructing deterministic counterparts of randomized

algorithms for specific problems, is an active topic of research in computer science. In this thesis, we shall see two such examples for problems on linear matroids. These algorithms are then used to obtain deterministic algorithms for some other problems.

Chapter 3

Preliminaries

We review some basic definitions and notations used in this thesis. For more details see, [Die12, BJG08].

3.1 Sets

Let $\mathbb{N}^*$ denote the set of natural numbers along with 0, and let $\mathbb{R}^*$ denote the set of non-negative real numbers. We define $[n]$ to be the set $\{1, \ldots, n\}$. We use the term *universe* to distinguish a set from it's subsets. For any two subsets X and Y of a universe U, we use $X \setminus Y$ or $X - Y$ to denote the subset of X whose elements are not present in Y. For any set U define $\binom{U}{i} = \{X \mid X \subseteq U,\ |X| = i\}$. We say that a family $\mathcal{S} = \{S_1, \ldots, S_t\}$ of subsets of a universe U is a *p-family* if each set in $\mathcal{S}$ has cardinality at most p. For two families $\mathcal{S}_1$ and $\mathcal{S}_2$ of a universe U, define $\mathcal{S}_1 \bullet \mathcal{S}_2 = \{S_i \cup S_j \mid S_i \in \mathcal{S}_1,\ S_j \in \mathcal{S}_2 \text{ and } S_i \cap S_j = \emptyset\}$. Throughout the thesis we use ω to denote the exponent in the running time of matrix multiplication, the current best known bound for which is $\omega < 2.373$ [Wil12].

Let $\mathcal{S}$ be a collection of subsets of an universe U, and let $\mathcal{P}$ be a subset $\mathcal{S}$. We say that $\widehat{\mathcal{P}} \subseteq \mathcal{P}$ is a *representative set* of $\mathcal{P}$, if for any $X \in \mathcal{P}$ and $Y \subseteq U$ such that $X \cup Y \in \mathcal{S}$, there is some $\widehat{X} \in \widehat{\mathcal{P}}$ such that $\widehat{X} \cup Y \in \mathcal{S}$. As we shall see later, this notion is especially useful when $\mathcal{S}$ represents a linear matroid.

3.2 Graphs

A *graph* G is a pair (V, E) where V is a set of elements called *vertices*, and E is a collection of pairs of vertices called the *edges*. The two vertices of an edge are called the *endpoints* of the edge and, the edge is said to be *incident* on the two vertices. A *digraph* or a directed graph D is a pair (V, E) where V is the vertex set and E is the set of *directed edges* or *arcs* of the digraph. A directed edge $e = (u, v)$ is an ordered pair of vertices where the vertex u is called the *initial vertex* or *tail* of e (denoted by $\mathsf{tail}(e)$), and the vertex v is called the *terminal vertex* or *head* of e (denoted by $\mathsf{head}(e)$). Note that for $u, v \in V(D)$, (u, v) and (v, u) denote two distinct directed edges, and they are *reverse edges* of each other. A *simple graph* is a graph where every edge is distinct and further the two endpoints of any edge are distinct vertices. A *multigraph* is a graph where we allow multiple copies of the same edge in the edge set (*parallel edges*) and further, we allow an edge to have the same vertex as both it's endpoints (*loops*). A *subdivision* of an edge $e = (u, v)$ of G yields a new digraph, G', containing one new vertex w, and with an edge set replacing e by two new edges, (u, w) and (w, v). That is, $V(G') = V(G) \cup \{w\}$ and $E(G') = (E(G) \setminus \{(u, v)\}) \cup \{(u, w), (w, v)\}$. Unless specified otherwise, the graphs and digraphs in this thesis are simple.

In the following, we define various terms with respect to undirected graphs. Many of these terms are similarly defined for digraphs and we only mention those that are defined differently. For a graph G we use $V(G)$ and $E(G)$ to denote the vertex set and the edge sets respectively. For $X \subseteq V(G)$, $G[X]$ denotes the *induced subgraph* on X of G, which has vertex set X, and the edge contains all edges in G whose both endpoints lie in X. Given a graph or digraph G and a subset X of $V(G)$ (or $E(G)$), by $G \setminus X$ we denote the graph obtained by deleting X from $V(G)$ (or from $E(G)$). When X contains just a single vertex v (or a single edge e), we often write $G - v$ (and $G - e$) to denote $G \setminus \{v\}$ (and $G \setminus \{e\}$). In a undirected graph, the *neighbourhood* of a vertex $v \in V(G)$ is defined as the set $N(v) = \{u \in V(G) | (u, v) \in E(G)\}$. In a directed graph, we define $N^+(v) = \{w \in V(D) | (v, w) \in E(D)\}$ and $N^-(v) = \{u \in V(D) | (u, v) \in E(D)\}$ to be the set of *out-neighbours* and *in-neighbours* of v respectively. The *underlying graph* of a digraph D is the undirected graph obtained from D by removing the direction of every

edge. The *union of graphs* G_1 and G_2 is the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$ A graph G_1 is said to be a subgraph of another graph G_2 is $V(G_1) \subseteq V(G_2)$ and $E(G_1) \subseteq E(G_2)$. A *walk* W in a graph G consists of a sequence of vertices and edges $\{v_0, e_1, v_1, e_2, \ldots, e_\ell, v_\ell\} \subseteq E(G)$, such that the edge e_i is incident on v_{i-1} and v_i. We say that a walk W visits the vertices and the edges in W. The vertices v_0 and v_ℓ are called the start and the end vertex, respectively, of the walk. All other vertices visited by W are called internal vertices. A walk is a closed walk if it's start vertex and end vertex are the same. A path P is a walk which visits any vertex at most once, i.e. there are at most two arcs in P which are incident on any vertex visited by W. A cycle is a closed walk which visits all the internal vertices exactly once and the start/end vertex exactly twice. Observe that any edge or vertex occurs at most once in a path P which induces an ordering of these edges, and we say that P visits these edges in that order. Similarly, for the collection of vertices which are present in P, P induces ordering of these vertices and we say that P visits them in that order. Let P be a path which visits a vertex u and then visits a vertex v. We write $P[u, v]$ to denote the sub-path of P which starts from u and ends at v. For two path P and Q such that the end vertex of P is same as the start vertex of Q, we write $P + Q$ to denote the walk from the start vertex of P to the end vertex of Q. A *path system* $\mathcal{P}$ in graph G is an ordered collection of paths in G. It is *edge-disjoint* if no two paths in the system share an edge. We use $V(\mathcal{P})$ and $E(\mathcal{P})$ for the set of vertices and edges, respectively, in a path system $\mathcal{P}$. Similarly, we may define a *vertex disjoint* path systems. A *cycle* is a graph (or digraph) with vertex set $V(P) = \{v_1, v_2, \ldots, v_\ell\}$ and edge set $E(P) = \{(v_i, v_{i+1}) \mid 1 \leq i \leq \ell - 1\} \cup \{(v_\ell, v_1)\}$, i.e. it is a path plus an edge from the last to the first vertex. A *acyclic graph* (or a digraph), as the name implies, contains no cycles. A connected acyclic graph is called a *tree*. An undirected acyclic graph is a union of trees, and it is called a *forest*. An acyclic digraph is are called DAG, which is short for "directed acyclic graph". A *clique* or a complete graph is a simple graph where every pair of vertices form an edge. A clique on n vertices is denoted by K_n. An independent set is a graph with an empty edge set.

3.2.1 Graph Connectivity

A graph is called *connected* is there if a path between every pair of vertices. Similarly, a digraph is called *strongly connected* if for every ordered pair of vertices, (u, v), there is a path from u to v. Observe that every vertex of a strongly connected digraph must be part of some cycle. A digraph is called *weakly connected* is the underlying graph of the digraph is connected. Let $P_1 = x_1x_2 \dots x_r$ and $P_2 = y_1y_2 \dots y_s$ be two *edge-disjoint* paths in graph G. If $x_r = y_1$ and $V(P_1) \cap V(P_2) = \{x_r\}$, then we use P_1P_2 to denote the path $x_1x_2 \dots x_ry_2 \dots y_s$. A connected component of a graph G is a maximal subgraph of G which is connected. From the definition of a connected graph it follows that the connected components form a partition of a graph. Similarly, in digraphs we have *strongly* connected components, and these form a partition of the digraph.

Let λ be a natural number. A graph G is called λ *edge connected* if for any pair of vertices u and v, there is a collection of λ edge disjoint paths with u and v as their endpoints. Similarly, a graph G is called λ *vertex connected* if for any pair of vertices u and v, there is a collection of λ paths with u and v as their endpoints, which are vertex disjoint except for the endpoints u and v. We have similar definitions for digraphs. A *edge cut* in a graph G is a partition $(X, \overline{X})$ of $V(G)$ and, $\delta_G(X)$ denotes edges of G with one endpoint in X and the other in $\overline{X}$. The *size of a cut* $(X, \overline{X})$ in G is the value $|\delta_G(X)$. When the graph is clear from context, we simply write $\delta(X)$. Note that in digraphs, $(X, \overline{X})$ and $(\overline{X}, X)$ denote different cuts, and $\delta(X)$ denotes those directed edges with their tail in X and head in $\overline{X}$. Observe that if $(X, \overline{X})$ is an edge cut in G, then there is no path from a vertex in X and a vertex in $\overline{X}$ in $G \setminus \delta(X)$. Often we use $\delta(X)$ to denote both the edge-set as well as the number of edges in this set. A *vertex cut* in a graph G is a pair (X, Y) of subsets of $V(G)$ such that $X \cup Y = V(G)$ and any path between a vertex in X to a vertex in Y (and vice-versa) passes through a vertex in $X \cap Y$. This definition implies that, there is no path between a vertex in X and a vertex in Y (and vice-versa) in $G \setminus (X \cap Y)$. The size of the cut (X, Y) is defined as $|X \cap Y|$. As in the case of edge cut, we often use $X \cap Y$ to denote the vertex cut (X, Y). In this thesis, we are largely concerned with the edge connectivity of graphs, and we use the terms "cut" and "λ connected graph" to mean an edge cut and a λ edge connected graph. For subsets of vertices A and B, we say that a cut $(X, \overline{X})$ separates

A and B f $A \subseteq X$ and $B \subseteq Y$. We have a similar definition for the case of vertex cuts. Menger's theorem is one of the earliest results in graph connectivity. It relates the size of a min-cut between two vertices to the number of edge disjoint paths between them.

Theorem 3.1 (Menger [Die12])**.** *Let G be a graph and let u, v be two distinct vertices. Then the size of minimum cut in G separating u and v is equal to the maximum number of edge-disjoint paths between u and v in G.*

Chapter 4

Illustration: A polynomial kernel for Feedback Arc Set on Bipartite Tournaments

In this chapter, we consider an example of a kernelization algorithm. A *feedback arc set* in a digraph is a subset of arcs whose deletion makes the digraph acyclic, and the Feedback Arc Set problem is to determine if an input digraph has a feedback arc set of a given size. We consider the Feedback Arc Set problem on bipartite tournaments, and show that it admits a polynomial kernel when parameterized by the solution size. Feedback Arc Set on tournaments is useful in *rank aggregation*. In *rank aggregation* we are given several rankings of a set of objects, and we wish to produce a single ranking that on average is as consistent as possible with the given ones, according to some chosen measure of consistency. This problem has been studied in the context of voting [Bor81, Con85], machine learning [CSS97], and search engine ranking [DKNS01]. A natural consistency measure for rank aggregation is the number of pairs that occur in a different order in the two rankings. This leads to *Kemeny rank aggregation* [Kem59, KS62], a special case of a weighted version of Feedback Arc Set on tournaments (FAST). Similarly, Feedback arc set on bipartite tournaments finds its usefulness in applications that require establishment of mappings between"ontologies". We refer to [SKH10] for more details.

Feedback Arc Set on Bipartite Tournaments (FASBT)	Parameter: k
Input: A bipartite tournament $H = (X \cup Y, E)$ and an integer k.	
Question: Does there exist a subset $F \subseteq E$ of at most k arcs whose removal makes H acyclic?	

In the last few years several algorithmic results have appeared on "feedback set" problems in tournaments and bipartite tournaments. Speckenmeyer [Spe89] showed that Feedback Vertex Set is NP-complete on tournaments. FAST was conjectured to be NP-complete for a long time [BJT92], and it was proved only a few years ago. Ailon et al. [ACN05] gave a randomized reduction from Feedback Arc Set on general directed graphs, which was independently derandomized by Alon [Alo06] and Charbit et al. [CTY07]. From the approximation perspective, initially a constant factor approximation was obtained for FAST in [vZHJW07] and later it was shown in [KMS07] that it admits a polynomial time approximation scheme. Now we turn to problems on bipartite tournaments. Cai et al. [CDZ02] showed that Feedback Vertex Set on bipartite tournaments is NP-complete. They have also established a min-max theorem for feedback vertex set on bipartite tournaments. However, only recently Guo et al. [GHM07] showed that FASBT is NP-complete. FASBT is also known to admit constant factor approximation algorithms [Gup08, vZ11]. These problems are also well studied in parameterized complexity. Raman and Saurabh [RS06] showed that FAST is FPT by obtaining an algorithm running in time $\mathcal{O}(2.415^k \cdot k^{4.752} + n^{\mathcal{O}(1)})$. Recently, Alon et al. [ALS09] have improved this result by giving an algorithm for FAST running in time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log^2 k)} + n^{\mathcal{O}(1)})$. Moreover, a new algorithm due to Karpinsky and Schudy [KS10] with running time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k})} + n^{\mathcal{O}(1)})$ improves again the complexity of FAST. Dom et al. [DGH+10] obtained an algorithm with running time $\mathcal{O}(3.373^k n^6)$ for FASBT based on a new forbidden subgraph characterization. It is well known that Feedback Vertex Set admits a kernel with $\mathcal{O}(k^2)$ and $\mathcal{O}(k^3)$ vertices on tournaments and bipartite tournaments respectively [AK10, ALS09, DGH+10]. And only recently a $\mathcal{O}(k)$ vertex kernel for Feedback Arc Set on tournaments was obtained [BFG+09] using maximal transitive modules.

In this chapter, we show that that FASBT admits a *cubic vertex* kernel, i.e we give

a polynomial time algorithm which given an instance (H, k) of FASBT produces an equivalent instance (H', k') on $\mathcal{O}(k^3)$ vertices. This is the first polynomial kernel for this problem and it completes the kernelization picture for the FEEDBACK ARC/VERTEX SET problem on tournaments and bipartite tournaments, as polynomial kernels were known for all the other problems. The main ingredient of our result is a data reduction rule that applies independent modules in a non trivial way. Our result adds to a small list of problems for which graph modules have turned out to be useful. Previously, clique modules and transitive modules were useful in obtaining kernels for CLUSTER EDITING and FAST [BFG+09, Guo09].

4.1 Preliminaries

Let us review a few definitions and results that relevant to this chapter. A bipartite tournament $H = (X \cup Y, E)$ is an orientation of a complete bipartite graph, i.e. its vertex set is the union of two independent disjoint sets X and Y and for every pair of vertices $u \in X$ and $v \in Y$ there is exactly one arc between them. By C_4 we mean a directed cycle of length 4. Given a directed graph $D = (V, E)$, a subset $F \subseteq E$ of arcs is called a *feedback arc set* (fas) if $D \setminus F$ is a directed acyclic graph. A feedback arc set F is called *minimal fas* if none of the proper subsets of F is a fas. Given a directed graph $D = (V, E)$ and a set F of arcs in E define $D\{F\}$ to be the directed graph obtained from D by reversing all arcs of F. In our arguments we will need the following folklore characterization of minimal feedback arc sets in directed graphs. (Follows from Proposition 15.0.1(b) of [BJG08].)

Proposition 4.1. *Let $D = (V, A)$ be a directed graph and let F be a minimal feedback arc set of D. Then, $D\{F\}$ is a directed acyclic graph.*

4.1.1 Modular Partitions

A *Module* of a directed graph $D = (V, E)$ is a set $S \subseteq V$ such that $\forall u, v \in S$, $N^+(u) \setminus S = N^+(v) \setminus S$, and $N^-(u) \setminus S = N^-(v) \setminus S$. Essentially, every vertex in S has the same set of in-neighbours and out-neighbors outside S. The empty set and the whole of the vertex set

are called *trivial modules*. We always mean non-trivial modules unless otherwise stated. Every vertex forms a singleton module. A *maximal module* is a module such that we cannot extend it by adding any vertex. The *modular partition*, $\mathcal{P}$, of a directed graph D, is a partition of the vertex set V into $(V_1, V_2, \ldots, V_\ell)$, such that every V_i is a maximal module. If A and B are two modules in a directed graph, then all the edges between them are directed either from A to B, or from B to A. We denote the two cases by $A \to B$ and by $B \to A$ respectively. Now we look at some simple properties of the modules of a bipartite tournament.

Lemma 4.2. *Let $H = (X \cup Y, E)$ be a bipartite tournament, and S be any non-trivial module of H. Then S is an independent set. Thus $S \subset X$ or $S \subset Y$.*

Proof. If $|S| = 1$ then it is obvious. So we assume that $|S| \geq 2$. But S cannot contain two vertices x and y such that $x \in X$ and $y \in Y$ because their neighborhoods excluding S are different. Hence, S contains vertices form only one of the partitions. Hence, S is an independent set. Now because H is a bipartite tournament we have that $S \subseteq X$ or $S \subseteq Y$. □

Consider the modular partition $\mathcal{P} = \mathbb{A} \cup \mathbb{B}$ of $X \cup Y$, where $\mathbb{A} = \{A_1, A_2, ...\}$ is a partition of X and $\mathbb{B} = \{B_1, B_2, ...\}$ is a partition of Y.

Lemma 4.3 ([TCHP08])**.** *For a bipartite tournament $H = (X \cup Y, E)$, a modular partition is unique and it can be computed in $\mathcal{O}(|X \cup Y| + |E|)$.*

Now we define the well known notion of quotient graph of a modular partition.

Definition 4.1. *To a modular partition $\mathcal{P} = \{V_1, \ldots, V_\ell\}$ of a directed graph D, we associate a quotient directed graph $\mathcal{D}_\mathcal{P}$, whose vertices $\{v_1, \ldots, v_\ell\}$ are in one to one correspondence with the parts of $\mathcal{P}$. There is an arc (v_i, v_j) from vertex v_i to vertex v_j of $\mathcal{D}_\mathcal{P}$ if and only if $V_i \to V_j$. We denote its vertex set by $V(\mathcal{D}_\mathcal{P})$ and the edge set by $E(\mathcal{D}_\mathcal{P})$.*

We conclude this section with the observation that for a bipartite tournament H, the quotient graph $\mathcal{H}_\mathcal{P}$ corresponding to the modular partition $\mathcal{P}$, is a bipartite tournament.

We refer to the recent survey of Habib and Paul [HP10] for further details and other algorithmic applications of modular decomposition.

4.2 A Cubic Kernel

In this section we show that FASBT admits a polynomial kernel with $\mathcal{O}(k^3)$ vertices. We provide a set of reduction rules and assume that at each step we use the first possible applicable rule. After each reduction rule we discuss its soundness, that is, we prove that the input and output instances are equivalent. If no reduction rule can be used on an instance (H, k), we claim that $|V(H)|$ is bounded by $\mathcal{O}(k^3)$. Throughout this chapter, whenever we say cycle, we mean directed cycle; whenever we speak of a fas, we mean a *minimal fas*, unless otherwise stated.

We start with some simple observations regarding the structure of directed cycles in bipartite tournaments.

Lemma 4.4 ([DGH$^+$10])**.** *A bipartite tournament H has a cycle C if and only if it has a C_4.*

Lemma 4.5. *Let $H = (V, E)$ be a bipartite tournament. If $v \in V$ is part of some cycle C, then it is also part of some C_4.*

Proof. We prove the statement of the lemma by induction on the length of C. If C is a C_4, then there is nothing to prove. Suppose the length of C is longer than 4 and assume that the statement of the lemma holds for all cycles shorter than C. Let $\ldots v, a, b, c \ldots$ be a sub-path of C. If the arc between c and v is (c, v) then, v, a, b, c is a C_4 containing v and if it is (v, c), then we have a shorter cycle which contains v and excludes a and b. By the induction hypothesis, there is a C_4 which contains v. □

4.2.1 Data Reduction Rules

We begin with the following simple rule.

Reduction Rule 4.6. *If $v \in V$ is not part of any cycle in H then remove v, that is, return an instance $(H \setminus v, k)$.*

Lemma 4.7. *Reduction Rule 4.6 is sound and can be applied in polynomial time.*

Proof. Let $H' = H \setminus v$ and F' be a fas of H'. Observe that, deleting v doesn't affect any cycle in H and, every cycle of H is present in H'.

Suppose F' is not an fas of H. Hence, after reversing the arcs of F' in H, there is still a cycle left. But this cycle does not involve v, so it is also present in H' with F' reversed. Therefore there is a cycle in H', which is not removed by F'. Hence, F' is not a fas of H', which is a contradiction.

And by Lemma 4.5, to determine if a vertex is part of any cycle, we only need to check if v is part of some C_4. This can easily be done in polynomial time. □

Reduction Rule 4.8. *If there is an arc $e \in E$, such that there are at least $k+1$ C_4's, which pairwise have only e in common, then reverse e and reduce k by 1. That is, return the instance $(H \setminus \{e\}, k-1)$.*

Lemma 4.9. *Reduction Rule 4.8 is sound and can be applied in polynomial time.*

Proof. Such an arc e must be in any fas of size $\leq k$. Otherwise, we have to pick at least one distinct arc for each of the cycles, and there are $\geq k+1$ of them.

To find such arcs, we use the idea of an *opposite arc*. Two vertex disjoint arcs $e_1 = (a, b)$ and $e_2 = (c, d)$ are called opposite arcs if (a, b, c, d) forms a C_4. Consider the set of opposite arcs of an arc e; this can be easily computed in polynomial time. It is easy to see that an arc e has $\geq k+1$ vertex disjoint opposite arcs if and only if there are $\geq k+1$ C_4 which pairwise have only e in common. Hence, we only need to check if the size of the set of opposite arcs, for any arc e, is larger than k and reverse e if that is the case. □

Let $H = (X \cup Y, E)$ be a bipartite tournament. From now onwards we fix the unique modular partition $\mathcal{P} = \mathbb{A} \cup \mathbb{B}$ of $X \cup Y$, where $\mathbb{A} = \{A_1, A_2, ...\}$ is a partition of X and

$\mathbb{B} = \{B_1, B_2, ...\}$ is a partition of Y. Next we show how a C_4 interacts with the modular partition.

Lemma 4.10. *Let H be a bipartite tournament, then any C_4 in H has each of its vertices in different modules of $\mathcal{P}$.*

Proof. Let u and v be any two vertices of a C_4 in H. If u and v are from different partitions of H then by Lemma 4.2, they cannot be in the same module. And if they are from the same partition, then there is some vertex w from the other partition that comes between them in the cycle as $u \to w \to v$, and hence in the outgoing neighbourhood of one but the incoming neighbourhood of the other. So they cannot be in the same module. □

The main reduction rule that enables us to obtain an $\mathcal{O}(k^3)$ kernel is based on the following crucial lemma. It states that there exists an optimum solution where all arcs between two modules are either a part of the solution or none of them are.

Lemma 4.11. *Let X_1 and Y_1 be two modules of a bipartite tournament $H(X \cup Y, E)$ such that $X_1 \subseteq X$ and $Y_1 \subseteq Y$ and let $E(X_1, Y_1)$ be the set of arcs between these two modules. Let F be any minimal fas of H. Then there exists an fas F^* such that $|F^*| \leq |F|$ and $E(X_1, Y_1) \subseteq F^*$ or $E(X_1, Y_1) \cap F^* = \phi$.*

Proof. Let $X_1 = \{x_1, x_2, ..., x_r\}$ and $Y_1 = \{y_1, y_2, ..., y_s\}$. Let $e = (x_1, y_1)$ be an arc of $E(X_1, Y_1)$. We define, what we call a mirroring operation with respect to the arc e, that produces a solution F' that contains all the arcs of the module if e was in F and does not contain any arc of the module if e was not in F; i.e. the mirroring operation mirrors, the intersection of the solution F with the set containing the arc e and all the arcs incident on the end points of e, to all arcs of the module. To obtain F^* we will mirror the arc that has the smallest intersection. We define this operation formally below.

The operation $mirror(F, e, E(X_1, Y_1))$ returns a subset $F'(e)$ of arcs obtained as follows. Let $F_{X_1Y_1}$ be the set of all arcs in F which have at least one end point in $X_1 \cup Y_1$. Let $e \in E(X_1, Y_1)$ and define $Ext(e) = \{f \in F_{X_1Y_1} | f \cap e \neq \phi, f \notin E(X_1, Y_1)\}$, i.e. $Ext(e)$ is the set of all external edges in $F_{X_1Y_1}$ which are incident on endpoints of e. Let $Ext = \cup_{i=1}^{rs} Ext(e_i)$.

$$\text{Let } F_1(e) = \{(x_i, y)|(x_1, y) \in Ext(e)\} \cup \{(y_i, x)|(y_1, x) \in Ext(e)\}$$
$$\cup\{(y, x_i)|(y, x_1) \in Ext(e)\} \cup \{(x, y_i)|(x, y_1) \in Ext(e)\}.$$
$$\text{Let } F_2(e) = \begin{cases} E(X_1, Y_1) & e \in F \\ \phi & e \notin F. \end{cases}$$

For an edge e define $F'(e) = F - F_{X_1Y_1} \cup F_1(e) \cup F_2(e)$. We claim that $F'(e)$ for any arc $e \in E(X_1, Y_1)$ is a feedback arc set, and that there exists an $e \in E(X_1, Y_1)$ such that $|F'(e)| \leq |F|$, and we will output F^* as $F'(e)$ for that e.

Claim 1. *$F'(e)$ is a feedback arc set for any arc $e \in E(X_1, Y_1)$.*

Proof. Suppose not. Let C be a cycle in the graph $H\{F'(e)\}$ and let $e = (x_s, y_t)$. Replace any vertex $x_j, j \neq s$ by x_s and any vertex $y_j, j \neq y_t$ by y_t in the cycle C to get C' (See Fig. 4.1). We claim that C' is a closed walk (from which a cycle can be obtained) in $H\{F\}$ contradicting the fact that F was a fas for H. That C' is a closed walk is clear because for any arc (u, x_j) or (x_j, u) incident on $x_j, j \neq s$, the corresponding arcs (u, x_s) or (x_s, u) exists as X_1 is a module.

Similarly, for any arc (u, y_j) or (y_j, u) incident on $y_j, j \neq t$, the corresponding arcs (u, y_t) or (y_t, u) exists as Y_1 is a module. Note that C' does not contain arcs incident on any $x_{j\neq s} \in X_1$ or any $y_{j\neq t} \in Y_1$. And the arcs incident on x_s and y_t are not affected by the mirroring operation to obtain F' (i.e. if the edge was initially present in F then it continues to be present in F', and if it was initially absent in F then it continues to be absent in F'). Hence C' is a walk in $H\{F\}$. This completes the proof of this claim. □

Claim 2. *There exists an arc $e \in E(X_1, Y_1)$ such that $|F'(e)| \leq |F|$.*

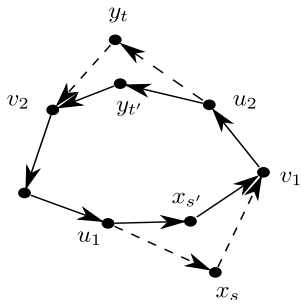

Figure 4.1: Constructing a closed walk from the cycle C.

Proof. Let $e_1, e_2, ...e_{rs}$ be the arcs in $E(X_1, Y_1)$. By the definition of $F'(e_i)$, we have

$$\sum_{i=1}^{rs} |F'(e_i)| = (rs)|F| - (rs)|F_{X_1Y_1}| + \sum_{i=1}^{rs} |F_1(e_i)| + \sum_{i=1}^{rs} |F_2(e_j)|.$$

We will show that the last three terms cancel out. For $x \in X_1$, define $Ext(x) = \{f \in Ext | f \cap x \neq \phi\}$. Whenever an edge incident on x is mirrored by other arcs, $Ext(x)$ is repeated on each of the r vertices in X_1 and there are s edges in $E(X_1, Y_1)$ which are incident on x. Therefore $Ext(x)$ is repeated (rs) times. Similarly we can define $Ext(y)$ and show that it is repeated (rs) times. Therefore the third term $\sum_{i=1}^{rs} |F_1(e_i)| = rs|Ext|$. Furthermore, whenever e is in F, it is repeated (rs) times. This implies that the last term $\sum_{i=1}^{rs} |F_2(e_i)| = rs|F_{X_1Y_1} - Ext|$. Therefore the sum of the last two terms is nothing but (rs) times $|F_{X_1Y_1}|$.

Hence, we have $\sum_{i=1}^{rs} |F'(e_i)| = (rs)|F|$. Therefore, there is an arc $e_i \in E(X_1, Y_1)$ such that $|F'(e_i)| \leq |F|$. Note that to find such an arc e_i, we can compute an arc e_i that has the smallest intersection between F and the arcs incident on its end points. □

This completes the proof of Lemma 4.11. □

From this point, we will only consider fas which either contains all the arcs between two modules or contains none of them. An easy consequence of this lemma is that if S is a module of size at least $k+1$, then a fas of size at most k contains no arc incident on S. This is because between any other module R and S there are at least $k+1$ arcs and they all cannot be in the fas.

Reduction Rule 4.12. *In $\mathbb{A} \cup \mathbb{B}$ truncate all modules of size greater than $k+1$ to $k+1$.*

In other words if S is a module of size more than $k+1$ then delete all but $k+1$ of its vertices arbitrarily.

Lemma 4.13. *Reduction Rule 4.12 is sound.*

Proof. Once we have a modular partition, we can see which modules have size $\geq k+1$ and arbitrarily delete vertices from them to reduce their size to $k+1$.

Next we show that the rule is correct. Suppose H is a bipartite tournament and H' is obtained from H by deleting a single vertex v from some large module S which has $\geq k+2$ vertices. Consider a fas F' of H' of size $\leq k$ and let $S' = S \setminus \{v\}$, which is a module in H'. Since $|S'| \geq k+1$, no arc incident on S' is in F'. Now reverse the arcs of F' in H. If F' is not a fas of H then there is a cycle C of length 4 in H. If C does not contain v then it will be present in H' with F' reversed, which is a contradiction. Otherwise C contains v. Let $u \in S'$ and consider the cycle C' of length 4 obtained from C by replacing v with u. C' is present in H' with F' reversed, which is again a contradiction.

Now we can use induction on the number of vertices deleted from a module to show that truncating a single module does not change the solution. We can then use induction on the number of truncated modules to show that the rule is correct. □

4.2.2 Analysis of the Kernel Size

We apply the above Reduction Rules 4.6, 4.8 and 4.12 exhaustively (until no longer possible) and obtain a reduced instance (H', k') of FASBT from the initial instance (H, k). Observe that we will not keep applying these rules indefinitely to an instance because either the size of the graph or the parameter k drops after each application of the reduction rules. For *brevity we abuse notation* and denote the reduced instance also by (H, k). As before we fix the unique modular partition $\mathcal{P} = \mathbb{A} \cup \mathbb{B}$ of $X \cup Y$, where $\mathbb{A} = \{A_1, A_2, ...\}$ is a partition of X and $\mathbb{B} = \{B_1, B_2, ...\}$ is a partition of Y. Let $\mathcal{H}_\mathcal{P}$ be the corresponding quotient graph.

The following lemma lists some properties of the quotient graph.

Lemma 4.14. *Let $\mathcal{H}_\mathcal{P}$ be the quotient graph of H. Then the following hold:*

(a) Every vertex in $\mathcal{H}_\mathcal{P}$ is part of some C_4.

(b) For each arc $e \in E(\mathcal{H}_\mathcal{P})$ there are $\leq k$ C_4 in $\mathcal{H}_\mathcal{P}$ which pairwise have e as the only is the common arc.

(c) If H has a fas of size $\leq k$ then $\mathcal{H}_\mathcal{P}$ has a fas of size $\leq k$.

(d) For all $u, v \in V(\mathcal{H}_\mathcal{P})$, $N^+(u) \neq N^+(v)$ or equivalently $N^-(u) \neq N^-(v)$.

Proof. (a) Consider a vertex $u \in \mathcal{H}_\mathcal{P}$. Let S be the corresponding module in H. Consider a vertex $x \in S$. If x is not part of any cycle then rule 4.6 applies, which is a contradiction. Otherwise x is part of some cycle in H and by Lemma 4.5 it is part of some C_4 x, y, z, p in H. By Lemma 4.10, each of x, y, z, p lies in a different module. Let v, w, t be vertices in $\mathcal{H}_\mathcal{P}$ corresponding to the modules containing y, z, p respectively. Then by definition of $E(\mathcal{H}_\mathcal{P})$, (u, v, w, t) forms a C_4 in $\mathcal{H}_\mathcal{P}$.

(b) Suppose there were an arc $e \in E(\mathcal{H}_\mathcal{P})$ which is the only pairwise common arc of $\geq k+1$ C_4 in $\mathcal{H}_\mathcal{P}$. Consider such a collection of C_4 in $\mathcal{H}_\mathcal{P}$. For each vertex in this collection, there is a module in H. Pick one vertex from each module and the arcs between these vertices in H. Let $e' \in H$ be the arc corresponding to e. This gives us a collection of $\geq k+1$ C_4 in H, which pairwise have only e' in common. But then Rule 4.8 applies.

(c) Let F' be any fas of H. Therefore $|F'| \leq k$. Let F be the corresponding set of arcs in $\mathcal{H}_\mathcal{P}$, so $|F| \leq k$. Every arc $(u, v) \in F$ corresponds to the set of all arcs between the modules S_u and S_v in F'. We reverse F in $\mathcal{H}_\mathcal{P}$ and F' in H respectively. If F is not a fas of $\mathcal{H}_\mathcal{P}$, then there is a $C_4 = (u, v, w, t) \in \mathcal{H}_\mathcal{P}$ with F reversed. Then consider the vertices $x, y, z, p \in H$ where $x \in S_u, y \in S_v, z \in S_w, p \in S_t$. By Lemma 4.11 the arcs $(u, v), (v, w), (w, t), (t, u)$ in $\mathcal{H}_\mathcal{P}$ with F reversed, imply the arcs $(x, y), (y, z), (z, p), (p, x)$ in H with F' reversed. This implies that F' is not a fas of H which is a contradiction.

(d) $\mathcal{H}_\mathcal{P}$ is a Bipartite Tournament. Hence, if for some $u, v \in \mathcal{H}_\mathcal{P}$, $N^+(u) = N^+(v) \implies N^-(u) = N^-(v)$. But then S_u is not a maximal module, because we can add any vertex from S_v to it. This is a contradiction to the fact that $\mathbb{A} \cup \mathbb{B}$ was a maximal modular partition. □

Let $\mathcal{H}_\mathcal{P}$ be the quotient graph of H. Let F be an fas of $\mathcal{H}_\mathcal{P}$ of size at most k. Let T be the topological sort of $\mathcal{H}_\mathcal{P}$ obtained after reversing the arcs of F. If we arrange the vertices of $\mathcal{H}_\mathcal{P}$ from left to right in order of T, then only the arcs of F go from right to left. For an arc $e \in F$, the *span* of e is the set of vertices which lie between the endpoints of e in T. We call a vertex v an *affected vertex* if there is some arc in F which is incident on v. Otherwise we call v an *unaffected vertex*.

Lemma 4.15. *If v is an unaffected vertex, then there exists an arc $e \in F$ such that v is in the span of e.*

Proof. Let v be any unaffected vertex in $\mathcal{H}_\mathcal{P}$. By Lemma 4.14(a) there is a C_4 u, v, w, t which contains v. Since v is unaffected the order $u \to v \to w$ is fixed in T. Therefore the arc $t \to u$ goes from right to left in T. Hence, v is contained in the span of (t, u). □

Lemma 4.16. *Let u and v be two unaffected vertices from the same vertex partition of the bipartite tournament $\mathcal{H}_\mathcal{P}$ such that u occurs before v in T. Then there exists w from the other partition which lies between u and v in T.*

Proof. Since $\mathcal{H}_\mathcal{P}$ is the quotient graph of H and u and v are different modules, therefore there is some vertex w in the other partition such that $(u, w), (w, v) \in E(\mathcal{H}_\mathcal{P})$. Furthermore, these two arcs are not in the fas F as u and v are unaffected. Hence, w comes between u and v in T. □

Lemma 4.17. *There are at most $2k + 2$ unaffected vertices in the span of any arc $e \in F$.*

Proof. Let the span of e contain $2k + 3$ unaffected vertices. Then without loss of generality assume that $k + 2$ of these vertices come from the partition A. Then there are at least $k + 1$ vertices of partition B and by Lemma 4.16 between each pair of consecutive vertices of A lies a vertex of B. This gives us $k + 1$ C_4's which pairwise have only e in common. But then the graph H contains an arc e' for which there are $k + 1$ C_4's which pairwise have only e' in common. This contradicts the fact that H was reduced with respect to Rule 4.8. □

Reduction Rule 4.18. *If $\mathcal{H}_\mathcal{P}$ contains more than $2k^2 + 2k$ vertices then return NO.*

Lemma 4.19. *Reduction Rule 4.18 is sound.*

Proof. By Lemma 4.14 (c), if H has a fas of size $\leq k$, then $\mathcal{H}_{\mathcal{P}}$ has a fas of size $\leq k$. If there are more than $2k^2 + 2k$ vertices in $\mathcal{H}_{\mathcal{P}}$, then there is some arc e whose span contains more than $2k + 2$ unaffected vertices; this contradicts Lemma 4.17. □

Hence, we may assume that $\mathcal{H}_{\mathcal{P}}$ contains $\mathcal{O}(k^2)$ vertices.

Lemma 4.20. *H contains $\mathcal{O}(k^3)$ vertices.*

Proof. Each vertex in $\mathcal{H}_{\mathcal{P}}$ corresponds to a module in H and any module in H has size at most $k + 1$ (due to Reduction Rule 4.12). Hence, there are $\mathcal{O}(k^3)$ vertices in H. □

Lemma 4.20 implies the following theorem.

Theorem 4.2. FASBT *admits a polynomial kernel with $\mathcal{O}(k^3)$ vertices.*

4.3 Discussion

In this chapter we obtained a polynomial kernel for FASBT with $\mathcal{O}(k^3)$ vertices. This illustrates the main idea behind a kernelization algorithm, which is a series of reduction rules, designed to cleverly exploit the structural properties of the input, exhaustively applied, reduces the instance to a kernel, in polynomial time. In later sections of this thesis, we shall see polynomial kernels for a few network design problems. We also remark that the kernel obtained in this chapter generalizes to multi-partite tournaments. Recently Guo and Xiao [XG12] gave kernel for this problem with $\mathcal{O}(k^2)$ vertices.

Chapter 5

An introduction to Matroids

Matroids generalize the notion of "independence" in linear algebra, to set systems. It was introduced by Whitney [Whi35], and since then it has grown into an important area, with connections to many branches of mathematics and theoretical computer science. Matroids are important mathematical objects in the theory of algorithms and combinatorial optimization. Often an algorithm for a class of matroids gives us an "algorithmic meta theorem", which gives a unified solution to a number of other problems. For example, it is known that any problem which admits a greedy algorithm can be embedded into a matroid and finding a minimum (maximum) weighted independent set corresponds to finding a solution to the problem. Other important examples are Matroid Intersection and Matroid Parity problems, which encompasses several combinatorial optimization problems such as Bipartite Matching, 2-Edge Disjoint Spanning Trees and Arborescence. We refer to the textbook of Oxley [Oxl06] for an introduction to this topic.

Recently, matroids have been applied in designing several FPT, Kernelization and Exact algorithms [FLPS14, FLS14, KW12, KW14, Mar09, GMP13, SZ14], which has resolved a number of long standing open problems in these areas. In particular, the notion of *representative families* over linear matroids has been the key to many of these results. Representative Sets were introduced by Bollobás [Bol65] for extremal set systems and later generalized to subspaces of a vector space by Lovász [Lov77] (see also [Fra82]). Their results are corner-stones in extremal set theory with numerous applications in graph and

hypergraph theory, combinatorial geometry and theoretical computer science. We refer to Section 9.2.2 of [Juk11], surveys of Tuza [Tuz94, Tuz96], and a blog post of Gil Kalai[1] for more information on the theorems and their applications.

Dynamic programming with representative sets over linear matroids is the second main technique in many of the results in this thesis. In this chapter we give a short overview of Matroids and Representative Sets. In the following chapter we shall illustrate this technique via a FPT algorithm for k-PATH. In later chapters we shall apply this technique to obtain FPT and Exact algorithms for network design problems.

5.1 Matroids

We begin with the definition of a matroid.

Definition 5.1. *A pair $M = (E, \mathcal{I})$, where E is a ground set and $\mathcal{I}$ is a family of subsets (called independent sets) of E, is a* matroid *if it satisfies the following conditions:*

(I1) $\emptyset \in \mathcal{I}$.

(I2) *If $A' \subseteq A$ and $A \in \mathcal{I}$ then $A' \in \mathcal{I}$.*

(I3) *If $A, B \in \mathcal{I}$ and $|A| < |B|$, then $\exists\ e \in (B \setminus A)$ such that $A \cup \{e\} \in \mathcal{I}$.*

The axiom (I2) is also called the hereditary property and a pair $(E, \mathcal{I})$ satisfying only (I2) is called a hereditary family. An inclusion wise maximal set of $\mathcal{I}$ is called a *basis* of the matroid. Using axiom (I3) it is easy to show that all the bases of a matroid have the same size. This is called the *rank* of the matroid M, and is denoted by $\mathsf{rank}(M)$. The notion of rank may be extended to any subset of E. For any $X \subseteq E$ we define $\mathsf{rank}_M(X)$ to be $\mathsf{max}\{|A| \mid A \subseteq X \text{ such that } A \text{ is independent in } M\}$. This is called the rank function of the matroid M. The following proposition can be proved easily by using the axiom (I3) of matroids.

Proposition 5.1. *Let $M = (E, \mathcal{I})$ be a matroid. If $A \in \mathcal{I}$ such that $|A| < \ell$, $B \subseteq E$ and $r_M(A \cup B) = \ell$ then there is a subset $B' \subseteq B$, $|B'| = \ell - |A|$ such that $A \cup B' \in \mathcal{I}$.*

[1] http://gilkalai.wordpress.com/2008/12/25/lovaszs-two-families-theorem/

An important notion in matroid theory is that of *Dual Matroids* which is defined as follows.

Dual of a Matroid. Let $M = (E, \mathcal{I})$ be a matroid. The *dual* of M is the matroid $M^* = (E, \mathcal{I}^*)$ where $\mathcal{I}^* = \{A \subseteq E \mid \mathsf{rank}(E \setminus A) = \mathsf{rank}(M)\}$

It can be easily verified the above definition indeed produces a valid matroid. Further, for any basis B^* of M^*, the set $E \setminus B^*$ is a basis of M and vice versa. This implies that $\mathsf{rank}(M^*) = |E| - \mathsf{rank}(M)$, and in fact $(M^*)^* = M$.

5.1.1 Linear Matroids and Representable Matroids

Let A be a matrix over an arbitrary field $\mathbb{F}$ and let E be the set of columns of A. We define the matroid $M = (E, \mathcal{I})$ as follows. A set $X \subseteq E$ is independent (that is $X \in \mathcal{I}$) if the corresponding columns are linearly independent over $\mathbb{F}$. The matroids that can be defined by such a construction are called *linear matroids*, and if a matroid can be defined by a matrix A over a field $\mathbb{F}$, then we say that the matroid is representable over $\mathbb{F}$. That is, a matroid $M = (E, \mathcal{I})$ is representable over a field $\mathbb{F}$ if there are vectors in $\mathbb{F}^r$ corresponding to each of the elements in the ground set such that, the linearly independent subsets of these vectors correspond precisely to the independent sets of the matroid. Here, $r = \mathsf{rank}(M)$. A matroid $M = (E, \mathcal{I})$ is called *representable* or *linear* if it is representable over some field $\mathbb{F}$. The dual matroid M^* of a linear matroid M is also linear and we have the following theorem about the representation of dual matroids.

Theorem 5.2 ([Oxl06, Section 2.2])**.** *Let M be a linear matroid. Given a matrix A over a field $\mathbb{F}$ which represents M, we can obtain a matrix A^* which represents the dual matroid M^* in polynomially many field operations.*

5.1.2 Direct Sum of Matroids

Let $M_1 = (E_1, \mathcal{I}_1)$, $M_2 = (E_2, \mathcal{I}_2)$, $\ldots$, $M_t = (E_t, \mathcal{I}_t)$ be t matroids with $E_i \cap E_j = \emptyset$ for all $1 \leq i \neq j \leq t$. The direct sum $M_1 \oplus \cdots \oplus M_t$ is a matroid $M = (E, \mathcal{I})$ with $E := \bigcup_{i=1}^{t} E_i$ and $X \subseteq E$ is independent if and only if for all $i \leq t$, $X \cap E_i \in \mathcal{I}_i$. Let A_i

be the representation matrix of $M_i = (E_i, \mathcal{I}_i)$. Then,

$$A_M = \begin{pmatrix} A_1 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_t \end{pmatrix}$$

is a representation matrix of $M_1 \oplus \cdots \oplus M_t$. The correctness of this construction is proved in [Mar09].

Proposition 5.2 ([Mar09, Proposition 3.4])**.** *Given representations of matroids $M_1, \ldots, M_t$ over the same field $\mathbb{F}$, a representation of their direct sum can be found in polynomial time.*

5.1.3 Truncation of a Matroid

The *t-truncation* of a matroid $M = (E, \mathcal{I})$ is a matroid $M' = (E, \mathcal{I}')$ such that $S \subseteq E$ is independent in M' if and only if $|S| \leq t$ and S is independent in M (that is $S \in \mathcal{I}$).

Proposition 5.3 ([Mar09, Proposition 3.7])**.** *Given a matroid M with a representation A over a finite field $\mathbb{F}$ and an integer t, a representation of the t-truncation M' can be found in randomized polynomial time.*

In a later chapter, we shall see a deterministic polynomial time algorithm for obtaining a representation of the truncation of a linear matroid.

5.1.4 Deletions and Contractions

Deletion in a matroid. Let $M = (E, \mathcal{I})$ be a matroid and $F \subseteq E$. Then the deletion of F in M, is a matroid denoted by $M \setminus F$, whose independent sets are those independent sets of M, that are contained in $E \setminus F$.

It is clear that if M is a linear matroid then $M \setminus F$ is also a linear matroid, and a representation can be found by deleting the columns corresponding to F in a representation of M.

Contraction in a matroid. Let $M = (E, \mathcal{I})$ be a matroid and $F \subseteq E$. Then the contraction of M by F, is a matroid, denoted by M/F, on the ground set $E \setminus F$, whose rank function is defined as, $r_{M/F}(A) = r_M(A \cup F) - r_M(F)$ for any $A \subseteq E \setminus F$.

We have the following proposition which relates the contraction of matroid to deletions in the dual matroid.

Proposition 5.4 ([Oxl06]). *Let $M = (E, \mathcal{I})$ be a matroid and $F \subseteq E$. Then $M/F = (M^* \setminus F)^*$.*

Hence, if M is a linear matroid then M/F is also a linear matroid and a representation of it may be computed in polynomial time.

5.1.5 Uniform and Partition Matroids

Uniform Matroid. A pair $M = (E, \mathcal{I})$ over an n-element ground set E, is called a uniform matroid if the family of independent sets is given by $\mathcal{I} = \{A \subseteq E \mid |A| \leq k\}$, where k is some constant. This matroid is denoted as $U_{n,k}$. Every uniform matroid is linear and can be represented over a finite field by a $k \times n$ matrix A_M where the $A_M[i, j] = j^{i-1}$.

$$A_M = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ 1 & 2^2 & 3^2 & \cdots & n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 2^{k-1} & 3^{k-1} & \cdots & n^{k-1} \end{pmatrix}$$

Observe that for A_M to be representable over a finite field $\mathbb{F}$, we need that the determinant of any $k \times k$ submatrix of A_M must not vanish over $\mathbb{F}$. The determinant of any $k \times k$ submatrix of A_M is upper bounded by $k! \times n^{k^2}$ (this follows from the Laplace expansion of determinants). Thus, choosing a field $\mathbb{F}$ of size larger than $k! \times n^{k^2}$ suffices. However, it can be shown that $U_{n,k}$ is actually representable over all fields with at least $n + 1$ elements [Oxl06]. When $k = 1$, the uniform matroid has a representation over any field with at least 2 elements [Oxl06].

Partition Matroid. A partition matroid $M = (E, \mathcal{I})$ is defined by a ground set E being partitioned into (disjoint) sets $E_1, \ldots, E_\ell$ and by ℓ non-negative integers $k_1, \ldots, k_\ell$. A set $X \subseteq E$ is independent if and only if $|X \cap E_i| \leq k_i$ for all $i \in \{1, \ldots, \ell\}$. Observe that a partition matroid is a direct sum of uniform matroids $U_{|E_1|,k_1}, \cdots, U_{|E_\ell|,k_\ell}$. So by Proposition 5.2 we have the following.

Proposition 5.5. *A representation of a partition matroid can be constructed in polynomial time, over a field of size* $|E| + 1$ *Here* $k = \mathsf{max}\{k_1, k_2, \ldots, k_\ell\}$.

However note that when $k = 1$, the partition matroid has a representation over any field with at least 2 elements [Oxl06].

5.1.6 Graphic Matroids

Given a graph G, the graphic matroid of G, $M = (E, \mathcal{I})$, is defined by the ground set $E = E(G)$ and $F \subseteq E(G)$ is in $\mathcal{I}$ if it forms a spanning forest in the graph G. A graphic matroid is representable over any field of size at least 2. Consider the matrix A_M with a row for each vertex $i \in V(G)$ and a column for each edge $e = ij \in E(G)$. In the column corresponding to $e = ij$, all entries are 0, except for a 1 in i or j (arbitrarily) and a -1 in the other. For a digraph D, we define the graphic matroid with respect to the underlying graph of D.

Proposition 5.6 ([Oxl06])**.** *Graphic matroids are representable over any field of size at least* 2.

5.2 Representative Sets

In this section we give brief introduction to representative sets.

Definition 5.3 (q**-Representative Family).** *Given a matroid* $M = (E, \mathcal{I})$ *and a family* $\mathcal{S}$ *of subsets of* E*, we say that a subfamily* $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ *is* q-representative *for* $\mathcal{S}$ *if the following holds: for every set* $Y \subseteq E$ *of size at most* q*, if there is a set* $X \in \mathcal{S}$ *disjoint from* Y

with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{X} \cup Y \in \mathcal{I}$. If $\hat{\mathcal{S}} \subseteq \mathcal{S}$ is q-representative for $\mathcal{S}$ we write $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$.

In other words if some independent set in $\mathcal{S}$ can be extended to a larger independent set by q new elements, then there is a set in $\widehat{\mathcal{S}}$ that can be extended by the same q elements. We define a weighted version of q-representative families, which is useful for solving problems where we are looking for objects of maximum or minimum weight. Note that the weight function can be any arbitrary non-negative function on the independent sets of the matroid.

Definition 5.4 (Min/Max q-Representative Family). *Given a matroid $M = (E, \mathcal{I})$, a family $\mathcal{S}$ of subsets of E and a non-negative weight function $w : \mathcal{S} \to \mathbb{N}$ we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is* min q-representative *(*max q-representative*) for $\mathcal{S}$ if the following holds: for every set $Y \subseteq E$ of size at most q, if there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y with*

(i) $\widehat{X} \cup Y \in \mathcal{I}$; and

(ii) $w(\widehat{X}) \leq w(X)$ $(w(\widehat{X}) \geq w(X))$.

We use $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ $(\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S})$ to denote a min q-representative (max q-representative) family for $\mathcal{S}$.

The following lemmas establish some properties of representative sets which are useful in algorithmic applications.

Lemma 5.7 ([FLS14])**.** *Let $M = (E, \mathcal{I})$ be a matroid and $\mathcal{S}$ be a family of subsets of E. If $\mathcal{S}' \subseteq^q_{rep} \mathcal{S}$ and $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}'$, then $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$.*

Lemma 5.8 ([FLS14])**.** *Let $M = (E, \mathcal{I})$ be a matroid and $\mathcal{S}$ be a family of subsets of E. If $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_\ell$ and $\widehat{\mathcal{S}}_i \subseteq^q_{rep} \mathcal{S}_i$, then $\cup_{i=1}^{\ell} \widehat{\mathcal{S}}_i \subseteq^q_{rep} \mathcal{S}$.*

Lemma 5.9 ([FLS14])**.** *Let $M = (E, \mathcal{I})$ be a matroid of rank k and $\mathcal{S}_1$ be a p_1-family of independent sets, $\mathcal{S}_2$ be a p_2-family of independent sets, $\widehat{\mathcal{S}}_1 \subseteq^{k-p_1}_{rep} \mathcal{S}_1$ and $\widehat{\mathcal{S}}_2 \subseteq^{k-p_2}_{rep} \mathcal{S}_2$. Then $\widehat{\mathcal{S}}_1 \bullet \widehat{\mathcal{S}}_2 \subseteq^{k-p_1-p_2}_{rep} \mathcal{S}_1 \bullet \mathcal{S}_2$.*

Given a representable matroid $M = (E, \mathcal{I})$ of rank $k = p + q$ with its representation matrix A_M and a p-family of independent sets $\mathcal{S} = \{S_1, \ldots, S_t\}$, a non-negative weight function $w : \mathcal{S} \to \mathbb{N}$, we can compute $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ and $\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$ of size $\binom{p+q}{p}$ deterministically in time $\mathcal{O}\left(\binom{p+q}{p} tp^\omega + t\binom{p+q}{q}^{\omega-1}\right)$.

Theorem 5.5 ([FLS14]). *Let $M = (E, \mathcal{I})$ be a linear matroid of rank $p + q = k$, $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a p-family of independent sets and $w : \mathcal{S} \to \mathbb{N}$ be a non-negative weight function. Then there exists $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$) of size $\binom{p+q}{p}$. Moreover, given a representation A_M of M over a field $\mathbb{F}$, we can find $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$) of size at most $\binom{p+q}{p}$ in $\mathcal{O}\left(\binom{p+q}{p} tp^\omega + t\binom{p+q}{q}^{\omega-1}\right)$ operations over $\mathbb{F}$.*

In Theorem 5.5 we assumed that $\mathsf{rank}(M) = p + q$. However, one can obtain a similar result even when $\mathsf{rank}(M) > p + q$. To do so, we first obtain a truncation of the matroid M to rank $p + q$ via Proposition 5.3. As noted before, the previously known algorithms for the truncation of a linear matroid were randomized, and hence the resulting algorithm is randomized as well.

Theorem 5.6 ([FLS14]). *Let $M = (E, \mathcal{I})$ be a linear matroid and let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a p-family of independent sets and w is a non-negative weight function on S. Then there exists $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$ of size $\binom{p+q}{p}$, where $p + q \leq \mathsf{rank}(M)$. Furthermore, given a representation A_M of M over a field $\mathbb{F}$, there is a randomized algorithm computing $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$) in $\mathcal{O}\left(\binom{p+q}{p} tp^\omega + t\binom{p+q}{q}^{\omega-1}\right)$ operations over $\mathbb{F}$.*

We shall see in a later chapter that, by using certain algebraic tools, we can make the above algorithm *deterministic*. In the case of uniform matroids one can avoid matrix multiplication required in the above algorithms and obtain much faster results. This is because in uniform matroids, $\mathcal{A}' \subseteq \mathcal{A}$ of the family $\mathcal{A}$ *q-represents* $\mathcal{A}$ if for every set B of size q such that there is an $A \in \mathcal{A}$ and $A \cap B = \emptyset$, there is a set $A' \in \mathcal{A}'$ such that $A' \cap B = \emptyset$.

Theorem 5.7 ([FLS14]). *There is an algorithm that given a family $\mathcal{A}$ of sets of size p over a universe U of size n and an integer q, computes in time $\mathcal{O}(|\mathcal{A}| \cdot (\frac{p+q}{q})^q \cdot \log n)$ a subfamily $\mathcal{A}' \subseteq A$ such that $|\mathcal{A}'| \leq \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ and $\mathcal{A}'$ q-represents $\mathcal{A}$.*

The proof of the above theorem requires a construction of a small *separating family* for separating subsets of size p and subsets of size q of the universe. See [FLS14] for more details.

Chapter 6

Illustration: An FPT algorithm for k-PATH

In this chapter, we shall see a simple algorithm for the k-PATH problem by using the technique of dynamic programming with representative sets. This algorithm is based on the work of Fomin et al. [FLS14]. The problem is defined as follows.

k-PATH	Parameter: k
Input: A graph G on n vertices and m edges, and an integer k.	
Question: Is there a path in G with k or more vertices ?	

This problem is well studied in parameterized complexity. Monien[] gave the first FPT algorithm for this problem, and interestingly enough this algorithm too relied on representative sets, although the algorithm for computing representative sets was much slower. It was conjectured that this problem is solvable in polynomial time for $k = \log n$ in an n vertex graph. This was proved in the seminal paper of Alon et.al [AYZ95], who introduced the technique of *color coding* to solve this problem. A number of results followed, improving the running time via more sophisticated techniques. Several randomized algorithms were also designed, which use sophisticated algebraic techniques [BHKK10, Wil09, KW09]. Currently the fastest known randomized algorithm runs in time $\mathcal{O}^*(1.66^k)$ [BHKK10]. Recently, Fomin et al. [FLS14, FLPS14] gave efficient algorithms for computing representative sets,

and then applied their algorithm in designing a deterministic algorithm with a running time of $\mathcal{O}^*(2.619^k)$. Interestingly, their results apply the techniques of Alon et al. in computing the representative sets. It has been further improved to an algorithm running in time $\mathcal{O}^*(2.597^k)$ [Zeh15].

In this chapter, we shall see a simple algorithm based on the approach of Fomin et al. [FLS14]. While the running time of this algorithm can be significantly improved, our focus will be on illustrating the technique of the algorithm, which is a key algorithmic technique in this thesis.

6.1 The Algorithm

Let us assume that the k-path in G starts from a fixed vertex $s \in V(G)$. This is easy to ensure, by adding a new vertex s to the graph and making it adjacent to all the vertices. The the new graph has a path on $k+1$ vertices if and only if the original graph has a path on k vertices.

Consider the following dynamic programming algorithm, $\mathcal{A}$, which computes a k-path in G which starts from s, if such a path exists. let $T[i]$ be the collection the those paths in G which start from s and contain exactly i vertices. For a vertex $v \in V(G)$, let $T[i, v]$ denote the subset of $T[i]$ which contains all the paths which end at the vertex v. Observe that $T[i, v]$ partition $T[i]$, i.e. $T[i] = \biguplus_{v \in V(G)} T[i, v]$. For $i = 1$, we have $T[1, s] = \{\{s\}\}$ and $T[1, v] = \emptyset$ for any $v \neq s$. Now, given the paths $T[i-1, v]$ for every $v \in V(G)$, we can compute each $T[i, v]$ as follows.

$$T[i, v] = \{P \in T[i-1, u] \text{ where } u \in N(v)\} \bullet \{\{v\}\}$$

It is easy to see the following.

Proposition 6.1. *Algorithm $\mathcal{A}$ computes a k-path, if there is such a path in G. Further, the running time of this algorithm is $\mathcal{O}^*(n^k)$.*

Observe that there could be as many as $\binom{n}{i-1}$ paths in $T[v, i-1]$, and hence computing

$T[v,i]$ could take $\mathcal{O}^*(\binom{n}{i})$ time. However, by using representative sets, we can "compress" the number of paths in $T[i,v]$ to $\mathcal{O}(2^k)$. To do so, first we have to show that the partial solutions are independent in a linear matroid. Then we must show that a representative set of a collection of partial solutions always contains a candidate which can be extended to a full solution. And finally, we show that the representative sets of the partial solutions can be computed at each stage, by using the representative sets from the previous stages. Let us consider each of these steps.

Consider the matroid $\mathcal{M}$ which is a uniform matroid of rank k over the ground set $V(G)$, i.e. any subset of k vertices is independent in $\mathcal{M}$. Then $T[i,v]$ is a collection of independent sets of size i in $\mathcal{M}$. Next, we say that a path $P \in T[i]$ is *extendable to a k-path* if and only if, there is a path Q with $k-i+1$ vertices such that, $P \cap Q = \{v\}$ where v is the end vertex of P and the start vertex of Q. Observe that $P+Q$ is a k-path in G. Let $\widehat{T}[i,v] \subseteq^{k-i}_{rep} T[i,v]$ with respect to the above matroid, and let $\widehat{T}[i] = \bigcup_{v \in V(G)} \widehat{T}[i,v]$. We have the following lemma.

Lemma 6.2. *$P \in T[i]$ is extendable to a k-path if and only if there is a $\widehat{P} \in \widehat{T}[i]$ which is extendable to a k-path.*

Proof. Consider the forward direction of the claim. Suppose P is extendable to a k-path by some path Q, and let v be the common vertex between them. Observe that $P \in T[v,i]$ and Q is a path on $k-i+1$ vertices. If $P \in \widehat{T}[v,i]$ then $\widehat{P} = P$ is the required path. Otherwise, consider the sets P and $Q-v$ and observe that they are of size i and $k-i$ respectively, and furthermore they are disjoint. Therefore by the definition of representative sets, there is a path $\widehat{P} \in \widehat{T}[i,v]$ such that $\widehat{P}$ and $Q-v$ are disjoint. Observe that $\widehat{P}$ is a path from s to v of length i, and hence $\widehat{P}+Q$ is a k-path in G.

The reverse direction of this claim is trivial since every $\widehat{P} \in \widehat{T}[i,v]$ is also contained in $T[i]$. □

From the above lemma, we conclude that it is sufficient to store $\widehat{T}[i]$ in place of $T[i]$, which we can compute by Theorem 5.5 in $\mathcal{O}^*(2^{\mathcal{O}(k)})$ time. However we must still address the issue of computing $\widehat{T}[i]$ when we are only given $\widehat{T}[i-1]$. Let $T'[i,v] = \{P \in \widehat{T}[i-1,u] \text{ where } u \in$

$N(v)\} \bullet \{\{v\}\}$. Then by Theorem 5.5 and Lemma 5.7, we can easily show the following.

Lemma 6.3. *$\widehat{T}'[i,v] \subseteq^{k-1}_{rep} T[i,v]$ and it contains at most 2^k paths. Further it may be computed in $\mathcal{O}^*(2^{\mathcal{O}(k)})$.*

Let $\widehat{T}'[i] = \bigcup_{v \in V(G)} \widehat{T}'[i,v]$, and from the two lemmas above we conclude that it is enough to store $\widehat{T}'[i]$ in place of $T[i]$. Thus we obtain a new algorithm $\widehat{\mathcal{A}}$ from $\mathcal{A}$, where we compute and store $\widehat{T}'[i]$ for every $i \in [k]$. Further, in every step of the algorithm we require a time $\mathcal{O}^*(2^{\mathcal{O}(k)}$. Therefore, we have the following theorem.

Theorem 6.1. *Let G be a graph and k be an integer. Then we can find a k-path in G in time $\mathcal{O}^*(2^{\mathcal{O}(k)})$.*

6.2 Discussion

The above algorithm illustrates the technique of dynamic programming with representative sets over linear matroids. The first step is to design a dynamic programming algorithm, whose partial solutions are independent sets in an appropriately chosen matroid. The next step is to show that, retaining a representative set of the collection of partial solutions at each step is sufficient. Finally, we must show that one can compute the representative sets at each step efficiently. Then from the properties of the representative sets, we obtain an efficient algorithm for the problem. A key benefit of using the above technique is that designing a fast FPT or Exact algorithm for a problem boils down to designing a dynamic programming algorithm which is "embeddable" in a linear matroid. In this later sections of this thesis, we shall see some algorithm for network design problems using dynamic programming over representative sets. The algorithms will have the same basic structure as illustrated above, but the dynamic program and the matroids will be much more involved.

We also remark that the running time of the algorithm presented in this section can be significantly improved by using more sophisticated analysis and methods for computing representative sets.

Chapter 7

Connectivity Matroids

In this chapter, we review some matroids that are associated with graph connectivity, and some of their algorithmic applications. We also mention two matroids which arise from the notion of matroid connectivity.

7.1 Co-Graphic Matroids

The co-graphic matroid characterized those edge subsets of a connected graph which may be safely deleted without disconnecting the graph.

Definition 7.1. *The* co-graphic matroid *of a connected graph G is defined as $M = (E(G), \mathcal{I})$ where $\mathcal{I} = \{S \subseteq E(G) \mid (G \setminus S)$ is connected $\}$.*

It is clear from the definition that, the co-graphic matroid of a graph G is the dual of the graphic matroid of G, and therefore it is a linear matroid. The rank of the cographic matroid of a connected graph G is $(|E(G)| - |V(G)| + 1)$. Further, a representation of the co-graphic matroid of G can be found in polynomial time over any field [Mar09, Oxl06] over any field.

Co-graphic matroids are useful in edge deletion problems where preserving the connectivity of the graph is one of the constraints. We may imagine that the solutions to a given instance are embedded in a co-graphic matroid related to the input. Then combined with

other structural properties, we may obtain an algorithm for the problem. We shall see an example of such an algorithm in a later chapter.

7.2 Gammoids

Gammoids characterize those vertex subsets which are reachable from a fixed set of source vertices, by vertex disjoint paths. They are defined as follows.

Definition 7.2. *Let G be any graph or digraph, $S \subseteq V(G)$ be a set of source vertices and $T \subseteq V(G)$. Then the gammoid on T in G with respect to S, is the pair $(T, \mathcal{I})$ where $\mathcal{I} = \{A \subseteq T \mid$ there are $|A|$ vertex disjoint paths from T to A in $G\}$.*

A gammoid is called a strict gammoid, when $T = V(G)$. Observe that any gammoid may be obtained from a strict gammoid by deletion. The following useful result was proved by Ingleton and Piff in 1972 [IP73](also see [Oxl06, Mar09]), shows that strict gammoids and transversal matroids are duals.

Lemma 7.1. *Let D be a digraph and $S \subseteq V(D)$. Let $T = V(D)$ and $V' = V(D) \setminus S$ be two copies of vertex subsets. Then there is bipartite graph $G = (T \uplus V', E)$ such that the strict gammoid with respect to D and S is the dual of the transversal matroid M_G on the ground set T. Moreover, given D and S, there is a polynomial time algorithm to output the graph G.*

Further, they are both linear matroids [Oxl06] and a representation can be obtained in *randomized* polynomial time [Mar09]. In a later chapter, we shall see moderately exponential deterministic algorithms for computing representations of transversal matroids and gammoids.

Recently gammoids have been applied in obtaining a polynomial kernels for a number of problems such as Odd Cycle Transversal, Multiway Cut and Almost 2-SAT [KW12]. While we go into the details of these algorithms, the main idea behind these results is to identify a subset of vertices which must be present in every solution. This set is computed by using the representative sets algorithm over a linear matroid,

which is a direct sum of several matroids including gammoids. Finally, it is shown that there is always a solution contained in the above subset of vertices. Hence, reducing the graph down to the above subset gives a kernel.

7.3 Linkage Matroids

Linkage matroids are an analogue of gammoids based on matroid connectivity, which was introduced by Geelan et al. [GGW07]. In gammoids the underlying object is a graph, and the independent sets correspond to those vertex subsets which are well connected to a fixed subset of vertices. In linkage matroids the underlying object is a matroid, and the independent sets are those subsets of the ground set which are well connected to a fixed subset of the ground set. Before we define these matroids, we need to define the notion of connectivity in a matroid.

Let $\mathcal{M} = (E, I)$ be a matroid and let r be the rank function of this matroid. The *connectivity function*, λ, of the matroid $\mathcal{M}$ is defined as follows.

$$\lambda(X) = r(X) + r(E \setminus X) - r(M), \text{ for any } X \subseteq E$$

Observe that $\lambda(X) = \lambda(\overline{X})$. For two disjoint subsets A and B of E, we define the connectivity of S and T in $\mathcal{M}$ as follows.

$$\kappa(A, B) = \mathsf{min}_{X \subseteq E} \{\lambda(X) \mid A \subseteq X,\ B \subseteq (E \setminus X)\}$$

This function was defined by Tutte, who then gave an analogue of Menger's theorem in graphs, in matroids [Tut65]. Let S be a subset of E. For any $X \subseteq (E \setminus S)$ consider the following function.

$$\alpha_S(X) = \kappa(X, S)$$

Geelan et al. [GGW07] showed the following theorem.

Theorem 7.3. *The function α_S is the rank function of a matroid over the ground set $E \setminus S$.*

This matroid is called the *linkage matroid* of $\mathcal{M}$, with respect to the set S. The independent sets of this matroids are those subsets of $E \setminus S$, which are well connected to the set S in $\mathcal{M}$.

7.4 Tangle Matroids

Before we define these matroids, let us define the notion of a separation in a matroid. A partition $(X, \overline{X})$ of E is called a *separation* of order $\lambda(X)$ in $\mathcal{M}$. Tangles are a collection of separations in a matroid with certain properties. A *tangle* $\mathcal{T}$ of order k in the matroid $\mathcal{M}$ is a collection of subsets of E which satisfies the following.

(i) for every $X \in \mathcal{T}$, $\lambda(X) < k$,

(ii) for every separation $(X, \overline{X})$ of order strictly less than k, either $X \in \mathcal{T}$ or $\overline{X} \in \mathcal{T}$,

(iii) if $X \uplus Y \uplus Z$ is a partition of E, then at least one of these sets is not a member of $\mathcal{T}$.

(iv) and for any $e \in E$, $E - e \notin \mathcal{T}$.

Tangles in matroids are an analogue of tangles in graphs. The members of the tangle $\mathcal{T}$ are called *small sets*. Now consider the following function.

$$\phi(X) = \begin{cases} \min\{\lambda(Y) \mid Y \in \mathcal{T} \text{ such that } X \subseteq Y\} & \text{if such a } Y \text{ exists,} \\ k & \text{otherwise} \end{cases}$$

Geelan et al. showed the following theorem [GGRW06].

Theorem 7.4. *The function ϕ is the rank function of a matroid over E.*

This matroid is called the *tangle matroid* of $\mathcal{M}$.

7.5 Discussion

In this thesis, we will see some algorithmic application of Co-graphic matroids and Gammoids. Some of the other matroids such as Graphic matroids and Partition matroid

will also find algorithmic applications. The Linkage matroid and Tangle matroid have been recently defined and the problem of determining if these matroids are representable, whenever the underlying matroid is representable, remains an open problem. Futhermore, characterization of the duals of these matroids and other properties of these matroids also open problems.

Chapter 8

An introduction to Network Design Problems with Connectivity Constraints

Network design problems with connectivity constraints are a well studied class of problems in computer science. Typically the aim is to design a cost effective network that can survive communication failures, which may be caused by any number of things such as a hardware or software breakage, human error or a broken link between two network components. Such problems often modeled as graphs, with the nodes representing the network components (such as computers, routers, etc.), and edges representing the communication links between the components. Designing a network which satisfies certain connectivity constraints, or augmenting a given network to a certain connectivity are important and well studied problems in network design. In terms of graph theory, these problems problems correspond to, finding a spanning subgraph of a weighted complete graph which satisfies given connectivity constraints and, augmenting the given graph with additional edges so that it satisfies the given constraints, respectively. These problems have many important real world applications and have been extensively studied in the framework of approximation algorithms [KN10, GK11, BDK09].

Designing a minimum cost network which connects all the nodes, is known as Minimum

Spanning Tree(MST) problem. Borůvka in 1926 presented an algorithm for MST which was motivated by design of economic electric power distribution networks [NMN01], and subsequently several other algorithms were designed for this problem. However such a network fails on the failure of a single link. This leads to the question of designing a minimum cost network which can survive one or more link failures. Observe that, such a network must be λ-connected, in order to survive $\lambda - 1$ link failures.

Adding a minimum number of edges to make the graph satisfy certain connectivity constrains is known as minimum augmentation problem. Minimum augmentation was studied in the design of survivable networks [FC70, JG86], and it has applications in other areas such as data security [Gus88, Kao96].

All these problems are special cases of the Generalized Steiner Network problems which is defined as follows.

Generalized Steiner Network (GSN)

Input: A graph or digraph G, a cost function $w : E(G) \to \mathbb{N}^*$, a requirement function $r : V(G) \times V(G) \to \mathbb{N}^*$.

Question: Find a subgraph H of minimum total cost such that, there are $r(u, v)$ edge disjoint paths from u to v in H for every $(u, v) \in V(G) \times V(G)$.

It encapsulates a number of network design problems with connectivity constraints, for particular weight and requirement functions. The study of these problems in the framework of parameterized complexity, has only recently began. Furthermore, nothing better the trivial bruteforce algorithms are known for these many of these problems in the framework of exact algorithms. In this chapter, we collect a small list of these problems and discuss some relevant parameterizations.

8.1 Network Augmentation

In network augmentation problems, the goal is to augment the connectivity of an input graph or digraph by add a set of links between the vertices of minimum total cost. Augmentation

problems may be modeled as GSN in the following way. The edge set $E(G)$ is partitioned as $E_0 \uplus L$ such that, the weight function assigns a weight 0 to every edge in E_0 and an arbitrary weight to edges in L. Then by choosing requirement function appropriately, we get the various network augmentation problems. Note that the set of links is called *unrestricted* if it is equal to $V(G) \times V(G)$ and the weight function assigns the same weight to every link. Otherwise, the set of links is said to be *restricted*.

The requirement function is called *uniform* if it assigns the same value to every pair of vertices, otherwise it is called a *general requirement* function. In the special case, when the set of links is unrestricted and the requirements are uniform, i.e $r(u, v) = \lambda$ for some constant $\lambda > 0$, this is called the Minimum Augmentation problem. Watanabe and Nakamura gave a polynomial time algorithm for solving this problem in an undirected graph [WNN89]. Frank gave a polynomial time algorithm for the same problem in directed graphs [Fra92a]. In undirected graph, this problem is solvable in polynomial time even for a general requirement function [Fra92a]. However in the weighted case or when the augmenting set must be a subset of a given set of links, the problem becomes NP-Hard [Fra92a]. Even the restricted case of augmenting the edge connectivity of a graph from $\lambda - 1$ to λ is NP-hard [BJG08]. A 2-approximation may be obtained for these problems, by choosing a suitable weight function and applying the algorithm of [KV94]. We refer to [KN10, BJG08, Khu97, BDK09] for more details. Let us now describe some of these problems.

In a directed graph, setting $r(u, v) = 1$ for every pair of vertices along with a restricted link set, gives us following problem.

Strong Connectivity Augmentation

Input: A digraph G, a subset L of $V(G) \times V(G)$ and a weight function w on L.

Question: Find a $F \subseteq L$ of minimum total cost such that $G \cup F$ is strongly connected.

The undirected version of the above problems is called the 2-Connectivity Augmentation problem. When the requirements are uniform $r(u, v) = \lambda$ for a constant $\lambda > 0$, we get the following problem.

λ-Connectivity Augmentation
Input: A graph or digraph G, a subset L of $V(G) \times V(G)$ and a weight function w on L.
Question: Find a $F \subset L$ of minimum total cost such that $G \cup F$ is λ-edge connected.

When it is additionally guaranteed that, G is $(\lambda - 1)$-edge connected in the above problem, it is called the Augmentation by One problem. Let us note that this problem is NP-hard even all the weights on L are $\{0, 1\}$, by a reduction from Hamiltonian Cycle, Similarly, for any constant $\alpha < \lambda$, such that G is guaranteed to be $(\lambda - \alpha)$-edge connected, we have the Augmentation by α problem.

We may also consider a "steiner" version of the above problem, where we are concerned with only a subset of the vertices of the input graph.

Steiner λ-connectivity Augmentation
Input: A graph or digraph G, a subset T of $V(G)$, a subset L of $V(G) \times V(G)$ and a weight function w on L.
Question: Find a $F \subset L$ of minimum total cost such that in $G \cup F$, there are λ edge disjoint paths between any pair of vertices in T.

8.1.1 Parameterizations of Network Augmentation Problems

A straightforward parameterization of Network Augmentation Problems is the size of a minimum augmenting set. A few results are known under this parameterization. The first parameterized algorithm for a connectivity augmentation problem was given by Nagamochi [Nag03], who gave an $2^{\mathcal{O}(k \log k)}|V|^{\mathcal{O}(1)}$ algorithm for Augmentation by One in the case when the weights on the links are identical and λ is odd. Guo and Uhlmann [GU10] gave a kernel with $\mathcal{O}(k^2)$ vertices and links for the same case. Most recently, Marx and Vegh [MV15] studied the problem in its full generality and gave a kernel with $\mathcal{O}(k)$ vertices, $\mathcal{O}(k^3)$ links and weights of $(k^6 \log k)$ bit integers. This as well as other previous results lead to an algorithm with running time $2^{\mathcal{O}(k \log k)}|V|^{\mathcal{O}(1)}$, even for unweighted version of the problem. In a later chapter, we shall see improved FPT algorithms for this problem.

One could also consider as a parameter, the cardinality of the complement of a solution, i.e. $|L \setminus F|$. In a later chapter, we consider Augmentation by One in undirected graphs with respect to this parameter. This is a much more difficult problem. Nonetheless, we obtain an FPT algorithm and a polynomial kernel for it.

Currently, no other results are known for any other augmentation problems in the realm of parameterized complexity. It is however suspected that Strong Connectivity Augmentation is FPT.

8.2 Network Optimization

In a network optimization problem, the goal is to find a subgraph of the input graph of digraph of minimum total costs that meets the given connectivity constraints.

When the requirements are uniform, i.e. $r(u, v) = \lambda$ for every pair of vertices where $\lambda > 0$, we obtain the following problem.

Minimum λ-edge connected Subgraph

Input: A graph or digraph G which is λ-edge connected and a weight function w on the edges of the graph.

Question: Find a spanning subgraph H of G of minimum total cost such that H is λ connected.

This problem is NP-hard, and a there is 2-approximation algorithm is known for it [KV94]. In the special case when the weights are 1 or ∞, i.e. we wish to find a minimum spanning λ-connected subgraph, a $1 + \frac{2}{k}$ approximation may be obtained in polynomial time [CT00].

For the special case of $\lambda = 1$ in directed graphs we obtain the Minimum Strongly Connected Spanning Subgraph (MSCSS) problem.

Minimum Strongly Connected Spanning Subgraph

Input: A graph or digraph G which is strongly connected and a weight function w on the edges of the graph.

Question: Find a spanning strong subgraph H of G of minimum total cost.

Observe that this problem is a generalization of the Hamiltonian Cycle problem, and furthermore the classical Minimum Equivalent Graph problem also reduces to this problem in polynomial time.

Let T be subset of $V(G)$, and let requirement function be $r(u,v) = 1$ if $u, v \in T$ and 0 otherwise. Then this gives the well known Steiner Tree problem. In digraphs where the requirement function is such that $r(u,v) = 2$ if $u, v \in T$ and 0 otherwise, we obtain the Minimum Strong Steiner Subgraph problem, which is a generalization of MSCSS.

Minimum Strong Steiner Subgraph
Input: A digraph G, a subset T of the vertices and a weight function w on the arcs.
Question: Find a minimum cost strongly connected subgraph H, such that $T \subseteq V(H)$.

Finally we have a "steiner" version of the above problems.

λ-connected Steiner Subgraph
Input: A graph or digraph G, a subset T of $V(G)$ and a weight function w on $E(G)$.
Question: Find a subgraph H of G of minimum total cost such that in H, there are λ edge disjoint paths between any pair of vertices in T.

8.2.1 Parameterizations of Network Optimization Problems

Generally, the standard parameter for a parameterized problem is the size of a solution to a given instance. For example for MSCSS, it will be the number of arcs in a minimum solution H. If G is a graph on n vertices, then observe that the number of arcs in H must be at least n, as H is also strongly connected. Thus either $\ell \geq n$ or the given instance, (G, ℓ), is a No instance. Hence, if we use the solution size as a parameter then the problem is trivially FPT – just try all arcs subsets of size at most ℓ. Clearly, this is at most $\binom{n^2}{\ell} \leq \binom{\ell^2}{\ell}$, and thus MSCSS is FPT with respect to ℓ with an algorithm running in time $2^{\mathcal{O}(\ell \log \ell)} + \mathcal{O}(m+n)$. So probably a more meaningful question is whether there is a subgraph H on at most $n + k$ arcs, where k is the parameter (such parameterizations are

called above/below guarantee parameterization, see [GY12, MRS09] for an introduction to this topic). However, it is not possible that we can even have an algorithm of the form $n^{g(k)}$, for any arbitrary function g. The reason for this is as follows. A digraph G has an equivalent sub-digraph of size n if and only if it has a directed Hamiltonian cycle. Thus an algorithm of the form $n^{g(k)}$ is polynomial for $\ell = 0$ and hence in polynomial time we could detect whether an input digraph has a directed Hamiltonian cycle and that would imply P=NP! So we must look for other relevant parameters.

It is known that a digraph is strong if and only if it contain an out-branching and an in-branching rooted at some vertex $v \in V(D)$ [BJG08, Proposition 12.1.1]. This implies that for MSCSS, the graph H has at most $\leq 2n-2$. Thus, a natural question is whether one can obtain an a solution with at most $2n-2-k$, with k being the parameter. Bang-Jensen and Yeo [BJY08] studied this problem and showed that it is FPT by designing an algorithm with running time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$. However, notice that if the number of arcs in the input digraph is less than $2n-2-k$ then this algorithm just returns the instance. For example consider a digraph G which has maximum total degree 3 (the number of in-neighbors and out-neighbors), and therefore the total number of arcs is bounded by $\frac{3}{2}n$. Hence, this parameterization is not helpful for such instances. For higher connectivity, it can be shown that any λ connected digraph has at most $2\lambda(n-1)$ arcs, when the graph is λ-edge connected. Then the corresponding question is whether there is a λ connected subgraph on at most $2\lambda(n-1)-k$ arcs. It is clear that, this parameterization is not helpful when the number of arcs is bounded by say $\frac{3}{2}\lambda(n-1)$.

Now, consider the set of arcs $D = A(G) \setminus A(H)$, which may be safely removed from the graph without affecting the strong connectivity. It is clear that the total weight of D is maximized when the graph H is the minimum solution. Hence, the cardinality of the set D could be chosen as a parameter. We shall see in a later chapter, this parameterization of MSCSS admits a polynomial kernel. Minimum λ-edge Connected Subgraph and Steiner Tree can also be parameterized in the same way. Currently, no other results are known for any other network optimization problems in the realm of parameterized complexity.

8.3 Discussion.

There are many other types of network design problem, which we have not discussed here. For example, problems with distance or latency constraints, degree constraints etc instead of connectivity constraints are also studied. Furthermore, network design problems are also studied in a game-theoretic setting where a collection of agents try to route commodities. See [GK11] for more details. One could also consider other structural parameters such as the treewidth of the graph for these problems.

There are also vertex-connectivity versions of the above problems, i.e. the requirement function asks for $r(u, v)$ vertex disjoint paths from u to v. These are substantially harder and very few results are known about them outside the framework of approximation algorithms.

To conclude, there are many interesting network design problems, with important practical applications, whose parameterized complexity has not been explored. Such an endeavor will perhaps require the development of new tools and techniques in parameterized complexity.

Deterministic Matroid Algorithms

Chapter 9

Deterministic Truncation of Linear Matroids and Applications

9.1 Introduction

In this chapter, we give a deterministic polynomial time algorithm for obtaining a representation of the truncation of a linear matroid. This is the first deterministic algorithm for this problem. In later sections of this chapter we shall see the applications of this algorithm in *de-randomizing* some of the results of Marx and Fomin et. al. [FLS14, Mar09].

Let us recall the definition of the truncation of a matroid. For a matroid $M = (E, \mathcal{I})$, the *k-truncation* of M is a matroid $M' = (E, \mathcal{I}')$ such that for any $A \subseteq E$, $A \in I'$ if and only if $|A| \leq k$ and $A \in \mathcal{I}$. Given a linear representation of a matroid M of rank n over a universe of size m (which has a representation matrix of dimension $n \times m$), the problem of finding a linear representation of the k-truncation of the matroid $M = (E, \mathcal{I})$, can be stated as the problem of obtaining a *rank k truncation* of the matrix M[1]. The rank k-truncation of a $n \times m$ matrix M, is a $k \times m$ matrix M_k such that for every subset $I \subseteq \{1, \ldots, m\}$ of size at most k, the set of columns corresponding to I in M has rank $|I|$ if and only of the corresponding set of columns in M_k has rank $|I|$. We can think of finding a rank k-truncation of a matrix as a dimension reduction problem such that linear independence

[1]We abuse notation slightly and denote both the matroid and it's representation matrix by M

among all sets of columns of size at most k is preserved. This problem is a variant of the more general *dimensionality reduction* problem, which is a basic problem in many areas of computer science such as machine learning, data compression, information processing and others. In dimensionality reduction, we are given a collection of points (vectors) in a high dimensional space, and the objective is to map these points to points in a space of small dimension while preserving some property of the original collection of points. For an example, one could consider the problem of reducing the dimension of the space, while preserving the pairwise distance, for a given collection of points. Using the Johnson-Lindenstrauss Lemma this can be done approximately for any collection of m points, while reducing the dimension of the space to $\mathcal{O}(\log m)$. Here, we study dimensionality reduction under the constraint that linear independence of any subcollection of size up to k of the given set of vectors is preserved. Here the objective is to map the set of column vectors of M (which lie in a space of dimension n) to vectors in a space of dimension k such that, any set S of column vectors of M with $|S| \leq k$ are linearly independent if and only if the corresponding set of vectors in the k-dimensional vector space are linearly independent.

As observed by Marx [Mar09], a common way to obtain a rank k-truncation of a matrix M, is to left-multiply M by a random matrix of dimension $k \times n$ (with entries from a field of an appropriate size). Then using the Schwartz-Zippel Lemma one can show that, the product matrix is a k-truncation of the matrix M with high probability.

Proposition 9.1 ([Mar09, Proposition 3.7])**.** *Given a matroid M with a representation A over a finite field $\mathbb{F}$ and an integer t, a representation of the t-truncation of M, say M', can be found in randomized polynomial time.*

Note that there is no known polynomial time algorithm to check the output of this algorithm, as such an algorithm must verify that every independent set of size at most k in M must be preserved. This raises a natural question of whether there is a deterministic algorithm for computing k-truncation of a matrix. In this chapter we settle this question by giving a polynomial time deterministic algorithm to solve this problem. We further show that for many fields, the k-truncation matrix can be represented over a finite degree extension.

A related notion is the *ℓ-elongation* of a matroid, where $\ell > \mathsf{rank}(M)$. It is defined as

the matroid $M' = (E, \mathcal{I}')$ such that $S \subseteq E$ is a basis of M' if and only if, it contains a basis of M and $|S| = \ell$. Note that the rank of the matroid M' is ℓ. As a corollary of the above algorithm, we obtain a deterministic polynomial time algorithm for computing the representation of the ℓ-elongation of any linear matroid.

9.2 Preliminaries

Let us review some definitions and results which are required for this chapter.

9.2.1 Fields and Polynomials

In this section we review some definitions and properties of fields. We refer to any graduate text on algebra for more details. The cardinality or the size of a field is called its *order*. It is well known that rational numbers $\mathbb{Q}$ and real numbers $\mathbb{R}$ are fields of infinite order. For every prime number p and a positive integer ℓ, there exists a finite field of order p^ℓ. For a prime number p, the set $\{0, 1, \ldots, p-1\}$ with addition and multiplication modulo p forms a field, which we denote by $\mathbb{F}_p$. Such fields are known as *prime fields*. Let $\mathbb{F}$ be a finite field and $\mathbb{F}[X]$ be the ring of polynomials in X over $\mathbb{F}$. For the ring $\mathbb{F}[X]$ we can define a field $\mathbb{F}(X)$ which is called the *field of fractions* of $\mathbb{F}[X]$ as follows. The elements of $\mathbb{F}(X)$ are of the form $P(X)/Q(X)$ where $P(X), Q(X) \in \mathbb{F}[X]$ and $Q(X)$ is not a zero polynomial. The addition and multiplication operations from $\mathbb{F}[X]$ are extended to $\mathbb{F}(X)$ in the usual way. The degree of a polynomial $P(X) \in \mathbb{F}(X)$ is the highest exponent on indeterminate X with a nonzero coefficient in $P(X)$. We will use $\mathbb{F}[X]^{<n}$ to denote the set the polynomials in $\mathbb{F}[X]$ of degree $< n$.

For a field $\mathbb{F}$, we use $+_{\mathbb{F}}$ and $\times_{\mathbb{F}}$ to denote the addition and multiplication operations. (Often we write $a + b$ and ab when the context is clear.) The *characteristic* of a field, denoted by $\mathrm{char}(\mathbb{F})$, is defined as least integer m such that $\sum_{i=1}^{m} 1 = 0$. For fields such as $\mathbb{R}$ where there is no such m, the characteristic is defined to be 0. For a finite field $\mathbb{F} = \mathbb{F}_{p^\ell}$, let $\mathbb{F}^* = \mathbb{F} \setminus \{0\}$. This is called the multiplicative group of $\mathbb{F}$ which is a cyclic group and has a generator $\alpha \in \mathbb{F}^*$. Every element of $\mathbb{F}^*$ can be written as α^i for some number i. The

element α is called a *primitive element* of the field $\mathbb{F}$. We say that an element $\beta \in \mathbb{F}$ has *order* r, if r is the least integer such that $\beta^r = 1$.

A polynomial $P(X) \in \mathbb{F}[X]$ is called *irreducible* if it cannot expressed as a product of two other non-trivial polynomials in $\mathbb{F}[X]$. Let $P(X)$ be an irreducible polynomial in $\mathbb{F}[X]$, of degree ℓ. Then $\mathbb{K} = \frac{\mathbb{F}[X]}{P(X)} = \mathbb{F}[X](\text{mod } P(X))$ is also a field. It is of order $|\mathbb{F}|^\ell$ and characteristic of $\mathbb{K}$ is equal to the characteristic of $\mathbb{F}$. We call $\mathbb{K}$ a *field extension* of $\mathbb{F}$ of degree ℓ. All finite fields are obtained as extensions of prime fields, and for any prime p and positive integer ℓ there is exactly one finite field of order p^ℓ up to isomorphism.

9.2.2 Vectors and Matrices

A vector v over a field $\mathbb{F}$ is an array of values from $\mathbb{F}$. A collection of vectors $\{v_1, v_2, \ldots, v_k\}$ are said to be linearly dependent if there exist values $a_1, a_2, \ldots, a_k$, not all zeros, from $\mathbb{F}$ such that $\sum_{i=1}^{k} a_i v_i = 0$. Otherwise these vectors are called linearly independent.

For a matrix $A = (a_{ij})$ over a field $\mathbb{F}$, the row set and the column set are denoted by $\mathbf{R}(A)$ and $\mathbf{C}(A)$ respectively. For $I \subseteq \mathbf{R}(A)$ and $J \subseteq \mathbf{C}(A)$, $A[I, J] = \big(a_{ij} \mid i \in I,\ j \in J\big)$ means the submatrix (or minor) of A with the row set I and the column set J. The matrix is said to have dimension $n \times m$ if it has n rows and m columns. For a matrix A (or a vector v) by A^T (or v^T) we denote its *transpose*. Note that each column of a matrix is a vector over the field $\mathbb{F}$. The rank of a matrix is the cardinality of the maximum sized collection of columns which are linearly independent. Equivalently, the rank of a matrix is the maximum number k such that there is a $k \times k$ submatrix whose determinant is non-zero. We can use the definition of rank of a matrix to certify the linear independence of a set of vectors. A collection of n vectors $v_1, v_2, \ldots, v_n$ are linearly independent if and only if the matrix M formed by $v_1, v_2, \ldots, v_n$ as column vectors have rank n. Determinant of a $n \times n$ matrix A is denoted by $\det(A)$ and is defined as

$$\det(A) = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^{n} A[i, \sigma(i)].$$

Here, S_n is the set of all permutations of $\{1, \ldots, n\}$ and $\text{sgn}(\sigma)$ denotes the signature of

the permutation σ.

9.2.3 Derivatives

Recall the definition of the formal derivative $\frac{d}{dx}$ of a function over $\mathbb{R}$. We denote the k-th formal derivative of a function f by $f^{(k)}$. We can extend this notion to finite fields. Let $\mathbb{F}$ be a finite field and let $\mathbb{F}[X]$ be the ring of polynomials in X over $\mathbb{F}$. Let $P \in \mathbb{F}[X]$ be a polynomial of degree $n-1$, i.e. $P = \sum_{i=0}^{n-1} a_i X^i$ where $a_i \in \mathbb{F}$. Then we define the *formal derivative* of as $P' = \sum_{i=1}^{n-1} i a_i X^{i-1}$. We can extend this definition to the k-th formal derivative of P as $P^{(k)} = (P^{(k-1)})'$. Note that higher derivatives are defined iteratively using lower derivatives, thus they are also called iterated formal derivatives.

Formal derivatives continue to carry many of their properties in $\mathbb{R}$ to finite fields $\mathbb{F}$. However not all properties carry through. For example, in $\mathbb{F}_3[X]$ the polynomial $P(X) = X^3$ has all derivatives 0. To remedy such problems, we require the notion of *Hasse Derivatives*. For a polynomial $P(X) \in \mathbb{F}[X]$, the i-th Hasse derivative $D^i(P)$ is defined as the coefficient of Z^i in $P(X+Z)$.

$$P(X+Z) = \sum_{i=0}^{\infty} D^i(P(X)) Z^i$$

We note some important properties of Hasse derivatives and how they relate to formal derivatives. We refer to [DKSS09] and [Gol03] for details.

Lemma 9.2. *Let $\mathbb{F}$ be a finite field of characteristic p, $P, Q \in \mathbb{F}[x]$. Then the following holds :*

1. *D^k is a linear map from $\mathbb{F}[X]$ to $\mathbb{F}[X]$.*
2. *Let k be some number and let $k!$ be non-zero in $\mathbb{F}$ (i.e. $k! \neq 0 \mod p$). Then $k!.D^k(P) = P^{(k)}$. In particular $D^0(P) = f$ and $D^1(P) = P^{(1)}$.*

Observe that the second statement in the above lemma shows that the value of k-th hasse derivatives and k-th formal derivatives differ only by a multiplicative value whenever $k!$ is non-zero in $\mathbb{F}$. In particular when $k < p$, (the characteristic of the field $\mathbb{F}$) then $k!$ is always non-zero in $\mathbb{F}$. In our setting this is always the case.

9.2.4 Determinants

Proposition 9.3 (Generalized Laplace expansion)**.** *For an $n \times n$ matrix A and $J \subseteq \mathbf{C}(A) = [n]$, it holds that*

$$\mathsf{det}(A) = \sum_{I \subseteq [n], |I| = |J|} (-1)^{\sum I + \sum J} \mathsf{det}(A[I, J]) \mathsf{det}(A[\bar{I}, \bar{J}])$$

We refer to [Mur00, Page number 33, Proposition 2.1.3] for a proof of the above identity. We always assume that the number of rows in the representation matrix A_M of M over a field $\mathbb{F}$ is equal to the $\mathsf{rank}(M) = \mathsf{rank}(M)$. Else, using Gaussian elimination we can obtain a matrix of the desired kind in polynomial time. See [Mar09, Proposition 3.1] for details.

9.3 Matrix Truncation

In this section we give our one of the main results. We start with an introduction to our tools and then we give two results that give rank k-truncation of the given matrix M.

9.3.1 Tools and Techniques

In this subsection we collect some known results, definitions and derive some new connections among them that will be central to our results.

Polynomials and Vectors

Let $\mathbb{F}$ be a field. The set of polynomials $P_1(X), P_2(X), \ldots, P_k(X)$ in $\mathbb{F}[X]$ are said to be *linearly independent* over $\mathbb{F}$ if there doesn't exist $a_1, a_2, \ldots, a_k \in \mathbb{F}$, not all zeros such that $\sum_{i=1}^{k} a_i P_i(X) \equiv 0$. Otherwise they are said to be linearly dependent.

Definition 9.1. *Let $P(X)$ be a polynomial of degree at most $n-1$ in $\mathbb{F}[X]$. We define the vector v corresponding to the polynomial $P(X)$ as follows: $v[j] = c_j$ where $P(X) = \sum_{j=1}^{n} c_j x^{j-1}$. Similarly given a vector v of length n over $\mathbb{F}$, we define the polynomial $P(X)$*

in $\mathbb{F}[X]$ *corresponding to the vector* v *as follows:* $P(X) = \sum_{j=1}^{n} v[j]x^{j-1}$.

The next lemma will be key to several proofs later.

Lemma 9.4. *Let* $v_1, \ldots, v_k$ *be vectors of length* n *over* $\mathbb{F}$ *and let* $P_1(X), \ldots, P_k(X)$ *be the corresponding polynomials respectively. Then* $P_1(X), \ldots, P_k(X)$ *are linearly independent over* $\mathbb{F}$ *if and only if* $v_1, v_2, \ldots, v_k$ *are linearly independent over* $\mathbb{F}$.

Proof. For $i \in \{1 \ldots k\}$, let $v_i = (c_{i1}, \ldots, c_{in})$ and let $P_i(X) = \sum_{j=1}^{n} c_{ij}x^{j-1}$ be the polynomial corresponding to v_i.

We first prove the forward direction of the proof. For a contradiction, assume that $v_1, \ldots, v_k$ are linearly dependent. Then there exists $a_1, \ldots, a_k \in \mathbb{F}$, not all zeros, such that $\sum_{i=1}^{k} a_i v_i = 0$. This implies that for each $j \in \{1, \ldots n\}$, $\sum_{i=1}^{k} a_i v_i[j] = 0$. Since $v_i[j] = c_{ij}$, we have $\sum_{i=1}^{k} a_i c_{ij} = 0$, which implies that $\sum_{i=1}^{k} a_i c_{ij} x^{j-1} = 0$. Summing over all these expressions we get $\sum_{i=1}^{k} a_i P_i(X) \equiv 0$, a contradiction. This completes the proof in the forward direction.

Next we prove the reverse direction of the lemma. To the contrary assume that $P_1(X), \ldots, P_k(X)$ are linearly dependent. Then there exists $a_1, \ldots, a_k \in \mathbb{F}$, not all zeros, such that $\sum_{i=1}^{k} a_i P_i(X) \equiv 0$. This implies that for each $j \in \{1, \ldots, n\}$, the coefficients of x^{j-1} satisfy $\sum_{i=1}^{k} a_i c_{ij} = 0$. Since c_{ij} is the j-th entry of the vector v_i for all i and j, we have $\sum_{i=1}^{k} a_i v_i = 0$. Thus the vectors $v_1, \ldots, v_k$ are linearly dependent, a contradiction to the given assumption. This completes this proof. □

We will use this claim to view the column vectors of a matrix M over a field $\mathbb{F}$ as elements in the ring $\mathbb{F}[X]$ and in the field of fractions $\mathbb{F}(X)$. We shall then use properties of polynomials to deduce properties of these column vectors and vice versa.

The Wronskian matrix

Let $\mathbb{F}$ be a field with characteristic at least n. Consider a collection of polynomials $P_1(X), \ldots, P_k(X)$ from $\mathbb{F}[X]$ of degree at most $n - 1$. We define the following matrix, called the *Wronskian*, of $P_1(X), \ldots, P_k(X)$ as follows.

$$W(P_1, \ldots, P_k) = \begin{pmatrix} P_1(X) & P_2(X) & \ldots & P_k(X) \\ P_1^{(1)}(X) & P_2^{(1)}(X) & \ldots & P_k^{(1)}(X) \\ \vdots & \vdots & \ddots & \vdots \\ P_1^{(k-1)}(X) & P_2^{(k-1)}(X) & \ldots & P_k^{(k-1)}(X) \end{pmatrix}_{k \times k}$$

Note that, the determinant of the above matrix actually yields a polynomial. For our purpose we will need the following well known result.

Theorem 9.2 ([BD10, GV87, Mui82])**.** *Let $\mathbb{F}$ be a field and $P_1(X), \ldots, P_k(X)$ be a set of polynomials from $\mathbb{F}[X]^{<n}$ and let $\mathrm{char}(\mathbb{F}) > n$. Then $P_1(X), \ldots, P_k(X)$ are linearly independent over $\mathbb{F}$ if and only if the Wronskian determinant $\det(W(P_1(X), \ldots, P_k(X))) \not\equiv 0$ in $\mathbb{F}[X]$.*

The notion of Wronskian dates back to 1812 [Mui82]. We refer to [BD10, GV87] for some recent variations and proofs. The switch between usual derivatives and Hasse derivatives multiplies the Wronskian determinant by a constant, which is non-zero as long as $n < \mathrm{char}(\mathbb{F})$, and thus this criterion works with both notions. Observe that the Wronskian determinant is a polynomial of degree at most nk in $\mathbb{F}[X]$. Thus to test if such a polynomial (of degree d) is identically zero, we only need to evaluate it at $d+1$ arbitrary points of the field $\mathbb{F}$, and check if it is zero at all those points.

The *Folded* Wronskian matrix

The above definition of Wronskian requires us to compute derivatives of degree $(n-1)$ polynomials. As noted earlier, they are well defined only if the underlying field has characteristic greater than or equal to n. However, we might have matrix over fields of small characteristic. For these kind of fields, we have the notion of *Folded Wronskian* which was recently introduced by Guruswami and Kopparty in the context of subspace design [GK13].

Consider a collection of polynomials $P_1(X), \ldots, P_k(X)$ from $\mathbb{F}[X]$ of degree at most $(n-1)$. Further, let $\mathbb{F}$ be of order at least $n+1$, and α be an element of $\mathbb{F}^*$. We define the the

α-*folded Wronskian*, of $P_1(X), \ldots, P_k(X)$ as follows.

$$W_\alpha(P_1, \ldots, P_k) = \begin{pmatrix} P_1(X) & P_2(X) & \ldots & P_k(X) \\ P_1(\alpha X) & P_2(\alpha X) & \ldots & P_k(\alpha X) \\ \vdots & \vdots & \ddots & \vdots \\ P_1(\alpha^{k-1}X) & P_2(\alpha^{k-1}X) & \ldots & P_k(\alpha^{k-1}X) \end{pmatrix}_{k \times k}$$

As before, the determinant of the above matrix is a polynomial of degree at most nk in $\mathbb{F}[X]$. Note that Guruswami and Kopparty defined the above notion of α-folded Wronskian only for those α that are *primitive element* of $\mathbb{F}$. In particular, they proved the following result about α-folded Wronskians [GK13, Lemma 12].

Theorem 9.3 ([GK13])**.** *Let $\mathbb{F}$ be a field of order $> n$, α be a primitive element of $\mathbb{F}$ and let $P_1(X), \ldots, P_k(X)$ be a set of polynomials from $\mathbb{F}[X]^{<n}$. Then $P_1(X), \ldots, P_k(X)$ are linearly independent over $\mathbb{F}$ if and only if the α-folded Wronskian determinant $\det(W_\alpha(P_1, \ldots, P_k)) \not\equiv 0$ in $\mathbb{F}[X]$.*

Theorem 9.3 requires a primitive element α of the underlying field $\mathbb{F}$. However, finding a primitive element in a finite field is a non-trivial problem and currently there are no deterministic polynomial time algorithm known for this problem. To overcome this difficulty, we prove a generalization of Theorem 9.3. The next theorem relaxes the requirement that α must be a primitive element of the field $\mathbb{F}$, and only requires that α be an element of *sufficiently large* order. Finding an element of large order is slightly easier task than finding a primitive element of a finite field $\mathbb{F}$, and this will be sufficient for our purposes. We note that the following theorem was also proved by Forbes and Shpilka [FS12], who applied it in construction pseudorandom objects for polynomial identity testing and other related problems. Here, we provide a different proof of this

Theorem 9.4. *Let $\mathbb{F}$ be a field, α be an element of $\mathbb{F}$ of order $> (n-1)(k-1)$ and let $P_1(X), \ldots, P_k(X)$ be a set of polynomials from $\mathbb{F}[X]^{<n}$. Then $P_1(X), \ldots, P_k(X)$ are linearly independent over $\mathbb{F}$ if and only if the α-folded Wronskian determinant $\det(W_\alpha(P_1, \ldots, P_k)) \not\equiv 0$ in $\mathbb{F}[X]$.*

In what follows we build towards the proof of Theorem 9.4. For the sake of brevity, we use

W_α to denote the matrix $W_\alpha(P_1, \ldots, P_k)$. We use the notation $Z_1, Z_2, \ldots, Z_k$ to denote the columns of W_α. That is, $Z_i = (P_i(X), P_i(\alpha X), \ldots, P_i(\alpha^{k-1}X))^T$. Observe that W_α is a matrix over the field $\mathbb{F}(X)$, with entries from the ring $\mathbb{F}[X]$. When we talk about linear independence of $\{Z_i\}_{i=1}^k$, the underlying field is $\mathbb{F}(X)$. We recall the following well known lemma about non-zero determinant of a square matrix and the linear independence of it's columns.

Lemma 9.5. *Let M be a $n \times n$ over a field $\mathbb{F}$. Then $\mathsf{det}(M) \neq 0$ if and only if the columns of M are linearly independent over $\mathbb{F}$.*

The next lemma will prove the reverse direction of Theorem 9.4.

Lemma 9.6. *If $P_1(X), \ldots, P_k(X)$ are linearly dependent over $\mathbb{F}$, then $\mathsf{det}(W_\alpha) \equiv 0$.*

Proof. Since $P_1(X), \ldots, P_k(X)$ are linearly dependent over $\mathbb{F}$, there exist $\lambda_1, \ldots, \lambda_k \in \mathbb{F}$ (not all equal to zero) such that $\sum_{i=1}^k \lambda_i P_i(X) = 0$. Therefore, for all $d \in \{0, 1, \ldots, k-1\}$ we have that $\sum_{i=1}^k \lambda_i P_i(\alpha^d X) = 0$. This implies that $\sum_{i=1}^k \lambda_i Z_i = 0$. That is, the columns of W_α are linearly dependent over $\mathbb{F} \subseteq \mathbb{F}(X)$. Therefore by Lemma 9.5 the polynomial $\mathsf{det}(W_\alpha)$ is identically zero. That is, $\mathsf{det}(W_\alpha) \equiv 0$. This completes the proof. □

The next lemma will be used in the forward direction of the proof.

Lemma 9.7. *Let $A(X)$ and $B(X)$ be non zero polynomials in $\mathbb{F}[X]$ of degree at most ℓ. Let $\beta \in \mathbb{F}$ be an element of order $> \ell$. If $A(X)B(\beta X) - A(\beta X)B(X) \equiv 0$ then $A(X) = \lambda B(X)$ where $0 \neq \lambda \in \mathbb{F}$.*

Proof. Let $A(X) = \sum_{i=0}^{\ell} a_i X^i$, and $B(X) = \sum_{j=0}^{\ell} b_j X^j$ where $a_i, b_j \in \mathbb{F}$.

Case 1: We first assume that $b_0 \neq 0$. Later we will show how all other cases reduce to this. Let

$$S_{A,B}(X) = A(X)B(\beta X) - A(\beta X)B(X).$$

Since $S_{A,B}(X) \equiv 0$ we have that for all $t \in \{0, 1, \ldots, 2l\}$, the coefficients of X^t in $S_{A,B}(X)$ is zero. Our proof is based on the following claim.

Claim 1. *For all $i \in \{1, \ldots l\}$, either $\frac{a_i}{b_i} = \frac{a_0}{b_0}$, or $a_i = b_i = 0$.*

Proof. We prove the claim using induction on i. For $i = 1$, consider the coefficient of X in $S_{A,B}(X)$. The coefficient of X in $S_{A,B}(X)$ is $(\beta - 1)(a_0b_1 - a_1b_0)$. Since the order of β is $> \ell$, $(\beta - 1) \neq 0$. This implies $(a_0b_1 - a_1b_0) = 0$. So if $b_1 = 0$, then $a_1 = 0$ (because $b_0 \neq 0$) and if $b_1 \neq 0$, then $\frac{a_0}{b_0} = \frac{a_1}{b_1}$. Thus we assume that $i \geq 2$ and by induction hypothesis the claim holds for $j \in \{1, \ldots, i-1\}$. Now, consider the coefficients of X^i in $S_{A,B}(X)$. The coefficient of X^i in $S_{A,B}(X)$ is

$$\beta^i(a_0b_i - a_ib_0) + \beta^{i-1}(a_1b_{i-1} - a_{i-1}b_1) + \ldots + (a_ib_0 - a_0b_i).$$

By our assumption we know that

$$\beta^i(a_0b_i - a_ib_0) + \beta^{i-1}(a_1b_{i-1} - a_{i-1}b_1) + \ldots + (a_ib_0 - a_0b_i) = 0.$$

Consider the term $\beta^j(a_{i-j}b_j - a_jb_{i-j})$ for $0 < j < i$. By induction hypothesis, one of the following statement is true.

- $a_j = b_j = 0$
- $a_{i-j} = b_{i-j} = 0$
- $\frac{a_{i-j}}{b_{i-j}} = \frac{a_0}{b_0} = \frac{a_j}{b_j}$

In all the three cases the term $\beta^j(a_{i-j}b_j - a_jb_{i-j})$ is zero. Therefore the coefficient of X^i simplifies to, $(\beta^i - 1)(a_0b_i - a_ib_0)$ and we get $(\beta^i - 1)(a_0b_i - a_ib_0) = 0$. Since the order of β is strictly larger than ℓ, $(\beta^i - 1) \neq 0$. This implies $(a_0b_i - a_ib_0) = 0$. So if $b_i = 0$, then $a_i = 0$ (because $b_0 \neq 0$) and if $b_i \neq 0$, then $\frac{a_0}{b_0} = \frac{a_i}{b_i}$. This concludes the claim. □

Let $\lambda = \frac{a_0}{b_0} \in \mathbb{F}$. Thus $a_i = \lambda b_i$. Therefore $A(X) = \lambda B(X)$. Since $A(X) \not\equiv 0$, $\lambda \neq 0$.

Case 2: Suppose $b_0 = 0$ and $a_0 \neq 0$. Then let $A_1(X) = B(X)$ and $B_1(X) = A(X)$. Since $A(X)B(\beta X) - A(\beta X)B(X) \equiv 0$, we have that

$$\begin{aligned} & B_1(X)A_1(\beta X) - B_1(\beta X)A_1(X) && \equiv 0 \\ \implies \quad & -(B_1(\beta X)A_1(X) - B_1(X)A_1(\beta X)) && \equiv 0. \end{aligned}$$

Thus $A_1(X)B_1(\beta X) - A_1(\beta X)B_1(X) \equiv 0$. So by applying **Case 1** with $A_1(X)$ and $B_1(X)$, we get $A_1(X) = \lambda B_1(X)$ where $0 \neq \lambda \in \mathbb{F}$. This implies that $A(X) = \lambda^{-1}B(X)$ where $0 \neq \lambda^{-1} \in \mathbb{F}$.

Case 3: Suppose $b_0 = 0$ and $a_0 = 0$. Let r be the least integer such that either $a_r \neq 0$ or $b_r \neq 0$. Then let $A(X) = X^r A_2(X)$ and $B(X) = X^r B_2(X)$. Since $A(X)B(\beta X) - A(\beta X)B(X) \equiv 0$, we have that $A_2(X)B_2(\beta X) - A_2(\beta X)B_2(X) \equiv 0$. Note that the coefficient of X^0 is non zero in at least one of the polynomials $A_2(X)$ or $B_2(X)$. Furthermore, $A_2(X), B_2(X) \not\equiv 0$. Thus, if the coefficient of $B_2(X)$ is non-zero then we are in **Case 1** else we are in **Case 2**. This completes the proof of the lemma. □

The next lemma will be useful in showing the forward direction of Theorem 9.4.

Lemma 9.8. *Let $P_1(X), \ldots, P_k(X)$ be a set of polynomials from $\mathbb{F}[X]^{<n}$ and α be an element of order $> (n-1)(k-1)$. If $\mathsf{det}(W_\alpha) \equiv 0$, then $P_1(X), \ldots, P_k(X)$ are linearly dependent over $\mathbb{F}$.*

Proof. We prove the lemma using induction on k – the number of polynomials. For $k = 1$, $W_\alpha = [P_1(X)]$ and the lemma vacuously holds. From now on we assume that $k \geq 2$ and that the lemma holds for all $t < k$. Recall that Z_i denotes the i-th column of the matrix W_α. We first give the proof for the case when there is a subset of columns of $Z_1, \ldots, Z_k$, of size $< k$ that are linearly dependent over $\mathbb{F}(X)$. Let $\{i_1, \ldots, i_t\} \subset \{1, \ldots, k\}$ of size at most $k-1$, such that $Z_{i_1}, \ldots, Z_{i_t}$ are linearly dependent. Thus, there exists $\delta_1(X), \ldots, \delta_t(X) \in \mathbb{F}(X)$, not all equal to zero polynomial in $\mathbb{F}(X)$, such that $\sum_{j=1}^{t} \delta_i(X) Z_{i_j} \equiv 0$. Let $Z'_{i_1}, \ldots, Z'_{i_t}$ denote the restriction of $Z_{i_1}, \ldots, Z_{i_t}$ to first t rows of W_α. Since $\sum_{j=1}^{t} \delta_i(X) Z_{i_j} \equiv 0$, this implies that $\sum_{j=1}^{t} \delta_i(X) Z'_{i_j} \equiv 0$. Therefore $Z'_{i_1}, \ldots, Z'_{i_t}$ are also linearly dependent over

$\mathbb{F}(X)$. Consider $W_\alpha(P_{i_1}, \ldots, P_{i_t})$. This is a $t \times t$ matrix with columns $Z'_{i_1}, \ldots, Z'_{i_t}$. Since $Z'_{i_1}, \ldots, Z'_{i_t}$ are linearly dependent, by Lemma 9.5, $\mathsf{det}(W_\alpha(P_{i_1}, \ldots, P_{i_t})) \equiv 0$. By induction hypothesis, this implies that $P_{i_1}(X), \ldots, P_{i_t}(X)$ are linearly dependent over $\mathbb{F}$ and thus $P_1(X), \ldots, P_k(X)$ are linearly dependent over $\mathbb{F}$. So from now on we assume that for any subset $\{i_1, \ldots, i_t\} \subset \{1, \ldots, k\}$ of size at most $k-1$, $Z_{i_1}, \ldots, Z_{i_t}$ are linearly independent and $\mathsf{det}(W_\alpha(P_{i_1}, \ldots, P_{i_t})) \not\equiv 0$.

Next we prove the claim that if $\{Z_i\}_{i=1}^k$ are linearly dependent then we can choose the polynomials $\delta_i(X)$ in $\mathbb{F}(X)$ which satisfy certain desirable properties.

Claim 1. *Let* $\mathsf{det}(W_\alpha(P_1, \ldots, P_k) \equiv 0$. *Then there exist non-zero polynomials* $\delta_1(X), \ldots, \delta_k(X) \in \mathbb{F}[X]$, *of degree at most* $(n-1)(k-1)$ *such that* $\sum_{i=1}^k \delta_i(X) Z_i = 0$.

Proof. For all $i \in \{1, \ldots, k\}$, define

$$\begin{aligned} \delta_i(X) &= (-1)^{1+i} \mathsf{det}(W_\alpha(P_1(\alpha X), \ldots, P_{i-1}(\alpha X), P_{i+1}(\alpha X), \ldots, P_k(\alpha X))) \\ &= (-1)^{1+i} \mathsf{det}(W_\alpha(P_1, \ldots, P_k)[\{2, \ldots, k\}, \{1, \ldots, k\} \setminus \{i\}]). \end{aligned}$$

Observe that

$$\sum_{i=1}^k \delta_i(X) P_i(X) = \mathsf{det}(W_\alpha(P_1, \ldots, P_k)) \equiv 0,$$

because by assumption $\mathsf{det}(W_\alpha(P_1, \ldots, P_k)) \equiv 0$. Now consider the matrix W_j obtained by replacing the first row of $W_\alpha(P_1, \ldots, P_k)$ with j^{th} row of $W_\alpha(P_1, \ldots, P_k)$. That is,

$$W_j = \begin{pmatrix} P_1(\alpha^{j-1}X) & P_2(\alpha^{j-1}X) & \ldots & P_k(\alpha^{j-1}X) \\ P_1(\alpha X) & P_2(\alpha X) & \ldots & P_k(\alpha X) \\ P_1(\alpha^2 X) & P_2(\alpha^2 X) & \ldots & P_k(\alpha^2 X) \\ \vdots & \vdots & \ddots & \vdots \\ P_1(\alpha^{k-1}X) & p_2(\alpha^{k-1}X) & \ldots & P_k(\alpha^{k-1}X) \end{pmatrix}_{k \times k}$$

Note that for any $j \in \{2, \ldots, k\}$, $\sum_{i=1}^k \delta_i(X) P_i(\alpha^{j-1}X) = \mathsf{det}(W_j) \equiv 0$ (as W_j has two identical rows). Hence, $\sum_{i=1}^k \delta_i(X) Z_i = 0$. Since we are in the case where for any subset $\{i_1, \ldots, i_t\} \subset \{1, \ldots, k\}$ of size at most $k-1$, $Z_{i_1}, \ldots, Z_{i_t}$ are linearly independent we have

that

$$\mathsf{det}(W_\alpha(P_1,\ldots,P_{i-1},P_{i+1},\ldots,P_k)) \not\equiv 0.$$

This implies that

$$\delta_i(X) = (-1)^{1+i}\mathsf{det}(W_\alpha(P_1(\alpha X),\ldots,P_{i-1}(\alpha X),P_{i+1}(\alpha X),\ldots,P_k(\alpha X))) \not\equiv 0$$

and the degree of $\delta_i(X)$ is at most $(n-1)(k-1)$ for all $i \in \{1,\ldots,k\}$. This completes the proof of the claim. □

From now on we work with the collection $\{\delta_i(X)\}_{i=1}^k$ provided by Claim 1. We have the following expression.

$$\sum_{i=1}^{k} \delta_i(X)Z_i = \sum_{i=1}^{k} \delta_i(X) \begin{bmatrix} P_i(X) \\ P_i(\alpha X) \\ \ldots \\ P_i(\alpha^{k-1}X) \end{bmatrix} = 0$$

This implies that for each $j \in \{0,\ldots,k-1\}$, we have

$$\sum_{i=1}^{k} \delta_i(X)P_i(\alpha^j X) = 0. \tag{9.1}$$

By rearranging Equation 9.1 and absorbing the negative sign into $\delta_k(X)$ we get

$$\sum_{i=1}^{k-1} \delta_i(X)P_i(\alpha^j X) = \delta_k(X)P_k(\alpha^j X). \tag{9.2}$$

Substitute αX for X in Equation 9.2 for all $j \in \{0,\ldots,k-2\}$ we get

$$\sum_{i=1}^{k-1} \delta_i(\alpha X)P_i(\alpha^{j+1} X) = \delta_k(\alpha X)P_k(\alpha^{j+1} X). \tag{9.3}$$

Substitute the value of $P_k(\alpha^{j+1}X)$ from Equation 9.2 into Equation 9.3, we get that for all $j \in \{0, 1, \ldots, k-2\}$

$$\delta_k(X) \sum_{i=1}^{k-1} \delta_i(\alpha X) P_i(\alpha^{j+1}X) = \delta_k(\alpha X) \sum_{i=1}^{k-1} \delta_i(X) P_i(\alpha^{j+1}X). \quad (9.4)$$

Thus for each $j \in \{0, 1, \ldots, k-2\}$ we have

$$\sum_{i=1}^{k-1} \Big\{ \delta_k(X)\delta_i(\alpha X) - \delta_k(\alpha X)\delta_i(X) \Big\} P_i(\alpha^{j+1}X) = 0. \quad (9.5)$$

Substitute βX for X in Equation 9.5, where $\beta = \alpha^{-1}$. Then for all $j \in \{0, 1, \ldots, k-2\}$

$$\sum_{i=1}^{k-1} \Big\{ \delta_k(\beta X)\delta_i(X) - \delta_k(X)\delta_i(\beta X) \Big\} P_i(\alpha^{j}X) = 0. \quad (9.6)$$

Let Z'_i be the column vector corresponding to Z_i in the matrix $W_\alpha(P_1, \ldots, P_k)$ restricted to the first $k-1$ rows. Then from Equation 9.6 we get that for all $j \in \{0, 1, \ldots, k-2\}$

$$\sum_{i=1}^{k-1} \Big\{ \delta_k(\beta X)\delta_i(X) - \delta_k(X)\delta_i(\beta X) \Big\} \begin{bmatrix} P_i(X) \\ P_i(\alpha X) \\ \cdots \\ P_i(\alpha^{k-2}X) \end{bmatrix} = 0. \quad (9.7)$$

which implies that

$$\sum_{i=1}^{k-1} \Big\{ \delta_k(\beta X)\delta_i(X) - \delta_k(X)\delta_i(\beta X) \Big\} Z'_i = 0. \quad (9.8)$$

Consider the $(k-1) \times (k-1)$ matrix $[Z'_1, Z'_2, \ldots, Z'_{k-1}] = W_\alpha(P_1(X), P_2(X), \ldots, P_{k-1}(X))$. By the case of proof we are currently dealing, we have that $W_\alpha(P_1(X), P_2(X), \ldots, P_{k-1}(X))$ has a non-zero determinant. In other words, the vectors $Z'_1, \ldots, Z'_{k-1}$ are linearly independent over $\mathbb{F}(X)$. This implies that for all $i \in \{1, \ldots, k-1\}$,

$$\delta_i(X)\delta_k(\beta X) - \delta_i(\beta X)\delta_k(X) = 0$$

Observe that $\delta_i(X)$, $i \in \{1, \ldots, k\}$, are non-zero polynomials in $\mathbb{F}[X]$ of degree at most $(n-1)(k-1)$. Furthermore, the order of β is $> (n-1)(k-1)$, and thus by applying Lemma 9.7 we get that $\delta_i(X) = \lambda_i \delta_k(X)$ for all $i \in \{1, \ldots, k-1\}$ where $0 \neq \lambda_i \in \mathbb{F}$. By simplifying we get the following.

$$
\begin{aligned}
& \sum_{i=1}^{k} \delta_i(X) Z_i && = 0 && \\
\iff & \sum_{i=1}^{k} \lambda_i \delta_k(X) Z_i && = 0 && \text{(because } \delta_i(X) = \lambda_i \delta_k(X)\text{)} \\
\iff & \sum_{i=1}^{k} \lambda_i Z_i && = 0 && \text{(because } \delta_k(X) \not\equiv 0\text{)} \\
\iff & \sum_{i=1}^{k} \lambda_i P_i(X) && = 0 &&
\end{aligned}
$$

Hence $P_1(X), \ldots P_k(X)$ are linearly dependent over $\mathbb{F}$. This completes the proof of the lemma. □

Combining Lemmata 9.6 and 9.8, we get the proof of Theorem 9.4. We can get the following corollary from Theorems 9.3 and 9.4.

Corollary 9.9. *Let $\mathbb{F}$ be a field of size $> n$, α be either a primitive element or an element of $\mathbb{F}$ of order $> (n-1)(k-1)$ and let $P_1(X), \ldots, P_k(X)$ be a set of polynomials from $\mathbb{F}[X]^{<n}$. Then $P_1(X), \ldots, P_k(X)$ are linearly independent over $\mathbb{F}$ if and only if the α-folded Wronskian determinant $\det(W_\alpha(P_1, \ldots, P_k)) \not\equiv 0$ in $\mathbb{F}[X]$.*

Finding irreducible polynomials and elements of large order

Whenever we will need to use folded Wronskians, we will also need to get hold of a primitive element or an element of large order of an appropriate field. We start by reviewing some known algorithms for finding irreducible polynomials over finite fields. Note that for a finite field of order p^ℓ, the field operations can be done in time $(l \log p)^{\mathcal{O}(1)}$. And for an infinite field, the field operations will require $(\log N)^{\mathcal{O}(1)}$ where N is the size of the largest value handled by the algorithm. Typically we will provide an upper bound on N when the need arises. A result by Shoup [Sho88, Theorem 4.1]) allows us to find an irreducible

polynomial of degree r over $\mathbb{F}_{p^\ell}$ in time polynomial in p, l and d. Adleman and Lenstra [AJ86, Theorem 2] gave an algorithm that allows us to compute an irreducible polynomial of degree at least r over a prime field $\mathbb{F}_p$ in time polynomial in $\log p$ and r.

Lemma 9.10 ([AJ86, Sho88])**.** (Finding Irreducible Polynomials)

1. *There is an algorithm that given prime p and r, it can compute an irreducible polynomial $f(X) \in \mathbb{F}_p[X]$ such that $r \leq \deg(f) \leq cr \log p$ in $(cr(\log p)^2)^c$ time, where c is a constant.*

2. *For $q = p^\ell$ and an integer r, we can compute an irreducible polynomial of $\mathbb{F}_q[X]$ of degree r in $\mathcal{O}\big(\sqrt{p}(\log p)^3 r^3 (\log r)^{\mathcal{O}(1)} + (\log p)^2 r^4 (\log r)^{\mathcal{O}(1)} + (\log p) r^4 (\log r)^{\mathcal{O}(1)} \ell^2 (\log \ell)^{\mathcal{O}(1)}\big)$ time.*

Next we consider a few algorithms for finding primitive elements in finite fields. For fields of large order but small characteristic, we have the following lemma, which follows from the results of Shparlinski [Shp90] and also from the results of Shoup [Vic90].

Lemma 9.11 ([Shp90, Vic90])**.** *Let $\mathbb{F} = \mathbb{F}_{p^\ell}$ be a finite field. Then we can compute a set $S \subset \mathbb{F}$, containing a primitive element, of size $\mathrm{poly}(p, \ell)$ in time $\mathrm{poly}(p, \ell)$.*

We use Lemma 9.11 to get the following result that allows us to find elements of sufficiently large order in a finite field or a primitive element in a field of small size.

Lemma 9.12. *Let $\mathbb{F} = \mathbb{F}_{p^\ell}$ be a finite field. Given a number n such that $n < p^\ell$, we can compute an element of $\mathbb{F}$ of order greater than n in $\mathrm{poly}(p, \ell, n)$ time. Furthermore, we can find a primitive element in $\mathbb{F}$ in time $|\mathbb{F}|^{\mathcal{O}(1)}$.*

Proof. We begin by applying Lemma 9.11 to the field $\mathbb{F}$ and obtain a set S of size $\mathrm{poly}(p, \ell)$. This takes time $\mathrm{poly}(p, \ell)$. Then for each element $\alpha \in S$ we compute the set $G_\alpha = \{\alpha^i \mid i = 1, 2, \ldots, n+1\}$. If for any α, $|G_\alpha| = n + 1$, then we return this as the required element of order greater than n. Since the set S contains at least one primitive element of $\mathbb{F}$, we will find some α in this step. Note this this step too takes $\mathrm{poly}(p, \ell, n)$ time.

To prove the second statement of the lemma do as follows. For each element $\alpha \in S$, consider the set $S(\alpha) = \{\alpha^i \mid i = 1, \ldots, p^\ell\}$. If $|S(\alpha)| = |\mathbb{F}^*| = p^\ell - 1$, then α generates $\mathbb{F}^*$. Since the set S contains at least one primitive element of $\mathbb{F}$, we will find some α in this step. Note this this step can be done in time $|\mathbb{F}|^{\mathcal{O}(1)}$. This completes the proof of this lemma. □

When given a small field, the following lemma allows us to increase the size of the field as well as find a primitive element in the larger field.

Lemma 9.13. *Given a field $\mathbb{F} = \mathbb{F}_{p^\ell}$ and a number n such that $p^\ell < n$, we can find an extension $\mathbb{K}$ of $\mathbb{F}$ such that $n < |\mathbb{K}| < n^2$ and a primitive element α of $\mathbb{K}$ in time $n^{\mathcal{O}(1)}$.*

Proof. Let r be smallest number such that $p^{\ell r} > n$. But then $p^{\ell r/2} < n$. Therefore we have that $p^{\ell r} < n^2$. Next we find an extension of $\mathbb{F}$ of degree r, by finding an irreducible polynomial over $\mathbb{F}$ of degree r using Lemma 9.10 in time polynomial in p, ℓ, r, which is $n^{\mathcal{O}(1)}$. Then we can use Lemma 9.12 to compute a primitive element of $\mathbb{K}$. Since $|\mathbb{K}| < n^2$, this can be done in time $n^{\mathcal{O}(1)}$. This completes the proof of this lemma. □

9.3.2 Deterministic Truncation of Matrices

In this section we look at algorithms for computing k-truncation of matrices. We consider matrices over the set of rational numbers $\mathbb{Q}$ or over some finite field $\mathbb{F}$. Therefore, we are given as input a matrix M of rank n over a field $\mathbb{F}$. Let p be the characteristic of the field $\mathbb{F}$ and N denote the size of the input in bits. The following theorem, gives us an algorithm to compute the truncation of a matrix using the classical wronskian, over an appropriate field. We shall refer to this as the *classical wronskian method of truncation.*

Lemma 9.14. *Let M be a $n \times m$ matrix of rank n over a field $\mathbb{F}$, where $\mathbb{F}$ is either $\mathbb{Q}$ or $\mathrm{char}(\mathbb{F}) > n$. Then we can compute a $k \times m$ matrix M_k of rank k over the field $\mathbb{F}(X)$ which is a k-truncation of the matrix M in $\mathcal{O}(mnk)$ field operations.*

Proof. Let $\mathbb{F}[X]$ be the ring of polynomials in X over $\mathbb{F}$ and let $\mathbb{F}(X)$ be the corresponding field of fractions. Let $C_1, \ldots, C_m$ denote the columns of M. Observe that we have a

polynomial $P_i(X)$ corresponding to the column C_i of degree at most $n-1$, and by Lemma 9.4 we have that $C_{i_1}, \ldots, C_{i_\ell}$ are linearly independent over $\mathbb{F}$ if and only if $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ are linearly independent over $\mathbb{F}$. Further note that P_i lies in $\mathbb{F}[X]$ and thus also in $\mathbb{F}(X)$. Let D_i be the vector $(P_i(X), P_i^{(1)}(X), \ldots, P_i^{(k-1)}(X))$ of length k with entries from $\mathbb{F}[X]$ (and also in $\mathbb{F}(X)$). Note that the entries of D_i are polynomials of degree at most $n-1$. Let us define the matrix M_k to be the $(k \times m)$ matrix whose columns are D_i^T, and note that M_k is a matrix with entries from $\mathbb{F}[X]$. We will show that indeed M_k is a desired k-truncation of the matrix M .

Let $I \subseteq \{1, \ldots, m\}$ such that $|I| = \ell \leq k$. Let $C_{i_1}, \ldots, C_{i_\ell}$ be a linearly independent set of columns of the matrix M over $\mathbb{F}$, where $I = \{i_1, \ldots, i_\ell\}$. We will show that the columns $D_{i_1}^T, \ldots, D_{i_\ell}^T$ are linearly independent in M_k over $\mathbb{F}(X)$. Consider the $k \times \ell$ matrix M_I whose column are the vectors $D_{i_1}^T, \ldots, D_{i_\ell}^T$. We shall show that M_I has rank ℓ by showing that there is a $\ell \times \ell$ submatrix whose determinant is a non-zero polynomial. Let $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ be the polynomials corresponding to the vectors $C_{i_1}, \ldots, C_{i_\ell}$. By Lemma 9.4 we have that $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ are linearly independent over $\mathbb{F}$. Then by Theorem 9.2, the $(\ell \times \ell)$ matrix formed by the column vectors $(P_{i_j}(X), P_{i_j}^{(1)}(X), \ldots, P_{i_j}^{(\ell-1)}(X))^T$, $i_j \in I$, is a non-zero determinant in $\mathbb{F}[X]$. But note that this matrix is a submatrix of M_I. Therefore M_I has rank ℓ in $\mathbb{F}(X)$. Therefore the vectors $D_{i_1}^T, \ldots, D_{i_\ell}^T$ are linearly independent in $\mathbb{F}(X)$. This completes the proof of the forward direction.

Let $I \subseteq \{1, \ldots, m\}$ such that $|I| = \ell \leq k$ and let $D_{i_1}^T, \ldots, D_{i_\ell}^T$ be linearly independent in M_k over $\mathbb{F}(X)$, where $I = \{i_1, \ldots, i_\ell\}$. We will show that the corresponding set of columns $C_{i_1}, \ldots, C_{i_\ell}$ are also linearly independent over $\mathbb{F}$. For a contradiction assume that $C_{i_1}, \ldots, C_{i_\ell}$ are linearly *dependent* over $\mathbb{F}$. Let $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ be the polynomials in $\mathbb{F}[X]$ corresponding to these vectors. Then by Lemma 9.4 we have that $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ are linearly dependent over $\mathbb{F}$. So there is a tuple $a_{i_1}, \ldots, a_{i_\ell}$ of values of $\mathbb{F}$ such that $\sum_{j=1}^{\ell} a_{i_j} P_{i_j}(X) = 0$. Therefore, for any $d \in \{1, \ldots, \ell-1\}$, we have that $\sum_{j=1}^{\ell} a_{i_j} P_{i_j}^{(d)}(X) = 0$. Now let $D_{i_1}^T, \ldots, D_{i_\ell}^T$ be the column vectors of M_k corresponding to $C_{i_1}, \ldots, C_{i_\ell}$. Note that $\mathbb{F}$ is a subfield of $\mathbb{F}(X)$ and by the above, we have that $\sum_{j=1}^{\ell} a_{i_j} D_{i_j} = 0$. Thus $D_{i_1}^T, \ldots, D_{i_\ell}^T$ are linearly dependent in M_k over $\mathbb{F}(X)$, a contradiction to our assumption.

Thus we have shown that for any $\{i_1, \ldots, i_\ell\} \subseteq \{1, \ldots, m\}$ such that $\ell \leq k$, $C_{i_1}, \ldots, C_{i_\ell}$ are linearly independent over $\mathbb{F}$ if and only if $D_{i_1}, \ldots, D_{i_\ell}$ are linearly independent over $\mathbb{F}(X)$. To estimate the running time, observe that for each C_i we can compute D_i in $\mathcal{O}(kn)$ field operations and thus we can compute M_k in $\mathcal{O}(mnk)$ field operations. This completes the proof of this lemma. □

Lemma 9.14 is useful in obtaining k-truncation of matrices which entries are either from the filed of large characteristic or from $\mathbb{Q}$. The following lemma allows us to find truncations in fields of small characteristic which have large order. We however require a primitive element or an element of high order of such a field to compute the truncation. In the next lemma we demand a lower bound on the size of the field as we need an element of certain order. We will later see how to remove this requirement from the statement of the next lemma.

Lemma 9.15. *Let $\mathbb{F}$ be a finite field and α be an element of $\mathbb{F}$ of order at least $(n-1)(k-1)+1$. Let M be a $(n \times m)$ matrix of rank n over a field $\mathbb{F}$. Then we can compute a $(k \times m)$ matrix M_k of rank k over the field $\mathbb{F}(X)$ which is a k-truncation of the matrix M in $\mathcal{O}(mnk)$ field operations.*

Proof. Let $\mathbb{F}[X]$ be the ring of polynomials in X over $\mathbb{F}$ and let $\mathbb{F}(X)$ be the corresponding field of fractions. Let $C_1, \ldots, C_m$ denote the columns of M. Observe that we have a polynomial $P_i(X)$ corresponding to the column C_i of degree at most $n-1$, and by Lemma 9.4 we have that $C_{i_1}, \ldots, C_{i_\ell}$ are linearly independent over $\mathbb{F}$ if and only if $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ are linearly independent over $\mathbb{F}$. Further note that $P_i(X)$ lies in $\mathbb{F}[X]$ (and also in $\mathbb{F}(X)$).

Let D_i be the vector $(P_i(X), P_i(\alpha X), \ldots, P_i(\alpha^{k-1}X))$. Observe that the entries of D_i are polynomials of degree at most $n-1$ and are elements of $\mathbb{F}[X]$. Let us define the matrix M_k to be the $(k \times m)$ matrix whose columns are the vectors D_i^T, and note that M_k is a matrix with entries from $\mathbb{F}[X] \subseteq \mathbb{F}(X)$. We will show that M_k is a desired k-truncation of the matrix M .

Let $I \subseteq \{1, \ldots, m\}$ such that $|I| = \ell \leq k$. Let $C_{i_1}, \ldots, C_{i_\ell}$ be a linearly independent set of columns of the matrix M over $\mathbb{F}$, where $I = \{i_1, \ldots, i_\ell\}$. We will show that $D_{i_1}^T, \ldots, D_{i_\ell}^T$

are linearly independent in M_k over $\mathbb{F}(X)$. Consider the $k \times \ell$ matrix M_I whose columns are the vectors $D_{i_1}^T, \ldots, D_{i_\ell}^T$. We shall show that M_I has rank ℓ by showing that there is a $\ell \times \ell$ submatrix whose determinant is a non-zero polynomial. Let $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ be the polynomials corresponding to the vectors $C_{i_1}, \ldots, C_{i_\ell}$. By Lemma 9.4 we have that $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ are linearly independent over $\mathbb{F}$. Then by Theorem 9.4, the $(\ell \times \ell)$ matrix formed by the column vectors $(P_{i_j}(X), P_{i_j}(\alpha X), \ldots, P_{i_j}(\alpha^{(l-1)} X))^T$, $i_j \in I$, is a non-zero determinant in $\mathbb{F}[X]$. But note that this matrix is a submatrix of M_I. Therefore M_I has rank ℓ in $\mathbb{F}(X)$. Therefore the vectors $D_{i_1}, \ldots, D_{i_\ell}$ are linearly independent in $\mathbb{F}(X)$. This completes the proof of the forward direction.

Let $I \subseteq \{1, \ldots, m\}$ such that $|I| = \ell \leq k$ and let $D_{i_1}^T, \ldots, D_{i_\ell}^T$ be linearly independent in M_k over $\mathbb{F}(X)$, where $I = \{i_1, \ldots, i_\ell\}$. We will show that the corresponding set of columns $C_{i_1}, \ldots, C_{i_\ell}$ are also linearly independent over $\mathbb{F}$. For a contradiction assume that $C_{i_1}, \ldots, C_{i_\ell}$ are linearly *dependent* over $\mathbb{F}$. Let $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ be the polynomials in $\mathbb{F}[X]$ corresponding to these vectors. Then by Lemma 9.4 we have that $P_{i_1}(X), \ldots, P_{i_\ell}(X)$ are linearly dependent over $\mathbb{F}$ So there is a tuple $a_{i_1}, \ldots, a_{i_\ell}$ of values of $\mathbb{F}$ such that $\sum_{j=1}^{\ell} a_{i_j} P_{i_j}(X) = 0$. Therefore, for any $d \in \{1, \ldots, \ell - 1\}$, we have that $\sum_{j=1}^{\ell} a_{i_j} P_{i_j}(\alpha^d X) = 0$. Now let $D_{i_1}^T, \ldots, D_{i_\ell}^T$ be the column vectors of M_k corresponding to $C_{i_1}, \ldots, C_{i_\ell}$. Note that $\mathbb{F}$ is a subfield of $\mathbb{F}(X)$ and by the above, we have that $\sum_{j=1}^{\ell} a_{i_j} D_{i_j} = 0$. Thus $D_{i_1}^T, \ldots, D_{i_\ell}^T$ are linearly dependent in M_k over $\mathbb{F}(X)$, a contradiction to our assumption.

Thus we have shown that for any $\{i_1, \ldots, i_\ell\} \subseteq \{1, \ldots, m\}$ such that $\ell \leq k$, $C_{i_1}, \ldots, C_{i_\ell}$ are linearly independent over $\mathbb{F}$ if and only if $D_{i_1}, \ldots, D_{i_\ell}$ are linearly independent over $\mathbb{F}(X)$. To estimate the running time, observe that for each C_i we can compute D_i in $\mathcal{O}(kn)$ field operations and thus we can compute M_k in $\mathcal{O}(mnk)$ field operations. This completes the proof of this lemma. □

In Lemma 9.15 we require that α be an element of order at least $(n-1)(k-1)+1$. This implies that the order of the field $\mathbb{F}$ must be at least $(n-1)(k-1)+1$. We can ensure these requirements by preprocessing the input before invoking the Lemma 9.15. Formally, we show the following lemma.

Lemma 9.16. *Let M be a matrix of dimension $n \times m$ over a finite field $\mathbb{F}$, and of rank n. Let $\mathbb{F} = \mathbb{F}_{p^\ell}$ where $p < n$. Then in polynomial time we can find an extension field $\mathbb{K}$ of order at least $nk+1$ and an element α of $\mathbb{K}$ of order at least $nk+1$, such that M is a matrix over $\mathbb{K}$ with the same linear independence relationships between it's columns as before.*

Proof. We distinguish two cases by comparing the values of p^ℓ and n.

Case 1: $p^\ell \leq nk+1$**.** In this case we use Lemma 9.13 to obtain an extension $\mathbb{K}$ of $\mathbb{F}$ of size at most $(nk+1)^2$, and a primitive element α of $\mathbb{K}$ in polynomial time.

Case 2: $nk+1 < p^\ell$**.** In this case we set $\mathbb{K} = \mathbb{F}$ and and use Lemma 9.12 to find an element of order at least nk, in time $\text{poly}(p, l, nk)$.

Observe that $\mathbb{F}$ is a subfield of $\mathbb{K}$, M is also a matrix over $\mathbb{K}$. Thus, any set of linearly dependent columns over $\mathbb{F}$ continue to be linearly dependent over $\mathbb{K}$. Similarly, linearly independent columns continue to be linearly independent. This completes the proof of this lemma. □

Next we show a result that allows us to find basis of matrices with entries from $\mathbb{F}[X]$.

Lemma 9.17. *Let M be a $m \times t$ matrix with entries from $\mathbb{F}[X]^{<n}$ and let $m \leq t$. Let $w : \mathbf{C}(M) \to \mathbb{R}^+$ be a weight function. Then we can compute minimum weight column basis of M in $\mathcal{O}(m^2n^2t + m^\omega nt)$ field operations in $\mathbb{F}$*

Proof. Let $S \subseteq \mathbb{F}^*$ be a set of size $(n-1)m+1$ and for every $\alpha \in S$, let $M(\alpha)$ be the matrix obtained by substituting α for X in the polynomials in matrix M. Now we compute the minimum weight column basis $C(\alpha)$ in $M(\alpha)$ for all $\alpha \in S$. Let $\ell = \mathsf{max}\{|C(\alpha)| \mid \alpha \in S\}$. Among all the column basis of size ℓ, let $C(\zeta)$ be a minimum weighted column basis for some $\zeta \in S$. Let C' be the columns in M corresponding to $C(\zeta)$. We will prove that C' is a minimum weighted column basis of M. Towards this we start with the following claim.

Claim 1. *The rank of M is the maximum of the rank of matrices $M(\alpha)$, $\alpha \in S$.*

Proof. Let $r \leq m$ be the rank of M. Thus, we know that there exists a submatrix $W$of M of dimension $r \times r$ such that $\det(W)$ is a non-zero polynomial. The degree of the polynomial $\det(W(X)) \leq (n-1) \times r \leq (n-1)m$. Thus, we know that it has at-most $(n-1)m$ roots. Hence, when we evaluate $\det(W(X))$ on set S of size more than $(n-1)m$, there exists at least one element in S, say β, such that $\det(W(\beta)) \neq 0$. Thus, the rank of M is upper bounded by the rank of $M(\beta)$ and hence upper bounded by the maximum of the rank of matrices $M(\alpha)$, $\alpha \in S$.

As before let $r \leq \ell$ be the rank of M. Let α be an arbitrary element of S. Observe that for any submatrix Z of dimension $r' \times r'$, $r' > r$ we have that $\det(Z(X)) \equiv 0$. Thus, for any α, the determinant of the corresponding submatrix of $M(\alpha)$ is also 0. This implies that for any α, the rank of $M(\alpha)$ is at most r. This completes the proof. □

Claim 1 implies that $\ell = \mathsf{max}\{|C(\alpha)| \mid \alpha \in S\}$ is equal to the rank of M. Our next claim is following.

Claim 2. *For any $\alpha \in S$, and $C \subseteq \mathbf{C}(M(\alpha))$, if C is linearly independent in $M(\alpha)$ then C is also linearly independent in M.*

The proof follows from the arguments similar to the ones used in proving reverse direction of Claim 1. Let $r \leq m$ be the rank of M and let C^* be a minimum weight column basis of M. Thus, we know that there exists a submatrix W of M of dimension $r \times r$ such that $\det(W)$ is a non-zero polynomial. The degree of the polynomial $\det(W(X)) \leq (n-1) \times r \leq (n-1)m$. Thus, we know that it has at most $(n-1)r$ roots. Hence, when we evaluate $\det(W(X))$ on set S of size more than $(n-1)r$, there exists at least one element in S, say β, such that $\det(W(\beta)) \neq 0$ and the set of columns C^* is linearly independent in $M(\beta)$. Using Claim 2 and the fact that C^* is linearly independent in both $M(\beta)$ and M, we can conclude that C^* is a column basis for $M(\beta)$. Since $|C'| = |C^*|$, $w(C') \leq w(C^*)$, C' is indeed a minimum weighted column basis of M.

We can obtain any $M(\alpha)$ with at most $\mathcal{O}(nmt)$ field operations in $\mathbb{F}$. Furthermore, we can compute minimum weight column basis of $M(\alpha)$ in $\mathcal{O}(tm^{\omega-1})$ field operations [BCKN13]. Hence the total number of field operations over $\mathbb{F}$ is bounded by $\mathcal{O}(m^2n^2t + m^\omega nt)$. □

Finally, we combine Lemma 9.14, Lemma 9.16 and Lemma 9.15 to obtain the following theorem.

Theorem 9.5. *Let $\mathbb{F} = \mathbb{F}_{p^\ell}$ be a finite field or $\mathbb{F} = \mathbb{Q}$. Let M be a $n \times m$ matrix over $\mathbb{F}$ of rank n. Given a number $k \leq n$, we can compute a matrix M_k over the field $\mathbb{F}(X)$ such that it is a representation of the k-truncation of M. Furthermore, given M_k, we can test whether a given set of l columns in M_k are linearly independent in $\mathcal{O}(n^2k^3)$ field operations.*

Proof. We first consider the case when the characteristic of the field $\mathbb{F}$ is 0, or if the characteristic p of the finite field is strictly larger than n ($\text{char}(\mathbb{F}) > n$). In this case we apply Lemma 9.14 to obtain a matrix M_k over $\mathbb{F}(X)$ which is a k-truncation of M. We now consider the case when $\mathbb{F}$ is a finite field and $\text{char}(\mathbb{F}) < n$. First apply Lemma 9.16 to ensure that the order of the field $\mathbb{F}$ is greater than $(n-1)(k-1)+1$ and to obtain an element of order at least $(n-1)(k-1)+1$ in the field $\mathbb{F}$. Of course by doing this, we have gone to an extension of $\mathbb{K}$ of $\mathbb{F}$ of size at least $(n-1)(k-1)+1$. However, for brevity of presentation we will assume that the input is given over such an extension. We then apply Lemma 9.15 to obtain a matrix M_k over $\mathbb{F}(X)$ which is a representation of the k-truncation of the matrix M. One should notice that in fact This completes the description of M_k in all the cases.

Let $I \subseteq \{1, \ldots, m\}$ such that $|I| = \ell \leq k$. Let $D_{i_1}, \ldots, D_{i_\ell}$ be a set of columns of the matrix M_k over $\mathbb{F}$, where $I = \{i_1, \ldots, i_\ell\}$. Furthermore, by M_I we denote the $k \times \ell$ submatrix of M_k containing the columns $D_{i_1}, \ldots, D_{i_\ell}$. To test whether these columns are linearly independent, we can apply Lemma 9.17 on M_I^T and see the size of column basis of M_I^T is ℓ or not. This takes time $\mathcal{O}(\ell^2 n^2 k + \ell^\omega nk) = \mathcal{O}(n^2k^3)$ field operations in $\mathbb{F}$. □

Representing the truncation over a finite field

In Theorem 9.5, the representation M_k is over the field $\mathbb{F}(X)$. However, in some cases this matrix can also be viewed as a representation over a finite extension of $\mathbb{F}$ of sufficiently large degree. That is, if $\mathbb{F} = \mathbb{F}_{p^\ell}$ is a finite field then M_k can be given over $\mathbb{F}_{p^{\ell'}}$ where $\ell' \geq nk\ell$. Formally we have the following lemma.

Theorem 9.6. *Let M be a $n \times m$ matrix over $\mathbb{F}$ of rank n, $k \leq n$ be a positive integer and N be the size of the input matrix. If $\mathbb{F} = \mathbb{F}_p$ be a prime field or $\mathbb{F} = \mathbb{F}_{p^\ell}$ where $p = N^{\mathcal{O}(1)}$, then in polynomial time we can find a k-truncation M_k of M over a finite extension $\mathbb{K}$ of $\mathbb{F}$ where $\mathbb{K} = \mathbb{F}_{p^{nk\ell}}$.*

Proof. Let M_k be the matrix returned by Theorem 9.5. Next we show how we can view the entries in M_k over a finite extension of $\mathbb{F}$. Consider any extension $\mathbb{K}$ of $\mathbb{F}$ of degree $r \geq nk$. Thus $\mathbb{K} = \frac{\mathbb{F}[X]}{r(X)}$, where $r(X)$ is a irreducible polynomial in $\mathbb{F}[X]$ of degree r. Recall that each entry of M_k is a polynomial in $\mathbb{F}[X]$ of degree at most $n-1$ and therefore they are present in $\mathbb{K}$. Further the determinant of any $k \times k$ submatrix of M_k is identically zero in $\mathbb{K}$ if and only if it is identically zero in $\mathbb{F}(X)$. This follows from the fact that the determinant is a polynomial of degree at most $(n-1)k$ and therefore is also present in $\mathbb{K}$. Thus M_k is a representation over $\mathbb{K}$.

To specify the field $\mathbb{K}$ we need to compute the irreducible polynomial $r(X)$. If $\mathbb{F}$ is a prime field, i.e. $\mathbb{F} = \mathbb{F}_p$, then we can compute the polynomial $r(X)$ using the first part of Lemma 9.10. And if $p = N^{\mathcal{O}(1)}$ we can use the second part of Lemma 9.10 to compute $r(X)$. Thus we have a well defined k-truncation of M over the finite field $\mathbb{K} = \frac{\mathbb{F}[X]}{r(X)}$. Furthermore, if degree of $r(X)$ is nk then $\mathbb{K}$ is isomorphic to $\mathbb{F}_{p^{nk\ell}}$. This completes the proof of this theorem. □

From the above theorems we conclude the following.

Theorem 9.7. *Let M be a linear matroid of which is representable over a field $\mathbb{F}$. Then we can find a representation of the k-truncation of M, deterministically, in polynomially many field operations.*

9.3.3 Representation of the ℓ-elongation of a Matroid

The ℓ-elongation of a matroid M may be obtained from a truncation of the dual matroid M^* by the following observation.

Observation 9.18 ([Mur00], page 75)**.** *Let M be a matroid of rank n over a ground set of size m. Let M^* denote the dual of the matroid M and $E(M, \ell)$ denote the ℓ-elongation*

of M. Further let $T(M^*, m-\ell)$ denote the $(m-\ell)$ truncation of M^*. Then $E(M,\ell) = \{T(M^*, m-\ell)\}^*$, i.e. the ℓ-elongation of M is the dual of the $(m-\ell)$-truncation of the dual of M.

When M is a linear matroid, this observation leads to the following corollary.

Corollary 9.19. *Let M be a linear matroid of rank n, over a ground set of size m, which is representable over a field $\mathbb{F}$. Given a number $\ell \geq n$, we can compute a representation of the ℓ-elongation of M, over the field $\mathbb{F}(X)$ in $\mathcal{O}(mn\ell)$ field operations over $\mathbb{F}$.*

9.4 Application to Computation of Representative Families

In this section we give deterministic algorithms to compute representative families of a linear matroid, given its representation matrix. Let us recall the definition of a *q-representative family*.

Definition 9.8 (q-Representative Family). *Given a matroid $M = (E, \mathcal{I})$ and a family $\mathcal{S}$ of subsets of E, we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is* q-representative *for $\mathcal{S}$ if the following holds: for every set $Y \subseteq E$ of size at most q, if there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{X} \cup Y \in \mathcal{I}$. If $\hat{\mathcal{S}} \subseteq \mathcal{S}$ is q-representative for $\mathcal{S}$ we write $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$.*

Let $p + q = k$. Fomin et al. [FLS14, Theorem 3.1] first gave a deterministic algorithm for computing q-representative of a p-family of independent sets if the rank of the corresponding matroid is $p+q$. For matroids of higher rank they first computes the representation matrix of a k-truncation of $M = (E, \mathcal{I})$. This step is randomized and it returns representation of a k-truncation of $M = (E, \mathcal{I})$ with a high probability. Given this matrix, one applies [FLS14, Theorem 3.1] and arrive at Theorem 9.11. We state their result below.

Theorem 9.9 ([FLS14])**.** *Let $M = (E, \mathcal{I})$ be a linear matroid and let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a p-family of independent sets. Then there exists $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$ of size $\binom{p+q}{p}$. Furthermore, given a representation A_M of M over a field $\mathbb{F}$, there is a randomized algorithm computing $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$ in $\mathcal{O}\left(\binom{p+q}{p} tp^\omega + t\binom{p+q}{q}^{\omega-1}\right)$ operations over $\mathbb{F}$.*

In this section we design fast deterministic algorithm for computing q-representative even if the underlying linear matroid has unbounded rank, using deterministic truncation of linear matroids. A deterministic algorithm can be easily obtained from the above recipe by using our deterministic truncation algorithm in place of the randomized one. However observe that the representation given by Theorem 9.5 is over $\mathbb{F}(X)$. For the purpose of computing q-representative of a p-family of independent sets, we need to find a set of linearly independent columns over a matrix with entries from $\mathbb{F}[X]$. However, deterministic algorithms to compute basis of matrices over $\mathbb{F}[X]$ is not as fast as compared to the algorithms where we do not need to do symbolic computations. We shall see that there is a faster algorithm for computing a representative set, but of a slightly larger size.

Definition 9.10. *Let $W = \{v_1, \ldots, v_m\}$ be a set of vectors over $\mathbb{F}$ and $w : W \to \mathbb{R}^+$. We say that $S \subseteq W$ is a* spanning set*, if every $v \in W$ can be written as linear combination of vectors in S with coefficients from $\mathbb{F}$. We say that S is a* nice spanning set of W*, if S is a spanning set and for any $z \in W$ if $z = \sum_{v \in S} \lambda_v v$, and $0 \neq \lambda_v \in \mathbb{F}$ then we have $w(v) \leq w(z)$.*

The following lemma enables us to find a spanning set of vectors over $\mathbb{F}(X)$, of small size.

Lemma 9.20. *Let $\mathbb{F}$ be a field and let $M \in \mathbb{F}[X]^{m \times t}$ be a matrix over $\mathbb{F}[X]^{<n}$ and let $w : \mathbf{C}(M) \to \mathbb{R}^+$ be a weight function. Then we can find a nice spanning set S of $\mathbf{C}(M)$ of size at most nm with at most $\mathcal{O}(t(nm)^{\omega-1})$ field operations.*

Proof. The essential idea is to do a "gaussian elimination" in M, but only over the subfield $\mathbb{F}$ of $\mathbb{F}(X)$. Let C_i be a column of the matrix M. It is a vector of length m over $\mathbb{F}[X]^{<n}$ and it's entries are polynomials $P_{ji}(X)$, where $j \in \{1, \ldots, m\}$. Observe that $P_{ji}(X)$ is a polynomial of degree $n-1$ with coefficients from $\mathbb{F}$. Let v_{ji} denote the vector of length n corresponding to the polynomial $P_{ji}(X)$. Consider the column vector v_i formed by concatenating each v_{ji} in order from $j = 1$ to m. That is, $v_i = (v_{1i}, \ldots, v_{mi})^T$. This vector has length nm and has entries from $\mathbb{F}$. Let N be the matrix where columns correspond to column vectors v_i. Note that N is a matrix over $\mathbb{F}$ of dimension $nm \times t$ and the time taken to compute N is $\mathcal{O}(tnm)$. For each column v_i of N we define it's weight to be $w(C_i)$. We now do a gaussian elimination in N over $\mathbb{F}$ and compute a minimum weight set of

column vectors S', which spans N. Observe that $|S'| \leq nm$ and time taken to compute S' is $\mathcal{O}(t(nm)^{\omega-1})$ [BCKN13]. Let S be the set of column vectors in M corresponding to the column vectors in S'. We return S as a nice spanning set of column vectors in M.

Now we show the correctness of the above algorithm. We first show that S is a spanning set of M. Let $v_1, \ldots, v_{|S|}$ be the set of vectors in S and let v_d be some column vector in N. Then $v_d = \sum_{i=1}^{|S|} a_i v_i$ where $a_i \in \mathbb{F}$. In particular for any $j \in \{1, \ldots, m\}$ we have $v_{jd} = \sum_{i=1}^{|S|} a_i v_{ji}$. Let $C_1, \ldots, C_{|S|}$ be the column vectors corresponding to $v_1, \ldots, v_{|S|}$ and let C_d be the column vector corresponding to v_d. We claim that $C_d = \sum_{i=1}^{|S|} a_i C_i$. Consider the j-th entry of the column vector C and of $C_1, \ldots, C_{|S|}$. Towards our claim we need to show that $P_{jd}(X) = \sum_{i=1}^{|S|} a_i P_{ji}(X)$. But since v_{dj} and $\{v_{ij} \mid j \in \{1, \ldots, m\}\}$ are the collection of vectors corresponding to $P_{jd}(X)$ and $\{P_{ji}(X) \mid j \in \{1, \ldots, m\}\}$, the claim follows.

Next we show that S is indeed a nice spanning set. Since S is a spanning set of M we have that any column $C_d = \sum_{C_i \in S} \lambda_i C_i$, $\lambda_i \in \mathbb{F}$. Let $C_j \in S$ be such that $\lambda_j \neq 0$ and $w(C_j) > w(C_d)$. Let v_d and v_j be the vectors corresponding to C_d and C_j respectively. We have that $v_d = \sum_{v_i \in S} \lambda_i v_i$, which implies $v_j = \lambda_j^{-1} v_d - \sum_{v_i \in S, v_i \neq v_j} \lambda_j^{-1} \lambda_i v_i$. But this implies that $S^* = (S \setminus \{v_j\}) \cup \{v_d\}$ is a spanning set of N, and $w(S^*) < w(S)$, which is a contradiction. Thus we have that for every column vector $C \in M$ if $C = \sum_{C_i \in S} \lambda_i C_i$ and $0 \neq \lambda_i \in \mathbb{F}$, then $w(C_i) \leq w(C)$. This completes the proof. □

The main theorem of this section is as follows.

Theorem 9.11. *Let $M = (E, \mathcal{I})$ be a linear matroid of rank n and let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a p-family of independent sets. Let A be a $n \times |E|$ matrix representing M over a field $\mathbb{F}$, where $\mathbb{F} = \mathbb{F}_{p^\ell}$ or $\mathbb{F}$ is $\mathbb{Q}$. Then there are deterministic algorithms computing $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$ as follows.*

(i) A family $\widehat{\mathcal{S}}$ of size $\binom{p+q}{p}$ in $\mathcal{O}\left(\binom{p+q}{p}^2 tp^3n^2 + t\binom{p+q}{q}^\omega np\right) + (n+|E|)^{\mathcal{O}(1)}$, operations over $\mathbb{F}$.

(ii) A family $\widehat{\mathcal{S}}$ of size $np\binom{p+q}{p}$ in $\mathcal{O}\left(\binom{p+q}{p} tp^3n^2 + t\binom{p+q}{q}^{\omega-1}(pn)^{\omega-1}\right) + (n+|E|)^{\mathcal{O}(1)}$ operations over $\mathbb{F}$.

Proof. Let $p+q=k$ and $|E|=m$. We start by finding k-truncation of A, say A_k, over $\mathbb{F}[X] \subseteq \mathbb{F}(X)$ using Theorem 9.5. We can find A_k with at most $(n+m)^{\mathcal{O}(1)}$ operations over $\mathbb{F}$. Given the matrix A_k we follow the proof of [FLS14, Theorem 3.1]. For a set $S \in \mathcal{S}$ and $I \in \binom{[k]}{p}$, we define $s[I] = \mathsf{det}(A_k[I,S])$. We also define

$$\vec{s_i} = (s_i[I])_{I \in \binom{[k]}{p}}.$$

Thus the entries of the vector $\vec{s_i}$ are the values of $\mathsf{det}(A_k[I,S_i])$, where I runs through all the p sized subsets of rows of A_k. Let $H_{\mathcal{S}} = (\vec{s_1}, \ldots, \vec{s_t})$ be the $\binom{k}{p} \times t$ matrix obtained by taking $\vec{s_i}$ as columns. Observe that each entry in A_k is in $\mathbb{F}[X]^{<n}$. Thus, the determinant polynomial corresponding to any $p \times p$ submatrix of A_k has degree at most pn. It is well known that we can find determinant of a $p \times p$ matrix over $\mathbb{F}[X]^{<n}$ in time $\mathcal{O}(p^3n^2)$ [MS03]. Thus, we can obtain $H_{\mathcal{S}}$ in time $\mathcal{O}(t\binom{p+q}{p}p^3n^2)$.

Let W be a spanning set of columns for $\mathbf{C}(H_{\mathcal{S}})$. We define $\widehat{W} = \{S_\alpha \mid \vec{s_\alpha} \in W\}$ as the corresponding subfamily of $\mathcal{S}$. The proof of [FLS14, Theorem 3.1] implies that if W is a spanning set of columns for $\mathbf{C}(H_{\mathcal{S}})$ then the corresponding $\widehat{W}$ is the required q-representative family for $\mathcal{S}$. That is, $\widehat{W} \subseteq^q_{rep} \mathcal{S}$. We get the desired running time by either using Lemma 9.17 to compute a basis of size $\binom{p+q}{p}$ for $H_{\mathcal{S}}$ or by using Lemma 9.20 to compute a spanning set of size $np\binom{p+q}{p}$ of $\mathbf{C}(H_{\mathcal{S}})$. This completes the proof. □

In fact one can prove Theorem 9.11 for a "weighted notion of representative family".

9.4.1 Weighted Representative Families

In this section we give deterministic algorithms to compute weighted version of representative families of a linear matroid. A weighted version of q-representative families is defined as follows. It is useful for solving problems where we are looking for objects of maximum or minimum weight. Given a non-negative weight function $w : E \to \mathbb{R}^+$ and $A \subseteq E$, we define $w(A) = \sum_{a \in A} w(a)$.

Definition 9.12 (Min/Max q-Representative Family). *Given a matroid $M = (E, \mathcal{I})$, a family $\mathcal{S}$ of subsets of E and a non-negative weight function $w : \mathcal{S} \to \mathbb{R}^+$, we say that*

a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ *is* min q-representative *(*max q-representative*) for* $\mathcal{S}$ *if the following holds: for every set* $Y \subseteq E$ *of size at most* q*, if there is a set* $X \in \mathcal{S}$ *disjoint from* Y *with* $X \cup Y \in \mathcal{I}$*, then there is a set* $\widehat{X} \in \widehat{\mathcal{S}}$ *disjoint from* Y *with*

1. $\widehat{X} \cup Y \in \mathcal{I}$*; and*
2. $w(\widehat{X}) \leq w(X)$ *(*$w(\widehat{X}) \geq w(X)$*).*

We use $\widehat{\mathcal{S}} \subseteq^{q}_{minrep} \mathcal{S}$ *(*$\widehat{\mathcal{S}} \subseteq^{q}_{maxrep} \mathcal{S}$*) to denote a min* q*-representative (max* q*-representative) family for* $\mathcal{S}$*.*

The main theorem in this section is as follows.

Theorem 9.13. *Let* $M = (E, \mathcal{I})$ *be a linear matroid of rank* n *and let* $\mathcal{S} = \{S_1, \ldots, S_t\}$ *be a* p*-family of independent sets. Let* $w : \mathcal{S} \to \mathbb{R}^+$ *is a non-negative weight function on* $\mathcal{S}$*. Let* A *be a* $n \times |E|$ *matrix representing* M *over a field* $\mathbb{F}$*, where* $\mathbb{F} = \mathbb{F}_{p^\ell}$ *or* $\mathbb{F}$ *is* $\mathbb{Q}$*. Then there are deterministic algorithms computing* $\widehat{\mathcal{S}} \subseteq^{q}_{minrep} \mathcal{S}$ *as follows.*

1. *A family* $\widehat{\mathcal{S}}$ *of size* $\binom{p+q}{p}$ *in* $\mathcal{O}\left(\binom{p+q}{p}^2 tp^3n^2 + t\binom{p+q}{q}^{\omega} np\right) + (n+|E|)^{\mathcal{O}(1)}$*, operations over* $\mathbb{F}$*.*
2. *A family* $\widehat{\mathcal{S}}$ *of size* $np\binom{p+q}{p}$ *in* $\mathcal{O}\left(\binom{p+q}{p} tp^3n^2 + t\binom{p+q}{q}^{\omega-1}(pn)^{\omega-1}\right) + (n+|E|)^{\mathcal{O}(1)}$ *operations over* $\mathbb{F}$*.*

Proof. Let $p + q = k$ and $|E| = m$. We start by finding k-truncation of A, say A_k, over $\mathbb{F}[X] \subseteq \mathbb{F}(X)$ using Theorem 9.5. We can find A_k with at most $(n+m)^{\mathcal{O}(1)}$ operations over $\mathbb{F}$. Given the matrix A_k we follow the proof of [FLS14, Theorem 3.1]. For a set $S \in \mathcal{S}$ and $I \in \binom{[k]}{p}$, we define $s[I] = \mathsf{det}(A_k[I, S])$. We also define

$$\vec{s_i} = (s_i[I])_{I \in \binom{[k]}{p}}.$$

Thus the entries of the vector $\vec{s_i}$ are the values of $\mathsf{det}(A_k[I, S_i])$, where I runs through all the p sized subsets of rows of A_k. Let $H_{\mathcal{S}} = (\vec{s}_1, \ldots, \vec{s}_t)$ be the $\binom{k}{p} \times t$ matrix obtained by taking $\vec{s_i}$ as columns. Observe that each entry in A_k is in $\mathbb{F}[X]^{<n}$. Thus, the determinant polynomial corresponding to any $p \times p$ submatrix of A_k has degree at most pn. It is well

known that we can find determinant of a $p \times p$ matrix over $\mathbb{F}[X]^{<n}$ in time $\mathcal{O}(p^3n^2)$ [MS03]. Thus, we can obtain $H_{\mathcal{S}}$ in time $\mathcal{O}(t\binom{p+q}{p}p^3n^2)$.

Now we define a weight function $w' : \mathbf{C}(H_{\mathcal{S}}) \to \mathbb{R}^+$ on the set of columns of $H_{\mathcal{S}}$. For the column $\vec{s}_i$ corresponding to $S_i \in \mathcal{S}$, we define $w'(\vec{s}_i) = w(S_i)$. Let W be a spanning set of columns for $\mathbf{C}(H_{\mathcal{S}})$. We define $\widehat{W} = \{S_\alpha \mid \vec{s}_\alpha \in W\}$ as the corresponding subfamily of $\mathcal{S}$. Now we claim that if W is a nice spanning set of columns for $\mathbf{C}(H_{\mathcal{S}})$ or minimum weight column basis of $\mathbf{C}(H_{\mathcal{S}})$, then the corresponding $\widehat{W}$ is the required min q-representative family for $\mathcal{S}$. That is, $\widehat{W} \subseteq^q_{minrep} \mathcal{S}$. If W is a minimum weight column basis of $\mathbf{C}(H_{\mathcal{S}})$, the claim follows from the proof of [FLS14, Theorem 3.1].

Now we show that if W is a nice spanning set of columns for $\mathbf{C}(H_{\mathcal{S}})$, then $\widehat{W} \subseteq^q_{minrep} \mathcal{S}$. Let $S_\beta \in \mathcal{S}$ such that $S_\beta \notin \widehat{\mathcal{W}}$. We show that if there is a set $Y \subseteq E$ of size at most q such that $S_\beta \cap Y = \emptyset$ and $S_\beta \cup Y \in \mathcal{I}$, then there exists a set $\widehat{S}_\beta \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{S}_\beta \cup Y \in \mathcal{I}$ and $w(\widehat{S}_\beta) \leq w(S_\beta)$. Let us first consider the case $|Y| = q$. Since $S_\beta \cap Y = \emptyset$ we have that $|S_\beta \cup Y| = p+q = k$. Furthermore, since $S_\beta \cup Y \in \mathcal{I}$, we have that the columns corresponding to $S_\beta \cup Y$ in M are linearly independent over $\mathbb{F}(X)$; that is, $\mathsf{det}(A_k[\mathbf{R}(A_k), S_\beta \cup Y]) \not\equiv 0$. Recall that, $\vec{s}_\beta = (s_\beta[I])_{I \in \binom{[k]}{p}}$, where $s_\beta[I] = \mathsf{det}(A_k[I, S_\beta])$. Similarly we define $y[L] = \mathsf{det}(A_k[L, Y])$ and $\vec{y} = (y[L])_{L \in \binom{[k]}{q}}$.

Let $\sum J = \sum_{j \in S_\beta} j$. Define

$$\gamma(\vec{s}_\beta, \vec{y}) = \sum_{I \in \binom{[k]}{p}} (-1)^{\sum I + \sum J} s_\beta[I] \cdot y[\bar{I}].$$

Since $\binom{k}{p} = \binom{k}{k-p} = \binom{k}{q}$ the above formula is well defined. Observe that by Proposition 9.3, we have that $\gamma(\vec{s}_\beta, \vec{q}) = \mathsf{det}(A_k[\mathbf{R}(A_k), S_\beta \cup Y]) \not\equiv 0$. We also know that $\vec{s}_\beta$ can be written as a linear combination of vectors in $W = \{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_\ell\}$. That is, $\vec{s}_\beta = \sum_{i=1}^{\ell} \lambda_i \vec{s}_i$, $\lambda_i \in \mathbb{F}$

and for some i, $\lambda_i \neq 0$. Thus,

$$\begin{aligned}
\gamma(\vec{s}_\beta, \vec{y}) &= \sum_I (-1)^{\sum I + \sum J} s_\beta[I] \cdot y[\bar{I}] \\
&= \sum_I (-1)^{\sum I + \sum J} \left(\sum_{i=1}^{\ell} \lambda_i s_i[I] \right) y[\bar{I}] \\
&= \sum_{i=1}^{\ell} \lambda_i \left(\sum_I (-1)^{\sum I + \sum J} s_i[I] y[\bar{I}] \right) \\
&= \sum_{i=1}^{\ell} \lambda_i \mathsf{det}(A_k[\mathbf{R}(A_k), S_i \cup Y]) \qquad \text{(by Proposition 9.3)}
\end{aligned}$$

Define

$$\mathsf{sup}(S_\beta) = \left\{ S_i \;\middle|\; S_i \in \widehat{\mathcal{S}},\ \lambda_i \mathsf{det}(A_k[\mathbf{R}(A_k), S_i \cup Y])) \not\equiv 0 \right\}.$$

Since $\gamma(\vec{s}_\beta, \vec{y}) \neq 0$, we have that $(\sum_{i=1}^{\ell} \lambda_i \mathsf{det}(A_k[\mathbf{R}(A_k), S_i \cup Y])) \not\equiv 0$ and thus $\mathsf{sup}(S_\beta) \neq \emptyset$. Observe that for all $S \in \mathsf{sup}(S_\beta)$ we have that $\mathsf{det}(A_k[\mathbf{R}(A_k), S \cup Y]) \not\equiv 0$ and thus $S \cup Y \in \mathcal{I}$. Since W is a nice spanning set, If $\vec{s}_\beta = \sum_{i=1}^{\ell} \lambda_i \vec{s}_i$ and $0 \neq \lambda_i \in \mathbb{F}$, then $w(\vec{s}_\beta) \geq w(\vec{s}_i)$. Thus $w(\widehat{S}) \leq w(S)$ for all $S \in \mathsf{sup}(S_\beta)$. Thus $\widehat{S}$ is a min q-representative of S.

Suppose that $|Y| = q' < q$. Since M is a matroid of rank $k = p + q$, there exists a superset $Y' \in \mathcal{I}$ of Y of size q such that $S_\beta \cap Y' = \emptyset$ and $S_\beta \cup Y' \in \mathcal{I}$. This implies that there exists a set $\widehat{S} \in \widehat{\mathcal{S}}$ such that $\mathsf{det}(A_k[\mathbf{R}(A_k), \widehat{S} \cup Y']) \not\equiv 0$. Thus the columns corresponding to $\widehat{S} \cup Y$ are linearly independent.

Thus, if W is a minimum weight column basis of $\mathbf{C}(H_{\mathcal{S}})$ or a nice spanning set of columns for $\mathbf{C}(H_{\mathcal{S}})$ then the corresponding $\widehat{W}$ is a min q-representative family for $\mathcal{S}$. By applying Lemma 9.17 to compute a basis of size $\binom{p+q}{p}$ for $H_{\mathcal{S}}$, we get min q-representative family for $\mathcal{S}$ of size $\binom{p+q}{p}$ in $\mathcal{O}\left(\binom{p+q}{p}^2 tp^3n^2 + t\binom{p+q}{q}^{\omega} np\right) + (n + |E|)^{\mathcal{O}(1)}$, operations over $\mathbb{F}$. By applying Lemma 9.20 to compute a nice spanning set of size $np\binom{p+q}{p}$ of $\mathbf{C}(H_{\mathcal{S}})$, we get min q-representative family for $\mathcal{S}$ of size $np\binom{p+q}{p}$ in $\mathcal{O}\left(\binom{p+q}{p} tp^3n^2 + t\binom{p+q}{q}^{\omega-1} (pn)^{\omega-1}\right) + (n + |E|)^{\mathcal{O}(1)}$ operations over $\mathbb{F}$. This completes the proof. □

9.4.2 Applications

Marx [Mar09] gave algorithms for several problems based on matroid optimization. The main theorem in his work is Theorem 1.1 [Mar09] on which most applications of [Mar09] are based. The proof of the theorem uses an algorithm to find representative sets as a black box. Applying our algorithm (Theorem 9.11) instead gives a deterministic version of Theorem 1.1 of [Mar09].

Proposition 9.21. *Let $M = (E, \mathcal{I})$ be a linear matroid where the ground set is partitioned into blocks of size ℓ. Given a linear representation A_M of M, it can be determined in $\mathcal{O}(2^{\omega k\ell}||A_M||^{\mathcal{O}(1)})$ time whether there is an independent set that is the union of k blocks. ($||A_M||$ denotes the length of A_M in the input.)*

Finally, we mention another application from [Mar09] which we believe could be useful to obtain single exponential time parameterized and exact algorithms.

ℓ-Matroid Intersection	Parameter: k
Input: Let $M_1 = (E, \mathcal{I}_1), \ldots, M_1 = (E, \mathcal{I}_\ell)$ be matroids on the same ground set E given by their representations $A_{M_1}, \ldots, A_{M_\ell}$ over the same field $\mathbb{F}$ and a positive integer k.	
Question: Does there exist k element set that is independent in each M_i ($X \in \mathcal{I}_1 \cap \ldots \cap \mathcal{I}_\ell$)?	

Using Theorem 1.1 of [Mar09], Marx [Mar09] gave a randomized algorithm for ℓ-Matroid Intersection. By using Proposition 9.21 instead we get the following result.

Proposition 9.22. *ℓ-Matroid Intersection can be solved in $\mathcal{O}(2^{\omega k\ell}||A_M||^{\mathcal{O}(1)})$ time.*

Chapter 10

Derandomization of Transversal Matroids and Gammoids in Moderately Exponential Time

In this chapter, we design exponential time deterministic algorithms for constructing a representation of transversal matroids and gammoids, which are much faster than the typical brute-force algorithm. Let us recall the definition of the representation of a matroid. A matrix A over a field $\mathbb{F}$ is called a linear representation of a matroid $M = (E, \mathcal{I})$, if there is a bijection between the columns of A and E such that, a subset $S \subseteq E$ is independent in M if and only if the corresponding columns in A are linearly independent over the field $\mathbb{F}$. While not all matroids admit a linear representation, a number of important classes of matroids do. And these classes of matroids have many algorithmic applications which need a representation of the matroid. This naturally motivates the question of constructing linear representations for various classes of matroids *efficiently*. Deterministic polynomial time algorithms for construction of a representation were known for several classes of matroids such as uniform matroids, partition matroids, graphic matroids and co-graphic matroids [Oxl06]. In this chapter, we consider the transversal matroids and gammoids, for which, only randomized polynomial time algorithms are known for construction of a linear representations. These matroids feature in many recent FPT

and Kernelization algorithms [Mar09, FLS14, KW12, KW14], which are randomized only because they require a representation of the matroid. Hence, deterministic algorithms to find linear representations of these two matroids will derandomize these algorithms.

Let us recall the definition of transversal matroid. Let $G = (U \uplus V, E)$ be a bipartite graph. The *transversal matroid* M_G on the ground set U has the following family of independent sets: $U' \subseteq U$ is independent in M_G if and only if there is a matching in G saturating U'. Furthermore assume that $|U| = |V| = m$ and G has a perfect matching. A natural question in this direction is as follows.

> **Question 1:** Could we exploit the fact that G has a perfect matching to design a deterministic polynomial time algorithm to find a linear representation for M_G?

Answer to this question is of course, Yes! A $m \times m$ identity matrix is a linear representation of M_G. This naturally leads to the following question.

> **Question 2:** Suppose G has a matching of size $m - \ell$, where ℓ is a constant. Can we use this fact to design a deterministic polynomial time algorithm to find a linear representation for M_G?

In fact, no deterministic polynomial time algorithm is known for finding a linear representation for M_G even when $\ell = 1$. Now consider the following symbolic representation of a transversal matroid.

> Let $G = (U \uplus V, E)$ be a bipartite graph, where $U = \{v_1, \ldots, v_m\}$ and $V = \{v_1, \ldots, v_n\}$. Let $X = \{x_{i,j} \mid i \in [n], j \in [m]\}$. Define a $n \times m$ matrix A as follows: for each $i \in [n], j \in [m], A[i,j] = 0$ if $(v_i, u_j) \notin E$ and $x_{i,j}$ otherwise. Then for any $R \subseteq [n], C \subseteq [m], |R| = |C|$, $\mathsf{det}(A[R,C]) \not\equiv 0$ if and only if there is a perfect matching in $G[\{v_i \mid i \in R\} \cup \{u_j \mid j \in C\}]$. This implies that A is in fact a linear representation of the transversal matroid M_G on the ground set U over the field of fractions $\mathbb{F}(X)$, where $\mathbb{F}$ is any field of size at least 2.

However, in the representation above, to check whether a set is linearly independent we need to check whether there is a corresponding determinant polynomial which is identically non-zero. This is a case of the well known polynomial identity testing (PIT) problem, and we do not know of a deterministic polynomial time algorithm for this problem. It appears that derandomizing the above approach has some obstacles, as this will have some important consequences on lower bounds in complexity theory [KI04]. We can obtain another representation, by substituting random values for each $x \in X$ from a field of size at least $2^p m 2^m$, where $p \in \mathbb{N}$, and succeed with probability at least $\left(1 - \frac{1}{2^p}\right)$. This leads to a randomized polynomial time construction [Mar09]. Observe that, the above approach also gives a deterministic algorithm of running time $2^{\mathcal{O}(m^4)}$, that tests all possible substitutions from a field of size $2m2^m$, since one of them will certainly be a linear representation of M_G.

In this chapter, we answer Questions 2 affirmatively. Our main theorem is the following.

Theorem 10.1. *Let $G = (U \uplus V, E)$ be a bipartite graph and r be the size of a maximum matching in G. Let $\mathbb{F}$ be a field of size strictly greater than $\binom{|U|}{r}$. Then there is a deterministic algorithm which outputs a linear representation of the transversal matroid $M_G = (U, \mathcal{I})$ over $\mathbb{F}$ in time $\mathcal{O}(\binom{|U|}{r}|V| \cdot |E| + N)$, where N is time required to do $\mathcal{O}(\binom{|U|}{r}|V| \cdot |U|)$ operations in $\mathbb{F}$.*

In the language of parameterized complexity, Theorem 10.1 gives an XP algorithm for finding a linear representation of transversal matroids parameterized by the rank of the given matroid. Observe that if r is the rank of the matroid then the size of a maximum matching of the graph is exactly r. That is, $r = |U| - \ell$ and hence $\ell = |U| - r$. This together with the fact that $\binom{m}{a} = \binom{m}{m-a}$ implies that Theorem 10.1 gives a polynomial time algorithm for Question 2, whenever ℓ is a constant.

Recall that, transversal matroids and strict gammoids are duals. Therefore obtaining an algorithm to construct a linear representation of a gammoid is at least as hard as that of transversal matroids. In this work we prove the following theorem.

Theorem 10.2. *Let D be a n-vertex digraph, $S \subseteq V(D)$, $|S| = r$, $T \subseteq V(D)$, $|T| = n'$ and $\mathbb{F}$ be a field of size strictly greater than $\binom{n'}{r}$. Then there is a deterministic algorithm which outputs a linear representation of a gammoid with ground set T, over $\mathbb{F}$ in time*

$\mathcal{O}(\binom{n'}{r}n^3 + N)$, *where* N *is time required to do* $\mathcal{O}(\binom{n'}{r}n^3)$ *operations in* $\mathbb{F}$.

10.1 The Algorithm

In this section, we first give a deterministic algorithm to compute a linear representation of transversal matroids. In the next section, we show that this algorithm may be modified to obtain more efficient algorithms for other classes of matroids that are related to transversal matroids.

Let $G = (U \uplus V, E)$ be a bipartite graph such that U and V contain m and n vertices, respectively. Let $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$. Let $M_G = (U, \mathcal{I})$ be the transversal matroid associated with G where the ground set is U. That is, $S \subseteq U$ is independent in M_G if and only if there is a matching in G, saturating S. Let A' be the bipartite adjacency matrix of G. That is, the rows of A' are indexed with elements from V, columns of A' are indexed with elements from U, and for any $v \in V, u \in U$, $A'[v, u] = 1$ if and only if $vu \in E$. Let A be an $n \times m$ matrix defined as follows. For any $v_i \in V, u_j \in U$, $A[v_i, u_j] = A'[v_i, u_j] \cdot x_{i,j}$, where $X = \{x_{i,j} \mid i \in [n], j \in [m]\}$ is a set of variables. Notice that for any $v \in V, u \in U$, $A[v, u] = 0$ if and only if $vu \notin E$. Also, note that each variable $x_{i,j}$ appears at most once in the matrix A. As mentioned earlier, there is a deterministic algorithm to find this representation in $2^{\mathcal{O}(m^2 n)}$ time.

We now describe a more efficient algorithm for this problem. We call this algorithm, Algorithm $\mathcal{A}$. For each $j \in [m]$, we define a set $X_j = \{x_{i,j} \mid i \in [n]$ *and* $x_{i,j}$ *is an entry of* $A\}$, i.e. it is the j-th column of A. Our algorithm is an iterative algorithm that produces values for $X_1, X_2, \ldots, X_m$ in order, by solving a system of linear inequalities the variables in X_i in the i-th iteration. Let r be the size of a maximum matching in G. Now we define a family of subsets of U as follows.

$$\mathcal{B} = \left\{ S \in \binom{U}{r} \;\middle|\; \textit{there is a matching in } G \textit{ saturating } S \right\}.$$

That is $\mathcal{B}$ is the set of bases in M_G and let $\mathcal{B} = \{S_1, \ldots, S_t\}$. Now, for any $S \in \mathcal{B}$, we fix a matching $\mathsf{M}(S)$ saturating S. For any $S \in \mathcal{B}$, let $R(S)$ be the set of vertices from V,

saturated by $\mathsf{M}(S)$. Note that $R(S) \in \binom{V}{r}$, and $G[S \cup R(S)]$ has a perfect matching (a matching of size r). Our goal is to assign values to all the variables in X from a field such that for any $S \in \mathcal{B}$, $\mathsf{det}(A[R(S), S]) \neq 0$. We will then show that this is enough to produce a linear representation of M_G (the details may be found in the correctness proof presented later in this subsection).

Let us now describe the steps of our algorithm. Recall that for each $j \in [m]$, $X_j = \{x_{i,j} \mid i \in [n]\}$ and let $X_{<j} = \bigcup_{j'=1}^{j-1} X_{j'}$ for any $j \in [m] \setminus \{1\}$. For $S \subseteq U$, and $j \in [m]$, we define $S(j) = \{u_{j'} \in S \mid j' \in [j]\}$. Let $\mathbb{F}$ be a field of size strictly more than $\binom{m}{r}$. Our algorithm assigns values for the variables in X_j in the increasing order of j. Initially, in the first iteration, set all the variables in X_1 to 1, i.e. for all $i \in [n]$, $x_{i,1} = 1$. Then, in j-th iteration, for $j \geq 2$, the algorithm will assign values for variables in X_j as follows. Note that at this stage, all the variables in $X_{<j}$ have been assigned values already. We denote by A_{j-1} the matrix A instantiated with the values for $X_{<j}$.

For any $S_i \in \mathcal{B}$, recall that $\mathsf{M}(S_i)$ is a fixed matching. Let M_{ij} be the set of edges in $\mathsf{M}(S_i)$ which saturate the vertices in $S_i(j)$. Let R_{ij} be the subset of V saturated by M_{ij}. Notice that the matching M_{ij} saturates the vertices $S_i(j) \cup R_{ij}$. Now the algorithm obtains values for X_j by solving the following t inequalities, one for each $S_i \in \mathcal{B}$.

$$\mathsf{det}(A_{j-1}[R_{ij}, S_i(j)]) \neq 0 \tag{10.1}$$

Observe that for each $i \in [t]$, $\mathsf{det}(A_{j-1}[R_{ij}, S_i(j)])$ is a linear function of the variables in X_j, since all the entries in $A_{j-1}[R_{ij}, S_i(j)]$ are elements from $\mathbb{F}$ except the entries in the last column which are either 0 or variables from X_j. Therefore, (10.1) is a system of linear inequations in the variables in X_j. We can express this system as,

$$DX_j \neq \vec{0} \tag{10.2}$$

where D is a $t' \times n$ matrix for some $t' \leq t$ and $\vec{0}$ is a zero column vector of length t'. All the entries in DX_j are linear functions, and note that the constraints require that every one of these linear functions is non-zero. Furthermore, we also know that $|\mathbb{F}| > t'$. Now our algorithm will execute the following algorithm (Lemma 10.1) to find a solution to the

system (10.2).

Lemma 10.1. *Let $\mathbb{F}$ be a field of size strictly greater than t and let D be a $t \times n$ matrix over $\mathbb{F}$ such that no row vector in D contains only zeros. Then there is an n-length column vector Y^* such that $DY^* \neq \vec{0}$. Moreover such a vector can be computed using $\mathcal{O}(t \cdot n)$ operations over the field $\mathbb{F}$.*

Proof. Let $Y = [y_1, \ldots, y_n]^T$ be a column vector containing n variables. We will directly give an n step iterative process to compute Y^*, an instantiation of Y satisfying the system of linear inequations $DY \neq \vec{0}$. Initially, set all the variables in Y to be 0 and we use $Y(0)$ to denote this assignment. In other words, $Y(0)$ be an n length zero column vector. At each step we find out new assignment for Y. In step i we find out an assignment $Y(i)$ for Y and prove that indeed $D \cdot Y(n) \neq \vec{0}$. Now for any $i \in [n]$, at step i, $Y(i)$ is computed as follows. Note that at this step we have the assignment $Y(i-1)$ for Y. In other words, $Y(i-1)$ is an n-length column vector such that the j^{th} entry is same as the value assigned for y_j in step $i-1$. Let Z_i be an n-length column vector where all entries are same as the entries in $Y(i-1)$, except the i^{th} entry which is the variable y_i. That is, in Z_i, we did not assign any value to y_i, but all other variables are assigned the same value as in the assignment $Y(i-1)$. Now consider the entries in the column vector DZ_i. Some entries are elements in the field $\mathbb{F}$ and some are linear functions on variable y_i. Let $P_1(y_i) = p_1 y_i + q_1, \ldots, P_{t'}(y_i) = p_{t'} y_i + q_{t'}$ be the entries in DZ_j which are linear functions. Notice that $t' \leq t$. Since the size of the field $\mathbb{F}$ is strictly larger than $t \geq t'$, there is an element $a \in \mathbb{F}$ such that for all $j \in [t']$, $P_j(y_i) \neq 0$. Now we set $Y(i)$ as follows. Each entry in $Y(i)$ is same as each entry in $Y(i-1)$, except the i^{th} entry which is set to a. Our algorithm will output $Y(n)$ as the required column vector. The correctness of the algorithm follows from the claim below.

Claim 1. *For all $i \in \{0, 1, \ldots, n\}$, $Y(i)$ satisfies the set of inequalities in $DY \neq \vec{0}$, containing variables from $\{y_1, \ldots, y_i\}$.*

Proof. We prove the claim by induction on i. For the base case, when $i = 1$, we set all variables to 0 except y_1, which is set to a value such that the set of inequalities in $DZ_1 \neq \vec{0}$ are satisfied. This implies that $Y(1)$ satisfies the set of inequalities in $DY \neq \vec{0}$ containing

y_1. Now suppose the statement holds true for all $i < k$, for some $k < n$, by induction hypothesis. We wish to show that $Y(k)$ satisfies all inequalities that contain a variable from $\{y_1, \ldots, y_k\}$. We divide the set of all such inequalities into those that don't contain y_k and those that do, which we call D_1 and D_2 respectively. Now observe that the inequalities in D_1 are unaffected by the choice of y_k and therefore they continue to hold as $Y(k-1)$ and $Y(k)$ agree on all values except for y_k. And for the inequalities in D_2, we first substitute values for all the other variables to obtain the system $DZ_k \neq \vec{0}$. Then we choose a value for y_k so that all these inequalities are satisfied. Hence, $Y(k)$ satisfies all inequalities which contains a variable from $\{y_1, \ldots, y_k\}$. Therefore $Y(n)$ satisfies all the inequalities. This completes the proof of the claim. □

At step i, the algorithm computes DZ_i and chooses a value for y_i by looking at a linear function of y_i in the t-length column vector DZ_i. Since Z_{i-1} and Z_i differ only in two entries, DZ_i can be computed from DZ_{i-1}, and the $(i-1)^{st}$ and i^{th} columns of D, using $\mathcal{O}(t)$ operations over $\mathbb{F}$. Since algorithm has n iterations, the number of field operations performed is $\mathcal{O}(t \cdot n)$. □

Our algorithm iterates over all values of j from $1, 2, \ldots m$, and produces an assignment of values from the field $\mathbb{F}$ for the variables in X_j in the j-th iteration. After m iterations an assignment for all variables will have been computed, and we let A_m be the instantiation of the matrix A with this assignment. Our algorithm outputs A_m as the representation of the transversal matroid M_G. This completes the description of Algorithm $\mathcal{A}$. The following two lemmata are required for proving the correctness of Algorithm $\mathcal{A}$.

Lemma 10.2. *For any $j \in [m]$ and $S_i \in \mathcal{B}$, let M_{ij} be the set of edges in $\mathsf{M}(S_i)$ which saturates $S_i(j)$. Let $R_{ij} \subseteq V$ be the set of vertices in V saturated by M_{ij}. Then $\mathsf{det}(A_j[R_{ij}, S_i(j)]) \neq 0$.*

Proof. We prove the lemma using induction on j. For the base case, $j = 1$. Let $S_i \in \mathcal{B}$. If $S_i(1) = \emptyset$, then the conclusion of the statement is empty and the lemma holds trivially. Otherwise, M_{ij} contains only one edge (say $u_1 v_{i'}$). That is, $R_{ij} = \{v_{i'}\}$. Then $\mathsf{det}(A_1[R_{ij}, S_i(j)]) = A_1[v_{i'}, u_1] = 1$, because $A'[v_{i'}, u_1] = x_{i',1}$ is set to 1. For the in-

duction hypothesis we assume that the statement in the lemma holds for all values $j' < j$. Now consider the induction step for $j \in [m]$. Note that $|R_{ij}| = |S_i(j)|$, and let $S_i(j) = \{u_{j_1}, \ldots u_{j_\ell}\}$ and $R_{ij} = \{v_{i_1}, \ldots, v_{i_\ell}\}$ where $j_1 < \ldots < j_\ell$ and $i_1 < \ldots < i_\ell$. We need to show that $\mathsf{det}(A_j[R_{ij}, S_i(j)]) \neq 0$. If $j_\ell = j' < j$ then $\mathsf{det}(A_j[R_{ij}, S_i(j)]) = \mathsf{det}(A_{j'}[R_{ij}, S_i(j)]) \neq 0$, by the induction hypothesis. Otherwise $j_\ell = j$. Now consider the determinant $\mathsf{det}(A_{j-1}[R_{ij}, S_i(j)])$.

$$\begin{aligned}\mathsf{det}(A_{j-1}[R_{ij}, S_i(j)]) &= \sum_{k=1}^{\ell} A_{j-1}[v_{i_k}, u_j] \cdot (-1)^{k+\ell} \cdot \mathsf{det}(A_{j-1}[R_{ij} \setminus \{v_{i_k}\}, S_i(j) \setminus \{u_j\}]) \\ &= \sum_{k=1}^{\ell} [v_{i_k} u_j] x_{i_k,j} (-1)^{k+\ell} \cdot \mathsf{det}(A_{j-1}[R_{ij} \setminus \{v_{i_k}\}, S_i(j) \setminus \{u_j\}])\end{aligned} \tag{10.3}$$

In the above equation $[v_{i_k} u_j] = 1$ if $v_{i_k} u_j \in E$ and 0 otherwise. Let $v_{k'} u_j$ be the edge in M_{ij} which is incident to u_j. This implies that $\mathsf{M}_{ij} \setminus \{v_{k'} u_j\}$ are the edges in $\mathsf{M}(S_i)$, which saturate $S_i(j-1)$. Thus, by the induction hypothesis, $\mathsf{det}(A_{j-1}[R_{ij} \setminus \{v_{i_k}\}, S_i(j) \setminus \{u_j\}]) = \mathsf{det}(A_{j-1}[R_{ij} \setminus \{v_{i_k}\}, S_i(j-1)]) \neq 0$. This implies that $\mathsf{det}(A_{j-1}[R_{ij}, S_i(j)])$ is a non-zero linear function of the variables in X_j, Therefore, in Step j, the algorithm would select values for X_j such that $\mathsf{det}(A_j[R_{ij}, S_i(j)]) \neq 0$. This completes the proof of the lemma. □

Lemma 10.3. *Let $S' \subseteq U$ such that there is no matching saturating S' (or in other words $S' \notin \mathcal{I}$). Then the columns corresponding to S' in A_m are linearly dependent.*

Proof. Suppose the columns corresponding to S' in A_m are linearly independent. Then then there is a subset $V' \subseteq V$ such that $\mathsf{det}(A_m[V', S']) \neq 0$. That is,

$$\mathsf{det}(A_m[V', S']) = \sum_{f: S' \xrightarrow{1-1} V'} \prod_{s \in S'} A_m[f(s), s] \neq 0,$$

where the summation is taken over all one to one maps f from S' to V'. This implies that there is a one to one map $f' : S' \to V'$ such that $\prod_{s \in S'} A_m[f'(s), s] \neq 0$ and hence $A_m[f'(s), s] \neq 0$ for all $s \in S'$. This implies that the pair $(f'(s)s)$ is an edge in E for all $s \in S'$, which gives a matching in G saturating S'. This is a contradiction to the assumption that $S' \notin \mathcal{I}$. □

Now we prove the correctness of Algorithm $\mathcal{A}$.

Lemma 10.4. *The matrix A_m is a linear representation of the matroid $M_G = (U, \mathcal{I})$.*

Proof. We need to show that for any $U' \subseteq U$, $U' \in \mathcal{I}$ if and only if the columns in A_m indexed with elements from U' are linearly independent. If $U' \notin \mathcal{I}$, then by Lemma 10.3, the columns in A_m indexed with elements from U' are linearly dependent.

Now we need to show that if $U' \in \mathcal{I}$, then the corresponding columns in A_m are linearly independent. Since $(U, \mathcal{I})$ is a matroid, there is a set $S \in \mathcal{B}$ such that $U' \subseteq S$. Note that $S(m) = S$ and $\mathsf{M}(S)$ is a fixed matching. Let R be the subset of V, saturated by the matching $\mathsf{M}(S)$. By Lemma 10.2 , $\mathsf{det}(A_m[R, S]) \neq 0$. This implies that the columns of A_m corresponding to S are linearly independent and since $U' \subseteq S$, the columns of A_m corresponding to U' are also linearly independent. This completes the proof of the lemma. □

Lemma 10.5. *Algorithm $\mathcal{A}$ runs in time $\mathcal{O}(\binom{m}{r} \cdot |E|\sqrt{n} + N)$, where N is the time required to perform $\mathcal{O}(\binom{m}{r} n \cdot m)$ operations over $\mathbb{F}$.*

Proof. Algorithm $\mathcal{A}$ first computes $\mathcal{B}$ and for each $S \in \mathcal{B}$ a matching $\mathsf{M}(S)$. This can be done by executing the bipartite maximum matching algorithm for each r-vertex subset of U. Since there are $\binom{m}{r}$ such subsets and each execution of the bipartite maximum matching algorithm is on a bipartite graph having at most $r + n \leq 2n$ vertices, this step takes time $\mathcal{O}(\binom{m}{r} \cdot |E|\sqrt{n})$ [HK73]. Following this, Algorithm $\mathcal{A}$ executes the algorithm of Lemma 10.1 once for each X_i, where $i \in [m]$. Since each execution of this algorithm takes $\mathcal{O}(\binom{m}{r} n)$ field operations, the total number of field operations required for the second phase of Algorithm $\mathcal{A}$ is $\mathcal{O}(\binom{m}{r} mn)$. This completes the proof of the lemma. □

The correctness and running time bounds we have obtained for Algorithm $\mathcal{A}$ imply Theorem 10.1.

10.2 Representing matroids related to transversal matroids

In this section, we give deterministic algorithms for constructing linear representations of gammoids and strict gammoids. These algorithms utilize the algorithm for constructing linear representation of transversal matroids.

10.2.1 Truncations and contractions of transversal matroids

Several algorithmic applications require a linear representation of the k-truncation of matroids [LMPS15, GMP+15, FGPS]. One way of deterministically computing a linear representation of the k-truncation of a transversal matroid $M_G = (U, \mathcal{I})$ of rank r is as follows. We first execute Algorithm $\mathcal{A}$ of Theorem 10.1 and *then* truncate the computed represenation matrix using Theorem 9.7. However, observe that the running time of Algorithm $\mathcal{A}$ is $\Omega(\binom{|U|}{r})$ since it has to iterate over all bases of the matroid.

However, when k is much smaller than r as is often the case when designing FPT algorithms, we can get a faster algorithm to compute a linear representation of the k-truncation of M_G by slightly modifying Algorithm $\mathcal{A}$ and using Theorem 9.7. In the new algorithm, call it Algorithm $\mathcal{A}'$, we define $\mathcal{B}$ as follows.

$$\mathcal{B} = \left\{ S \in \binom{U}{k} \;\middle|\; \textit{there is a matching in } G \textit{ saturating } S \right\}.$$

That is, $\mathcal{B}$ is directly defined to be the set of bases in the k-truncation of M_G. Then we follow the steps of algorithm $\mathcal{A}$. This algorithm will output an $n \times m$ matrix $\hat{A}$ over a field of size strictly more than $\binom{|U|}{k}$. Lemmata 10.2 and 10.3 are clearly true for Algorithm $\mathcal{A}'$ as well. That is, all the columns corresponds to a basis in k-truncation of M_G form a set of linearly independent vectors. But there may be a set of columns of size strictly greater than k which are also linearly independent. To get rid of this, we apply Theorem 9.7 to obtain the k-truncation of $\hat{A}$, which gives us a linear representation of the k-truncation of M_G.

Theorem 10.3. *There is a deterministic algorithm that, given a bipartite graph $G = (U \uplus V, E)$ and $k \in \mathbb{N}$, outputs a linear representation of the k-truncation of the transversal*

matroid $M_G = (U, \mathcal{I})$ *over a field* $\mathbb{F}(Y)$ *where* $\mathbb{F}$ *has size strictly greater than* $\binom{|U|}{k}$*, in time* $\mathcal{O}(\binom{|U|}{k}|V| \cdot |E| + N)$*, where* N *is the time required to perform* $\mathcal{O}(\binom{|U|}{k}|V| \cdot |U|)$ *operations over* $\mathbb{F}$*.*

Contractions of transversal matroids. Other useful matroids are matroids obtained by contracting some elements of the ground set of a second matroid. Like truncation, contraction also reduces the rank of the matroid. Here, we give a faster algorithm to compute a linear representation for a contracted transversal matroid by modifying Algorithm $\mathcal{A}$. We will use this linear representation to obtain a linear representation of gammoids (see Subsection 10.2.2). Let $M_G = (U, \mathcal{I})$ be the transversal matroid associated with the graph $G = (U \uplus V, E)$ on ground set U. Let $F \subseteq U$. One way of getting a representation of M_G/F is to find a linear representation of M_G and then find a linear representation of M_G/F by applying Proposition 5.4, but we can do better. Let r be the size of a maximum matching in G, that is $r_{M_G}(U) = r$. Let $r_{M_G}(F) = \ell$ and $k = r_{M_G}(U) - r_{M_G}(F)$. Note that the rank of M_G/F is k. Now we explain how to modify Algorithm $\mathcal{A}$ to get an algorithm, $\mathcal{A}''$, for computing a linear representation of M_G/F. Towards that we first define $\mathcal{B}$ as follows.

$$\mathcal{B} = \left\{ S \in \binom{U}{r} \;\middle|\; \textit{there is a matching in } G \textit{ saturating } S \textit{ and } |S \setminus F| = k \right\}.$$

Now, Algorithm $\mathcal{A}''$ follows the steps of Algorithm $\mathcal{A}$ and it constructs an $n \times m$ matrix A_m. Lemmata 10.2 and 10.3 are true in this case as well. Let $M[A_m] = (E, \mathcal{I}'')$ be the matroid represented by the matrix A_m. Now Algorithm $\mathcal{A}''$ run the algorithm mentioned in Proposition 5.4 to compute a linear representation C of $M[A_m]/F$.

Lemma 10.6. *The matrix* C *is a linear representation of* M_G/F*.*

Proof. Let $Q = M[A_m]/F$ be the matroid represented by C. To prove Q is indeed M_G/F, we need to show that for all $T \subseteq E \setminus F$, $r_Q(T) = r_{M_G}(T \cup F) - r_{M_G}(F)$. So it is enough to show that (i) $r_{M[A_m]}(F) = r_{M_G}(F)$ and (ii) for any $T \subseteq E \setminus F$, $r_{M[A_m]}(T \cup F) = r_{M_G}(T \cup F)$. Since statement (ii) includes statement (i) as well, when $T = \emptyset$, it is enough to show that statement (ii) is true.

Fix an arbitrary $T \subseteq E \setminus F$. Let $F' \subseteq F$ such that $|F'| = r_{M_G}(F) = \ell$. By Proposition 5.1, there is set $T' \subseteq T$ such that $|T' \cup F'| = r_{M_G}(T' \cup F') = r_{M_G}(T \cup F)$. Again, by Proposition 5.1, there is set $S' \in E \setminus (T' \cup F')$ such that $|S' \cup T' \cup F'| = r_{M_G}(S' \cup T' \cup F') = r$. Since $|S' \cup T'| = r - \ell = k$, the set $S' \cup T' \cup F'$ belongs to $\mathcal{B}$. Thus, by Lemma 10.2, columns corresponding to $S' \cup T' \cup F'$ are linearly independent in A_m. This implies that $r_{M[A_m]}(T \cup F) \geq r_{M[A_m]}(T' \cup F') = |T' \cup F'| = r_{M_G}(T \cup F)$. Since $|T' \cup F'|$ is the size of a maximum matching in $G[T \cup F \cup V]$, by Lemma 10.3, $r_{M[A_m]}(T \cup F) \leq |T' \cup F'| = r_{M_G}(T \cup F)$. Therefore, we have that $r_{M[A_m]}(T \cup F) = r_{M_G}(T \cup F)$. This completes the proof of the lemma. □

Now consider the running time of Algorithm $\mathcal{A}''$. Let M be a maximum matching in $G[F \cup V]$ and the F' be the set of vertices from F saturated by M. By Proposition 5.1, for any $S' \subseteq U \setminus F$ of cardinality k, there is a matching of size r in $G[F \cup S']$ if and only if there is a matching of size r in $G[F' \cup S']$. This implies that algorithm $\mathcal{A}''$ can construct $\mathcal{B}$, by running the bipartite maximum matching algorithm at most $\binom{n-|F|}{k}$ times. Thus, we get the following theorem.

Theorem 10.4. *There is a deterministic algorithm that, given a bipartite graph $G = (U \uplus V, E)$ and a vertex set $F \subseteq U$, outputs a linear representation of M_G/F over a field $\mathbb{F}$ of size strictly greater than $\binom{|U|-|F|}{k}$, in time $\mathcal{O}(\binom{|U|}{k}|V| \cdot |E| + N)$, where $k = \mathsf{rank}(M_G/F)$ and N is the time required to perform $\mathcal{O}(\binom{|U|-|F|}{k}|V| \cdot |U|)$ operations over $\mathbb{F}$.*

10.2.2 Gammoids

In this subsection we explain how to get a linear representation of a gammoid efficiently. By Lemma 7.1, we know that there is a polynomial time algorithm which given a digraph D and $S \subseteq V(D)$, outputs a bipartite graph $G = (T \uplus V', E)$, where $T = V(D)$ and $V' = V(D) \setminus S$, such that the strict gammoid with respect to D and S is the dual of the transversal matroid M_G on the ground set T. Thus by Lemma 7.1, Theorem 10.1 and Proposition 5.4, we get the following theorem.

Theorem 10.5. *Let D be an n-vertex digraph, $S \subseteq V(D)$, $|S| = r$ and $\mathbb{F}$ be a field of size strictly greater than $\binom{n}{n-r}$. Then there is a deterministic algorithm which outputs*

a linear representation of the strict gammoid with respect to D and S, over $\mathbb{F}$ in time $\mathcal{O}(\binom{n}{n-r}n^3 + N)$, where N is the time required to perform $\mathcal{O}(\binom{n}{n-r}n^2)$ operations over $\mathbb{F}$.

One way to get a representation of a gammoid is to first construct a representation of the *strict* gammoid in the graph and then delete some elements from the strict gammoid. However, observe that if we compute the representation of the strict gammoid via the algorithm of Theorem 10.5, the running time depends on the total number of vertices in the graph. We can obtain a much faster algorithm as follows. Let D be a digraph and let S and W be subsets of $V(D)$. Let M be the gammoid in D with ground set $W \subseteq V(D)$, with respect to $S \subseteq V(D)$ of rank r. We may assume that $r = |S|$. Otherwise, we construct the graph D' obtained from D by adding S', a set of r new vertices, and all possible edges from S' to S. Now consider the gammoid in D' with ground set W with respect to S'. It is easy to see that, for any subset of X of W, there has $|X|$ vertex disjoint paths from S' to X in D' if and only if there are $|X|$ vertex disjoint paths from S to X in D, So these two gammoids are the same and our assumption holds. Now let M_S be the strict gammoid in D with respect to S and note that the rank of M_S and M are same. Let M_S^* be the transversal matroid which is the dual of M_S and it is defined on the bipartite graph $G = (V(D) \uplus V', E)$, where $V' = V(D) \setminus S$. Now let N be the matroid obtained from M_S^* by contracting $V(D) \setminus W$. It is easy to see the following lemma.

Lemma 10.7. $M = N^*$

Proof. Since N is a matroid obtained by contracting $V(D) \setminus W$ in M_S^*, Proposition 5.4 implies that N^* is the matroid $M_S \setminus (V(D) \setminus W)$. That is $N^* = M$. □

Combining Lemma 10.7, Theorem 10.4 and Proposition 5.4 we obtain Theorem 10.2.

Algorithms for Graph Connectivity Problems

Chapter 11

Finding Even Subgraphs Even Faster

Many well-studied algorithmic problems on graphs can be phrased in the following way: Let $\mathcal{F}$ be a family of graphs or digraphs. Given as input a graph (digraph) G and a positive integer k, can we delete k vertices (or edges or arcs) from G such that the resulting graph (digraph) belongs to the class $\mathcal{F}$? Recent research in parameterized algorithms has focused on problems of this kind where the class $\mathcal{F}$ consists of all graphs/digraphs whose vertices satisfy certain *parity* constraints [CMP+14, FG14, CY11, DGvHP14]. In this chapter we obtain significantly faster parameterized algorithms for two such problems, improving the previous best bounds due to Cygan et al. [CMP+14]. We also settle the parameterized complexity of a third problem, disproving a conjecture of Cai and Yang [CY11] and solving an open problem posed by Fomin and Golovach [FG14].

An undirected graph G is *even* (respectively, *odd*) if every vertex of G has even (resp. odd) degree. A directed graph D is *balanced* if the in-degree of each vertex of D is equal to its out-degree. An undirected graph is *Eulerian* if it is connected and even; and a directed graph is *Eulerian* if it is strongly connected and balanced. Cai and Yang [CY11] initiated the systematic study of parameterized Eulerian subgraph problems. In this work we take up the following edge-deletion problems of this kind:

Undirected Eulerian Edge Deletion	Parameter: k
Input: A connected undirected graph G and an integer k.	
Question: Does there exist a set S of at most k edges in G such that $G \setminus S$ is Eulerian?	

Undirected Connected Odd Edge Deletion	Parameter: k
Input: A connected undirected graph G and an integer k.	
Question: Does there exist a set S of at most k edges in G such that $G \setminus S$ is odd and connected?	

Directed Eulerian Edge Deletion	Parameter: k
Input: A strongly connected directed graph D and an integer k.	
Question: Does there exist a set S of at most k arcs in D such that $D \setminus S$ is Eulerian?	

Our algorithms for these problems also find such a set S of edges/arcs when it exists; so we slightly abuse the notation and refer to S as a *solution* to the problem in each case.

Previous Work and Discussion

Cai and Yang [CY11] listed sixteen odd/even undirected subgraph problems in their pioneering paper, and settled the parameterized complexity of all but four. The first two problems above are among these four; Cai and Yang conjectured that these are both W[1]-hard, and so are unlikely to have fixed-parameter tractable (FPT) algorithms: those with running times of the form $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f where n is the number of vertices in the input graph. Cygan et al. [CMP+14] disproved this conjecture for the first problem: they used a novel and non-trivial application of the colour-coding technique to solve both Undirected Eulerian Edge Deletion and Directed Eulerian Edge Deletion in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$. They also posed as open the question whether there exist $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$-time algorithms for these two problems. It was also posed as an open problem at the School on Parameterized Algorithms and Complexity 2014, Będlewo, Poland [CFJ+]. Fomin and Golovach [FG14] settled the parameterized complexity of the other two problems—not defined here—left open by Cai and Yang, but left the status of Undirected Connected Odd Edge Deletion open.

We devise deterministic algorithms which run in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ for all the three problems defined above. This answers the question of Cygan et al. [CMP+14] in the affirmative, solves the problem posed by Fomin and Golovach, and disproves the conjecture of Cai and Yang for Undirected Connected Odd Edge Deletion.

Theorem 11.1. Undirected Eulerian Edge Deletion, Undirected Connected Odd Edge Deletion, *and* Directed Eulerian Edge Deletion *can all be solved in time* $\mathcal{O}(2^{(2+\omega)k} \cdot n^2m^3k^6) + m^{\mathcal{O}(1)}$ *where* $n = |V(G)|$, $m = |E(G)|$ *and* ω *is the exponent of matrix multiplication.*

Our main conceptual contribution is *to view the solution as an independent set of a co-graphic matroid*, which we believe will be useful in other problems where one of the constraints that need to be satisfied is that of connectivity.

We now give a high-level overview of our algorithms. Given a subset of vertices T of a graph G, a *T-join* of G is a set $S \subseteq E(G)$ of edges such that T is exactly the set of odd degree vertices in the subgraph $H = (V(G), S)$. Observe that T-joins exist only for even-sized vertex subsets T. The following problem is long known to be solvable in polynomial time [EJ73].

Min T-Join
Input: An undirected graph G and a set of terminals $T \subseteq V(G)$.
Question: Find a T-join of G of the smallest size.

Consider the two problems we get when we remove the connectivity (resp. strong connectivity) requirement on the graph $G \setminus S$ from Undirected Eulerian Edge Deletion and Directed Eulerian Edge Deletion; we call these problems Undirected Even Edge Deletion and Directed Balanced Edge Deletion, respectively. Cygan et al. show that Undirected Even Edge Deletion can be reduced to Min T-Join, and Directed Balanced Edge Deletion to a minimum cost flow problem with unit costs, both in polynomial time [CMP+14]. Thus it is not the local requirement of even degrees which makes these problems hard, but the simultaneous global requirement of (strong) connectivity.

To handle this situation we turn to a *matroid* which correctly captures the connectivity requirement. Let $\mathcal{I}$ be the family of all subsets $X \subseteq E(G)$ of the edge set of a graph G such that the subgraph $(V(G), E(G) \setminus X)$ is connected. Then the pair $(E(G), \mathcal{I})$ forms a linear matroid called the co-graphic matroid of G (See Section 11.0.1 for definitions). Let T be the set of odd-degree vertices of the input graph G. Observe that for Undirected Eulerian Edge Deletion, the solution S we are after is *both* a T-join *and* an independent set of the co-graphic matroid of G. We exploit this property of S to design a dynamic programming algorithm which finds S by computing "representative sub-families" [FLS14, KW12, Mar09, Mon85] of certain families of edge subsets in the context of the co-graphic matroid of G. We give simple characterizations of solutions which allow us to do dynamic programming, where at every step we only need to keep a representative family of the family of partial solutions where each partial solution is an independent set of the corresponding co-graphic matroid. Our methods also imply that Undirected Connected Odd Edge Deletion admits an algorithm with running time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$.

11.0.1 Notations used in this chapter

For an edge set $E' \subseteq E(G)$, we use (i) $V(E')$ to denote the set of *end vertices* of the edges in E', (ii) $G \setminus E'$ to denote the subgraph $G' = (V(G), E(G) \setminus E')$ of G, and (iii) $G(E')$ to denote the subgraph $(V(E'), E')$ of G. We say that a path system $\mathcal{P} = \{P_1, \ldots, P_r\}$ *ends* at a vertex u if the path P_r ends at u, and u is called the *final vertex* of $\mathcal{P}$. We use $V^e(\mathcal{P})$ to denote the set of end vertices of paths in a path system $\mathcal{P}$. For a path system $\mathcal{P}$ in a digraph D, we use $V^i(\mathcal{P})$ and $V^f(\mathcal{P})$, respectively, to denote the set of initial vertices and the set of final vertices, respectively, of paths in $\mathcal{P}$. For a path system $\mathcal{P} = \{P_1, \ldots, P_r\}$ and an edge/arc (u, v), we define $\mathcal{P} \circ (u, v)$ as follows.

$$\mathcal{P} \circ (u,v) = \begin{cases} \{P_1, \ldots, P_r v\} & \text{if } u \text{ is the final vertex of } P_r \text{ and } v \notin V(P_r) \\ \{P_1, \ldots, P_r, uv\} & \text{if } u \text{ is not the final vertex of } P_r \end{cases}$$

Let $\mathcal{A}$ be a family of path systems in a graph G. Let $e = (u, v)$ be an edge in G (or an arc in D), and let $M = (E, \mathcal{I})$ be the co-graphic matroid of graph G (or of digraph D). We

use $\mathcal{A} \bullet \{e\}$ to denote the family of path systems

$$\mathcal{A} \bullet \{e\} = \left\{ \mathcal{P}' = \mathcal{P} \circ e \mid \mathcal{P} \in \mathcal{A}, e \notin E(\mathcal{P}), E(\mathcal{P}') \in \mathcal{I} \right\}.$$

11.1 Undirected Eulerian Edge Deletion

In this section we describe our $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$-time algorithm for Undirected Eulerian Edge Deletion. Let (G, k) be an instance of the problem. Cygan et al. [CMP+14] observed the following characterization.

Observation 11.1. *A set $S \subseteq E(G)$; $|S| \leq k$ of edges of a graph G is a solution to the instance (G, k) of* Undirected Eulerian Edge Deletion *if and only if it satisfies the following conditions:*

(a) $G \setminus S$ is a connected graph; and,

(b) S is a T-join where T is the set of all odd degree vertices in G.

For a designated set $T \subseteq V(G)$ of *terminal* vertices of graph G, we call a set $S \subseteq E(G)$ a *co-connected T-join* of graph G if (i) $G \setminus S$ is connected and (ii) S is a T-join. From Observation 11.1 we get that the Undirected Eulerian Edge Deletion problem is equivalent to checking whether the given graph G has a co-connected T-join of size at most k, where T is the set of all *odd-degree* vertices in G. We present an algorithm which finds a co-connected T-join for an *arbitrary* (even-sized) set of terminals T within the claimed time-bound. That is, we solve the following more general problem

Co-Connected T-Join	Parameter: k
Input: A connected graph G, an even-sized subset $T \subseteq V(G)$ and an integer k.	
Question: Does there exist a co-connected T-join of G of size at most k?	

We design a dynamic programming algorithm for this problem where the partial solutions which we store satisfy the first property of co-connected T-join and "almost satisfy" the second property. To limit the number of partial solutions which we need to store, we compute and store instead, at each step, a *representative family* of the partial solutions in

the corresponding co-graphic matroid. We start with the following characterization of the T-joins of a graph G.

Proposition 11.2. [Fra93, Proposition 1.1] *Let T be an even-sized subset of vertices of a graph G, and let $\ell = \frac{|T|}{2}$. A subset S of edges of G is a T-join of G if and only if S can be expressed as a union of the edge sets of (i) ℓ paths which connect disjoint pairs of vertices in T, and (ii) zero or more cycles, where the paths and cycles are all pairwise edge-disjoint.*

This proposition yields the following useful property of *inclusion-minimal* co-connected T-joins (*minimal* co-connected T-joins for short) of a graph G.

Lemma 11.3. *Let T be an even-sized subset of vertices of a graph G, and let $\ell = \frac{|T|}{2}$. Let S be a minimal co-connected T-join of G. Then (i) the subgraph $G(S)$ is a forest, and (ii) the set S is a union of the edge-sets of ℓ pairwise edge disjoint paths which connect disjoint pairs of vertices in T.*

Proof. Suppose the subgraph $G(S)$ is not a forest. Then there exists a cycle C in $G(S)$. The degree of any vertex v of G in the subgraph $G(S \setminus E(C))$ is either the same as its degree in the subgraph $G(S)$, or is smaller by exactly two. So the set $S \setminus E(C)$ is also a T-join of G. And since the subgraph $G \setminus S$ is connected by assumption, we get that the strictly larger subgraph $G \setminus (S \setminus E(C))$ is also connected. Thus $S \setminus E(C)$ is a co-connected T-join of G which is a strict subset of S. This contradicts the minimality of S, and hence we get that $G(S)$ is a forest.

Thus there are no cycles in the subgraph $G(S)$, and hence we get from 11.2 that S is a union of the edge sets of ℓ pairwise edge-disjoint paths which connect disjoint pairs of vertices in T. □

Note that the set of paths described in Lemma 11.3 are just pairwise *edge-disjoint*. *Vertices* (including terminals) may appear in more than one path as *internal* vertices. A partial converse of the above lemma follows directly from Proposition 11.2.

Lemma 11.4. *Let T be an even-sized subset of vertices of a graph G, and let $\ell = \frac{|T|}{2}$. Let a subset $S \subseteq E(G)$ of edges of G be such that (i) $G \setminus S$ is connected, and (ii) S is a union*

of the edge-sets of ℓ pairwise edge-disjoint paths which connect disjoint pairs of vertices in T. Then S is a co-connected T-join.

Proof. Since S is a union of the edge sets of ℓ pairwise edge-disjoint paths which connect disjoint pairs of vertices in T, we get from 11.2 that S is a T-join. Since $G \setminus S$ is connected as well, S is a co-connected T-join. □

An immediate corollary of Lemma 11.3 is that for any set $T \subseteq V(G)$, *any* T-join of the graph G has at least $|T|/2$ edges. Hence if $|T| > 2k$ then we can directly return No as the answer for Co-Connected T-Join. So from now on we assume that $|T| \leq 2k$. From Lemmas 11.3 and 11.4 we get that to solve Co-Connected T-Join it is enough to check for the existence of a pairwise edge-disjoint collection of paths $\mathcal{P} = \{P_1, \ldots, P_{\frac{|T|}{2}}\}$ such that (i) the subgraph $(G \setminus E(\mathcal{P}))$ is connected, (ii) $|E(\mathcal{P})| \leq k$, and (iii) the paths in $\mathcal{P}$ connect disjoint pairs of terminals in T. We use dynamic programming to find such a path system.

We first state some notation which we need to describe the dynamic programming table. We use $\mathcal{Q}$ to denote the set of *all* path systems in G which satisfy the above conditions. For $1 \leq i \leq k$ we use $\mathcal{Q}^{(i)}$ to denote the set of all *potential partial* solutions of *size* i : Each $\mathcal{Q}^{(i)}$ is a collection of path systems $\mathcal{Q}^{(i)} = \{\mathcal{P}_1^{(i)}, \ldots, \mathcal{P}_t^{(i)}\}$ where each path system $\mathcal{P}_s^{(i)} = \{P_1, \ldots, P_r\} \in \mathcal{Q}^{(i)}$ has the following properties:

(i) The paths $P_1, \ldots, P_r$ are pairwise edge-disjoint.

(ii) The end-vertices of the paths $P_1, \ldots, P_r$ are all terminals and are pairwise disjoint, *with one possible exception.* One end-vertex (the *final* vertex) of the path P_r may be a non-terminal, or a terminal which appears as an end-vertex of another path as well.

(iii) $|E(\mathcal{P}_s^{(i)})| = i$, and the subgraph $G \setminus E(\mathcal{P}_s^{(i)})$ is connected.

Note that the only ways in which a partial solution $\mathcal{P}_s^{(i)}$ may violate one of the conditions in Lemma 11.4 are: (i) it may contain strictly less than $\frac{|T|}{2}$ paths, and/or (ii) there may be a path P_r (and only one such), which has *one* end-vertex v_r which is a non-terminal or is a

terminal which is an end-vertex of another path as well. For a path system $\mathcal{P} = \{P_1, \ldots, P_r\}$ and $u \in V(G) \cup \{\epsilon\}$, we use $W(\mathcal{P}, u)$ to denote the following set.

$$W(\mathcal{P}, u) = \begin{cases} V^e(\mathcal{P}) & \text{if } u = \epsilon \\ (V^e(\mathcal{P} \setminus \{P_r\})) \cup \{v \,|\, v \text{ is the initial vertex of } P_r\} & \text{if } u \neq \epsilon \end{cases}$$

Finally, for each $1 \leq i \leq k$, $T' \subseteq T$, and $v \in (V(G) \cup \{\epsilon\})$ we define

$$\mathcal{Q}[i, T', v] = \{\mathcal{P} \in \mathcal{Q}^{(i)} \mid W(\mathcal{P}, v) = T', \text{ and if } v \neq \epsilon \text{ then } v \text{ is the final vertex of } \mathcal{P}\}$$

as the set of all potential partial solutions of size i whose set of end vertices is exactly $T' \cup \{v\}$. Observe from this definition that in the case $v = \epsilon$, the last path P_r in each path system $\mathcal{P} = \{P_1, \ldots, P_r\} \in \mathcal{Q}[i, T', \epsilon]$ ends at a "good" vertex; that is, at a terminal vertex which is different from all the end vertices of the other paths $P_1, \ldots, P_{(r-1)}$ in $\mathcal{P}$.

It is not difficult to see that this definition of $\mathcal{Q}[i, T', v]$ is a correct notion of a partial solution for Co-Connected T-Join:

Lemma 11.5. *Let (G, T, k) be a* Yes *instance of* Co-Connected T-Join *which has a minimal solution of size $k' \leq k$, and let $\ell = \frac{|T|}{2}$. Then for each $1 \leq i \leq k'$ there exist $T' \subseteq T$, $v \in (V(G) \cup \{\epsilon\})$, and path systems $\mathcal{P} = \{P_1, P_2, \ldots, P_r\} \in \mathcal{Q}[i, T', v]$ and $\mathcal{P}' = \{P'_r, P'_{r+1}, \ldots, P'_\ell\}$ in G (where $E(P'_r) = \emptyset$ if $v = \epsilon$) such that (i) $E(\mathcal{P}) \cap E(\mathcal{P}') = \emptyset$, (ii) $P_r P'_r$ is a path in G, and (iii) $\mathcal{P} \cup \mathcal{P}' = \{P_1, P_2, \ldots, P_r P'_r, P'_{r+1}, \ldots, P'_\ell\}$ is an edge-disjoint path system whose edge set is a solution to the instance (G, T, k).*

Proof. Let $\widehat{\mathcal{P}} = \{\widehat{P}_1, \ldots, \widehat{P}_\ell\}$ be a path system in graph G which witnesses —as per Lemma 11.3— the fact that (G, T, k) has a solution of size k'. If $i = \sum_{j=1}^{r} |E(\widehat{P}_j)|$ for some $1 \leq r \leq \ell$ then the path systems $\mathcal{P} = \{\widehat{P}_1, \widehat{P}_2, \ldots, \widehat{P}_r\} \in \mathcal{Q}[i, T', v]$ and $\mathcal{P}' = \{\emptyset, \widehat{P}_{r+1}, \widehat{P}_{r+2}, \ldots, \widehat{P}_\ell\}$ satisfy the claim, where $T' = T \cap V^e(\mathcal{P})$ and $v = \epsilon$.

If i takes another value then let $1 \leq r \leq \ell$ be such that $\sum_{j=1}^{r-1} |E(\widehat{P}_j)| < i < \sum_{j=1}^{r} |E(\widehat{P}_j)|$. "Split" the path $\widehat{P}_r$ as $\widehat{P}_r = \widehat{P}^1_r \widehat{P}^2_r$ such that $\sum_{j=1}^{r-1} |E(\widehat{P}_j)| + |E(\widehat{P}^1_r)| = i$. Now the path systems $\mathcal{P} = \{\widehat{P}_1, \widehat{P}_2, \ldots, \widehat{P}_{r-1}, \widehat{P}^1_r\} \in \mathcal{Q}[i, T', v]$ and $\mathcal{P}' = \{\widehat{P}^2_r, \widehat{P}_{r+1}, \widehat{P}_{r+2}, \ldots, \widehat{P}_\ell\}$ satisfy the claim, where $T' = T \cap V^e(\mathcal{P})$ and v is the final vertex of the path $\widehat{P}^1_r$. □

Given this notion of a partial solution the natural dynamic programming approach is to try to compute, in increasing order of $1 \leq i \leq k$, partial solutions $\mathcal{Q}[i, T', v]$ for all $T' \subseteq T$, $v \in (V(G) \cup \{\epsilon\})$ at step i. But this is not feasible in polynomial time because the sets $\mathcal{Q}[i, T', v]$ can potentially grow to sizes exponential in $|V(G)|$. Our way out is to observe that to reach a final solution to the problem we do not need to store *every* element of a set $\mathcal{Q}[i, T', v]$ at each intermediate step. Instead, we only need to store a *representative family* $\mathcal{R}$ of partial solutions corresponding to $\mathcal{Q}[i, T', v]$, where $\mathcal{R}$ has the following property: If there is a way of extending—in the sense of Lemma 11.5—any partial solution $\mathcal{P} \in \mathcal{Q}[i, T', v]$ to a final solution then there exists a $\hat{\mathcal{P}} \in \mathcal{R}$ which can be extended the *same* way to a final solution.

Observe now that our final solution and all partial solutions are independent sets in the co-graphic matroid M_G of the input graph G. We now use Theorem 9.11(ii) to compute these representative families of potential partial solutions at each intermediate step. In step i of the dynamic programming we store, in place of the set $\mathcal{Q}[i, T', v]$, its $(k-i)$-representative set $\widehat{\mathcal{Q}[i, T', v]} \subseteq^{k-i}_{rep} \mathcal{Q}[i, T', v]$ with respect to the co-graphic matroid M_G; for the purpose of this computation we think of each element $\mathcal{P}$ of $\mathcal{Q}[i, T', v]$ as the *edge set* $E(\mathcal{P})$. Lemma 11.6 below shows that this is a safe step. Whenever we talk about representative families in this section, it is always with respect to the co-graphic matroid M_G associated with G; we do not explicitly mention the matroid from now on. We start with the following definitions.

Definition 11.2. *Let $1 \leq i \leq k$, $T' \subseteq T, \ell = \frac{|T|}{2}$ and $v \in (V(G) \cup \{\epsilon\})$, and let $\mathcal{Q}[i, T', v]$ be the corresponding set of partial solutions. Let $\mathcal{P} = \{P_1, \ldots, P_r\}$ be a path system in the set $\mathcal{Q}[i, T', v]$. Let $\mathcal{P}' = \{P'_r, P'_{r+1}, \ldots, P'_\ell\}$ be a path system in G (where $E(P'_r) = \emptyset$ if $v = \epsilon$) such that (i) $|E(\mathcal{P}')| \leq (k-i)$, (ii) $P_r P'_r$ is a path in G, (iii) $\mathcal{P} \cup \mathcal{P}' = \{P_1, P_2, \ldots, P_r P'_r, P'_{r+1}, \ldots, P'_\ell\}$ is an edge-disjoint path system that connects disjoint pairs of terminals in T, (iv) $V^e(\mathcal{P} \cup \mathcal{P}') = T$ and (v) $G \setminus (E(\mathcal{P}) \cup E(\mathcal{P}'))$ is connected. Then $\mathcal{P}'$ is said to be an **extender** for $\mathcal{P}$.*

Definition 11.3. *Let $1 \leq i \leq k$, $T' \subseteq T$ and $v \in (V(G) \cup \{\epsilon\})$, and let $\mathcal{Q}[i, T', v]$ be the corresponding set of partial solutions. We say that $\mathcal{J}[i, T', v] \subseteq \mathcal{Q}[i, T', v]$ is a* **path-system equivalent set** *to $\mathcal{Q}[i, T', v]$ if the following holds: If $\mathcal{P} \in \mathcal{Q}[i, T', v]$ and $\mathcal{P}'$ be*

an extender for $\mathcal{P}$, then there exists $\mathcal{P}^* \in \mathcal{J}[i, T', v]$ such that $\mathcal{P}'$ is an extender for $\mathcal{P}^*$ as well. We say that $\mathcal{J}[i, T', v] \sqsubseteq^{k-i}_{peq} \mathcal{Q}[i, T', v]$.

The next lemma shows that a representative family is indeed a path-system equivalent set to $\mathcal{Q}[i, T', v]$.

Lemma 11.6. *Let (G, T, k) be an instance of* Co-Connected T-Join *such that the smallest co-connected T-join of G has size k and let $\ell = \frac{|T|}{2}$. Let $1 \leq i \leq k$, $T' \subseteq T$ and $v \in (V(G) \cup \{\epsilon\})$, and let $\mathcal{Q}[i, T', v]$ be the corresponding set of partial solutions. If $\widehat{\mathcal{Q}[i, T', v]} \subseteq^{k-i}_{rep} \mathcal{Q}[i, T', v]$, then $\widehat{\mathcal{Q}[i, T', v]} \sqsubseteq^{k-i}_{peq} \mathcal{Q}[i, T', v]$. More generally, if $\mathcal{J}[i, T', v] \subseteq \mathcal{Q}[i, T', v]$ and $\widehat{\mathcal{J}[i, T', v]} \subseteq^{k-i}_{rep} \mathcal{J}[i, T', v]$ then $\widehat{\mathcal{J}[i, T', v]} \sqsubseteq^{k-i}_{rep} \mathcal{J}[i, T', v]$.*

Proof. We first prove the first claim. The second claim of the lemma follows by similar arguments. Let $\widehat{\mathcal{Q}[i, T', v]} \subseteq^{k-i}_{rep} \mathcal{Q}[i, T', v]$, let $\mathcal{P} = \{P_1, \ldots, P_r\}$ be a path system in the set $\mathcal{Q}[i, T', v]$, and let $\mathcal{P}' = \{P'_r, P'_{r+1}, \ldots, P'_\ell\}$ be a path system in G (where $E(P'_r) = \emptyset$ if $v = \epsilon$) which is an extender for $\mathcal{P}$. We have to show that there exists a path system $\mathcal{P}^* \in \widehat{\mathcal{Q}[i, T', v]}$ such that $\mathcal{P}'$ is an extender for $\mathcal{P}^*$ as well. Since $\mathcal{P}'$ is an extender for $\mathcal{P}$ we have, by definition, that (i) $|E(\mathcal{P}')| \leq (k - i)$, (ii) $P_r P'_r$ is a path in G, (iii) $\mathcal{P} \cup \mathcal{P}' = \{P_1, \ldots, P_r P'_r, P'_{r+1}, \ldots, P'_\ell\}$ is an edge-disjoint path system that connects disjoint pairs of terminals in T, (iv) $V^e(\mathcal{P} \cup \mathcal{P}') = T$ and (v) $G \setminus (E(\mathcal{P}) \cup E(\mathcal{P}'))$ is connected.

Since (i) $\mathcal{P} \in \mathcal{Q}[i, T', v]$, (ii) $E(\mathcal{P}) \cap E(\mathcal{P}') = \emptyset$, (iii) $G \setminus (E(\mathcal{P}) \cup E(\mathcal{P}'))$ is connected, and (iv) $\widehat{\mathcal{Q}[i, T', v]} \subseteq^{k-i}_{rep} \mathcal{Q}[i, T', v]$, there exists a path system $\mathcal{P}^* = \{P^*_1, P^*_2, \ldots, P^*_r\}$ in $\widehat{\mathcal{Q}[i, T', v]}$ such that (i) $E(\mathcal{P}^*) \cap E(\mathcal{P}') = \emptyset$ and (ii) $G \setminus (E(\mathcal{P}^*) \cup E(\mathcal{P}'))$ is connected. This follows directly from the definitions of a co-graphic matroid and a representative set.

We now show that $\mathcal{P}'$ is indeed an extender for $\mathcal{P}^*$. Since $\mathcal{P}$ and $\mathcal{P}^*$ both belong to the set $\mathcal{Q}[i, T', v]$ we get that $|E(\mathcal{P})| = |E(\mathcal{P}^*)| = i$ and that $\mathcal{P}^*$ is an edge-disjoint path system. And since $E(\mathcal{P}^*) \cap E(\mathcal{P}') = \emptyset$, we have that $\mathcal{P}^* \cup \mathcal{P}' = \{P^*_1, \ldots, P^*_{r-1}, P^*_r P'_r, P'_{r+1}, \ldots, P'_\ell\}$ is an edge-disjoint path system but for $P^*_r P'_r$ which could be an *Eulerian walk* (walk where vertices could repeat but not the edges). Now we prove that the "path system" $\mathcal{P}^* \cup \mathcal{P}'$ connects disjoint pairs of terminals in T, but for a pair which is connected by an Eulerian walk. We now consider two cases for the "vertex" v.

Case 1: $v = \epsilon$**.** In this case, since $\mathcal{P}$ and $\mathcal{P}^*$ both belong to the set $\mathcal{Q}[i, T', \epsilon]$ we have that $V^e(\mathcal{P}) = V^e(\mathcal{P}^*) = T'$. Also $E(P'_r) = \emptyset$, and $\mathcal{P} \cup \mathcal{P}'$ is the path system $\{P_1, \ldots, P_r, P'_{r+1}, P'_{r+2}, \ldots, P'_\ell\}$ with exactly $\ell = \frac{|T|}{2}$ paths which connect disjoint pairs of terminals in T. Since $V^e(\mathcal{P} \cup \mathcal{P}') = T$, $\mathcal{P} = \{P_1, \ldots, P_r\}$ and $V^e(\mathcal{P}) = T'$, we get that $V^e(\mathcal{P}') = T \setminus T'$. Now since $V^e(\mathcal{P}^*) = T'$ it follows that $\mathcal{P}^* \cup \mathcal{P}'$ is a path system which connects disjoint pairs of terminals in T.

Case 2: $v \neq \epsilon$**.** In this case, since $\mathcal{P}$ and $\mathcal{P}^*$ both belong to the set $\mathcal{Q}[i, T', v]$ we have that $V^e(\mathcal{P}) = V^e(\mathcal{P}^*) = T' \cup \{v\}$, and that the final vertex of each of these two path systems is v. Also $\mathcal{P} \cup \mathcal{P}' = \{P_1, \ldots, P_r P'_r, P'_{r+1}, P'_{r+2}, \ldots, P'_\ell\}$ is a path system with exactly $\ell = \frac{|T|}{2}$ paths which connect disjoint pairs of terminals in T. Since (i) $V^e(\mathcal{P} \cup \mathcal{P}') = T$, (ii) $\mathcal{P} = \{P_1, \ldots, P_r\}$, (iii) $\mathcal{P}' = \{P'_r, P'_{r+1}, \ldots, P'_\ell\}$, (iv) $V^e(\mathcal{P}) = T' \cup \{v\}$, and (v) the final vertex of the path P_r in $\mathcal{P}$ is v, we get that (i) the initial vertex of the path P'_r in $\mathcal{P}'$ is v and (ii) $V^e(\mathcal{P}') = (T \setminus T') \cup \{v\}$. Now since $V^e(\mathcal{P}^*) = T' \cup \{v\}$ and (ii) the final vertex of $\mathcal{P}^*$ is v it follows that $\mathcal{P}^* \cup \mathcal{P}'$ is a path system which connects disjoint pairs of terminals in T, where $P^*_r P'_r$ which could be an Eulerian walk.

Thus, we have shown that $\mathcal{P}^* \cup \mathcal{P}'$ connects disjoint pairs of terminals in T with paths, except for $P^*_r P'_r$ which could be an Eulerian walk. Combining this with Proposition 11.2 and the fact that $G \setminus (E(\mathcal{P}^*) \cup E(\mathcal{P}'))$ is connected, we get that $E(\mathcal{P}^*) \cup E(\mathcal{P}')$ is a co-connected T-join of G.

Finally, we show that $\mathcal{P}^* \cup \mathcal{P}'$ is a path system. Towards this we only need to show that $P^*_r P'_r$ is not an Eulerian walk but a path. Observe that $|E(\mathcal{P}^*) \cup E(\mathcal{P}')| \leq |E(\mathcal{P}^*)| + |E(\mathcal{P}')| \leq k$. However, $E(\mathcal{P}^*) \cup E(\mathcal{P}')$ is a co-connected T-join of G and thus by our assumption, $E(\mathcal{P}^*) \cup E(\mathcal{P}')$ has size exactly k – thus a minimum sized solution. By Lemma 11.3 this implies that $E(\mathcal{P}^*) \cup E(\mathcal{P}')$ is a forest and hence $P^*_r P_r$ is a path in G. This completes the proof. □

For our proofs we also need the transitivity property of the relation $\sqsubseteq^q_{peq}$.

Lemma 11.7. *The relation $\sqsubseteq^q_{peq}$ is transitive.*

Proof. Let $\mathcal{A} \sqsubseteq^q_{peq} \mathcal{B}$ and $\mathcal{B} \sqsubseteq^q_{peq} \mathcal{C}$. We need to show that $\mathcal{A} \sqsubseteq^q_{peq} \mathcal{C}$. Let $\mathcal{P} \in \mathcal{C}$ and $\mathcal{P}'$ be

an extender for $\mathcal{P}$. By the definition of $\mathcal{B} \sqsubseteq^q_{peq} \mathcal{C}$, there exists $\mathcal{P}_b \in \mathcal{B}$ such that $\mathcal{P}'$ is also an extender of $\mathcal{P}_b$. Since $\mathcal{A} \sqsubseteq^q_{peq} \mathcal{B}$, there exists $\mathcal{P}_a \in \mathcal{A}$ such that $\mathcal{P}'$ is also an extender of $\mathcal{P}_a$. This implies $\mathcal{A} \sqsubseteq^q_{peq} \mathcal{C}$. □

Our algorithm is based on dynamic programming and stores a table $\mathcal{D}[i, T', v]$ for all $i \in \{0, \ldots, k\}$, $T' \subseteq T$ and $v \in V(G) \cup \{\epsilon\}$. The idea is that $\mathcal{D}[i, T', v]$ will *store a path-system equivalent set to* $\mathcal{Q}[i, T', v]$. That is, $\mathcal{D}[i, T', v] \sqsubseteq^{k-i}_{peq} \mathcal{Q}[i, T', v]$. The recurrences for dynamic programming is given by the following.

For $i = 0$, we have the following cases.

$$\mathcal{D}[0, T', v] := \begin{cases} \{\emptyset\} & \text{if } T' = \emptyset \text{ and } v = \epsilon \\ \emptyset & \text{otherwise} \end{cases} \tag{11.1}$$

For $i \geq 1$, we have the following cases based on whether $v = \epsilon$ or not.

$$\begin{aligned} \mathcal{D}[i, T', v] := \quad & \Big(\bigcup_{\substack{t \in T' \\ (t,v) \in E(G)}} \mathcal{D}[i-1, T' \setminus \{t\}, \epsilon] \bullet \{(t, v)\} \Big) \bigcup \\ & \Big(\bigcup_{(u,v) \in E(G)} \mathcal{D}[i-1, T', u] \bullet \{(u, v)\} \Big) \end{aligned} \tag{11.2}$$

$$\begin{aligned} \mathcal{D}[i, T', \epsilon] := \quad & \Big(\bigcup_{\substack{t_1, t_2 \in T' \\ (t_1,t_2) \in E(G)}} \mathcal{D}[i-1, T' \setminus \{t_1, t_2\}, \epsilon] \bullet \{(t_1, t_2)\} \Big) \bigcup \\ & \Big(\bigcup_{\substack{t \in T' \\ (u,t) \in E(G)}} \mathcal{D}[i-1, T' \setminus \{t\}, u] \bullet \{(u, t)\} \Big) \end{aligned} \tag{11.3}$$

The next lemma will be used in proving the correctness of the algorithm.

Lemma 11.8. *For all* $i \in \{0, \ldots, k\}, T' \subseteq T, v \in V(G) \cup \{\epsilon\}$, $\mathcal{D}[i, T', v] \sqsubseteq^{k-i}_{peq} \mathcal{Q}[i, T', v]$.

Proof. Let $\mathcal{I}$ denote the family of independent sets in M_G, the co-graphic matroid associated with G. We prove the lemma using induction on i. The base case is $i = 0$. From the definition of $\mathcal{Q}[0, T', v]$, we have that $\mathcal{Q}[0, T', v] = \{\emptyset\}$ if $T' = \emptyset$ and $v = \epsilon$, and

$\mathcal{Q}[0, T', v] = \emptyset$ otherwise.

Now we prove that the claim holds for $i \geq 1$. Let us also assume that by induction hypothesis the claim is true for all $i' < i$. Fix a $T' \subseteq T$, and $v \in V(G) \cup \{\epsilon\}$ and let $\mathcal{Q}[i, T', v]$ be the corresponding set of partial solutions. Let $\mathcal{P} = \{P_1, \ldots, P_r\} \in \mathcal{Q}[i, T', v]$ and $\mathcal{P}' = \{P'_r, P'_{r+1}, \ldots, P'_\ell\}$ be a path system such that $\mathcal{P}'$ is an extender for $\mathcal{P}$. We need to show that there exists a $\mathcal{P}^* \in \mathcal{D}[i, T', v]$ such that $\mathcal{P}'$ is also an extender for $\mathcal{P}^*$.

Case 1: $v \neq \epsilon$**.** Consider the path system $\mathcal{P} = \{P_1, \ldots, P_r\} \in \mathcal{Q}[i, T', v]$. $\mathcal{P}$ has i edges and its set of end-vertices is $T' \cup \{v\}$. Also, its final vertex is v. Let (u, v) be the last edge in path P_r. Let P''_r be the path obtained by deleting edge (u, v) from P_r. More precisely: If P_r has at least two edges then P''_r is the non-empty path obtained by deleting the edge (u, v) and the vertex v *from* P_r, and if (u, v) is the only edge in P_r (in which case $u \in T'$) then $P''_r = \emptyset$. Note that the initial vertex of $P'_r \in \mathcal{P}'$ is v. Let uP'_r be the path obtained by concatenating the path uv and P'_r. Let $\mathcal{P}_1 = \{P_1, \ldots, P''_r\}$ and $\mathcal{P}'_1 = \{uP'_r, P'_{r+1}, \ldots, P'_\ell\}$. Then $\mathcal{P}_1$ has $(i-1)$ edges and $\mathcal{P}'_1$ is an extender for $\mathcal{P}_1$. Now we consider two cases:

(u, v) **is the only edge in** P_r**:** Here $P''_r = \emptyset$ and $u \in T'$; let $t = u$. Note that $\mathcal{P}_1 = \{P_1, \ldots, P_{r-1}\} \in \mathcal{Q}[i-1, T' \setminus \{t\}, \epsilon]$. Hence by induction hypothesis there exists $\mathcal{P}^*_1 \in \mathcal{D}[i-1, T' \setminus \{t\}, \epsilon]$ such that $\mathcal{P}'_1$ is also an extender for $\mathcal{P}^*_1$. Since $\mathcal{P}'_1$ is an extender for $\mathcal{P}^*_1$, $E(\mathcal{P}^*_1) \cup E(\mathcal{P}'_1) \in \mathcal{I}$ (by the definition of extender). This implies that $E(\mathcal{P}^*_1) \cup \{(t, v)\} \in \mathcal{I}$. Since $\mathcal{P}^*_1 \in \mathcal{D}[i-1, T' \setminus \{t\}, \epsilon]$ and $(t, v) \in E(G)$, by Equation 11.2, we get a path system $\mathcal{P}^* \in \mathcal{D}[i, T', v]$ by adding the new path $P_r = tv$ to $\mathcal{P}^*_1$. Since $\mathcal{P}'_1$ is an extender of $\mathcal{P}^*_1$, $\mathcal{P}'$ is an extender of $\mathcal{P}^*$ as well.

(u, v) **is not the only edge in** P_r**:** Here $P''_r \neq \emptyset$, and u is the final vertex in P'_r. Hence $\mathcal{P}_1 = \{P_1, \ldots, P''_r\} \in \mathcal{Q}[i-1, T', u]$. Since $\mathcal{P}'_1$ is an extender for $\mathcal{P}_1$, by induction hypothesis there exists $\mathcal{P}^*_1 \in \mathcal{D}[i-1, T', u]$ such that $\mathcal{P}'_1$ is also an extender for $\mathcal{P}^*_1$. By the definition of extender, we have that $E(\mathcal{P}^*_1) \cup E(\mathcal{P}'_1) \in \mathcal{I}$. This implies that $E(\mathcal{P}^*_1) \cup \{(u, v)\} \in \mathcal{I}$. Since $\mathcal{P}^*_1 \in \mathcal{D}[i-1, T', u]$ and $(u, v) \in E(G)$, by Equation 11.2, we get a path system $\mathcal{P}^* \in \mathcal{D}[i, T', v]$ by adding the new edge $\{(u, v)\}$ to $\mathcal{P}^*_1$. Since $\mathcal{P}'_1$ is an extender of $\mathcal{P}^*_1$, $\mathcal{P}'$ is an extender of $\mathcal{P}^*$ as well.

Case 2: $v = \epsilon$**.** We have that $\mathcal{P} = \{P_1, \ldots, P_r\} \in \mathcal{D}[i, T', \epsilon]$. Then $\mathcal{P}$ has i edges, its set of end-vertices is T', and no end-vertex repeats. Let (u, t) be the last edge in path P_r. Then $t \in T'$. Let P_r'' be the path obtained by deleting edge (u, t) from P_r. More precisely: If P_r has at least two edges then P_r'' is the non-empty path obtained by deleting the edge (u, t) and the vertex t *from* P_r, and if (u, t) is the only edge in P_r then $P_r'' = \emptyset$. Let $\mathcal{P}_1 = \{P_1, \ldots, P_r''\}$ and $\mathcal{P}_1' = \{ut, P_r', P_{r+1}', \ldots, P_\ell'\}$. Then $\mathcal{P}_1$ has $(i-1)$ edges and $\mathcal{P}_1'$ is an extender for $\mathcal{P}_1$. Now we consider two cases:

(u, t) **is the only edge in** P_r**:** Here $P_r'' = \emptyset$, and $\{u, t\} \subseteq T'$. Let $t_1 = u, t_2 = t$. Then $\mathcal{P}_1$ is a path system in $\mathcal{Q}[i-1, T' \setminus \{t_1, t_2\}, \epsilon]$. By induction hypothesis there exists $\mathcal{P}_1^* \in \mathcal{D}[i-1, T' \setminus \{t_1, t_2\}, \epsilon]$ such that $\mathcal{P}_1'$ is also an extender of $\mathcal{P}_1^*$. By the definition of extender, we have that $E(\mathcal{P}_1^*) \cup E(\mathcal{P}_1') \in \mathcal{I}$. This implies that $E(\mathcal{P}_1^*) \cup \{(t_1, t_2)\} \in \mathcal{I}$. Since $\mathcal{P}_1^* \in \mathcal{D}[i-1, T' \setminus \{t_1, t_2\}, \epsilon]$ and $(t_1, t_2) \in E(G)$, by Equation 11.3, we get a path system $\mathcal{P}^* \in \mathcal{D}[i, T', v]$ by adding the new path $t_1 t_2$ to $\mathcal{P}_1^*$. Since $\mathcal{P}_1'$ is an extender of $\mathcal{P}_1^*$, $\mathcal{P}'$ is an extender of $\mathcal{P}^*$ as well.

(u, t) **is not the only edge in** P_r**:** Here $P_r'' \neq \emptyset$, u is the final vertex in P_r''. Then $\mathcal{P}_1 \in \mathcal{Q}[i-1, (T' \setminus t), u]$. By induction hypothesis there exists $\mathcal{P}_1^* \in \mathcal{D}[i-1, (T' \setminus t), u]$ such that $\mathcal{P}_1'$ is also an extender of $\mathcal{P}_1^*$. By the definition of extender, we have that $E(\mathcal{P}_1^*) \cup E(\mathcal{P}_1') \in \mathcal{I}$. This implies that $E(\mathcal{P}_1^*) \cup \{(u, t)\} \in \mathcal{I}$. Since $\mathcal{P}_1^* \in \mathcal{D}[i-1, (T' \setminus t), u]$ and $(u, t) \in E(G)$, by Equation 11.3, we get a path system $\mathcal{P}^* \in \mathcal{D}[i, T', \epsilon]$ by adding the new edge (u, t) to $\mathcal{P}_1^*$. Since $\mathcal{P}_1'$ is an extender of $\mathcal{P}_1^*$, $\mathcal{P}'$ is an extender of $\mathcal{P}^*$ as well.

In both cases above we showed that $\mathcal{D}[i, T', v] \sqsubseteq_{peq}^{k-i} \mathcal{Q}[i, T', v]$. □

Algorithm, Correctness and Running Time.

We now describe the main steps of the algorithm. It finds a smallest sized co-connected T-join (of size at most k) for G. The algorithm iteratively tries to find a solution of size $\frac{|T|}{2} \leq k' \leq k$ and returns a solution corresponding to the smallest k' for which it succeeds; else it returns No. By Lemma 11.6 it is enough, in the dynamic programming (DP) table, to store the representative set $\widehat{\mathcal{Q}[i, T', v]} \subseteq_{rep}^{k-i} \mathcal{Q}[i, T', v]$ instead of the complete set $\mathcal{Q}[i, T', v]$, for all $i \in \{1, 2, \ldots, k\}$, $T' \subseteq T$ and $v \in (V(G) \cup \{\epsilon\})$. In the algorithm we compute and

store the set $\widehat{\mathcal{Q}[i,T',v]}$ in the DP table entry $\mathcal{D}[i,T',v]$. We follow Equations 11.1, 11.2 and 11.3 and fill the table $\mathcal{D}[i,T',v]$. For $i=0$ we use Equation 11.1 and fill the table. After this we compute the values of $\mathcal{D}[i,T',v]$ in increasing order of i from 1 to k. At the i^{th} iteration of the **for** loop, we compute $\mathcal{D}[i,T',v]$ from the DP table entries computed at the previous iteration. Since we need to keep the size of potential partial solutions in check, we compute the representative family $\widehat{\mathcal{D}[i,T',v]}$ for each DP table entry $\mathcal{D}[i,T',v]$ constructed in the i^{th} iteration and then set $\mathcal{D}[i,T',v] \leftarrow \widehat{\mathcal{D}[i,T',v]}$. By the definition of $\mathcal{Q}[i,T,\epsilon]$ and Lemma 11.4, any path system in $\mathcal{D}[i,T,\epsilon]$ is a solution to the instance (G,T,k); we check for such a solution as the last step. This completes the description of the algorithm.

The correctness of the algorithm follows from the following. By Lemma 11.8 we know that $\mathcal{D}[i,T',v] \sqsubseteq^{k-i}_{peq} \mathcal{Q}[i,T',v]$ and by Lemma 11.6 we have that $\widehat{\mathcal{D}[i,T',v]} \sqsubseteq^{k-i}_{peq} \mathcal{D}[i,T',v]$. Thus, by transitivity of $\sqsubseteq^{q}_{peq}$ (by Lemma 11.7) we have that $\widehat{\mathcal{D}[i,T',v]} \sqsubseteq^{k-i}_{peq} \mathcal{Q}[i,T',v]$. This completes the proof of correctness. We now compute an upper bound on the running time of the algorithm.

Lemma 11.9. *The above algorithm runs in time* $\mathcal{O}(2^{(2+\omega)k} \cdot n^2m^3k^5) + m^{\mathcal{O}(1)}$ *where* $n = |V(G)|$ *and* $m = |E(G)|$.

Proof. Let $1 \leq i \leq k$ and $T' \subseteq T$ and $v \in (V(G) \cup \{\epsilon\})$ be fixed, and let us consider the running time of computing $\widehat{\mathcal{D}[i,T',v]}$. That is, the running time to compute $(k-i)$-representative family of $\mathcal{D}[i,T',v]$. We know that the co-graphic matroid M_G is representable over $\mathbb{F}_2$ and that its rank is bounded by $m-n+1$. By Theorem 9.11, the running time of this computation of the $(k-i)$-representative family is bounded by $\mathcal{O}\left(\binom{k}{i} \cdot |\mathcal{D}[i,T',v]|i^3m^2 + |\mathcal{D}[i,T',v]| \cdot \binom{k}{i}^{\omega-1}(i \cdot m)^{\omega-1}\right) + m^{\mathcal{O}(1)}$.

The family $\mathcal{D}[i,T',v]$ is computed using Equation 11.2 or Equation 11.3 from the DP table entries $\mathcal{D}[i-1,T'',u]$, computed in the previous iteration and the size of $\mathcal{D}[i-1,T'',u]$ is bounded according to Theorem 9.11. Thus the size of the family $\mathcal{D}[i,T',v]$ is upper bounded by, $|\mathcal{D}[i,T',v]| \leq ((2k)^2 + 2kn) \cdot \left(\mathsf{max}_{T'' \subseteq T', u \in V} \mathcal{D}[i-1,\widehat{T''},u]\right)$. Theorem 9.11 gives bounds on the sizes of these representative families $\mathcal{D}[i-1,\widehat{T''},u]$, from which we get $|\mathcal{D}[i,T',v]| \leq 4kn \cdot mi\binom{k}{i-1}$. Observe that since the number choices for (T',v) such that $T' \subseteq T$ and $v \in V(G)\{\epsilon\}$ is bounded by $4^k(n+1)$, and we compute DP table entries for

$i = 1$ to k, the overall running time can be bounded by $\mathcal{O}\Big(4^k n \sum_{i=1}^{k} \Big(\binom{k}{i} \cdot \binom{k}{i-1} kni^4 m^3 + \binom{k}{i-1} \cdot \binom{k}{i}^{\omega-1} kn(im)^\omega\Big)\Big) + m^{\mathcal{O}(1)}$. The running time above simplifies to $\mathcal{O}(2^{(2+\omega)k} \cdot n^2 m^3 k^5) + m^{\mathcal{O}(1)}$. □

Putting all these together we get

Theorem 11.4. Co-Connected T-Join *can be solved in* $\mathcal{O}(2^{(2+\omega)k} \cdot n^2 m^3 k^6) + m^{\mathcal{O}(1)}$ *time where* $n = |V(G)|$ *and* $m = |E(G)|$.

Using Theorem 11.1 and Theorem 11.4 we get

Theorem 11.5. Undirected Eulerian Edge Deletion *can be solved in time* $\mathcal{O}(2^{(2+\omega)k} \cdot n^2 m^3 k^6) + m^{\mathcal{O}(1)}$ *where* $n = |V(G)|$ *and* $m = |E(G)|$.

11.2 Directed Eulerian Edge Deletion

Directed Eulerian Edge Deletion In this section we modify the algorithm described for Undirected Eulerian Edge Deletion to solve the directed version of the problem. The main ingredient of the proof is the characterization of "solution" for the directed version of the problem. We begin with a few definitions. For a digraph D, we call $S \subseteq A(D)$ a *balanced arc deletion set*, if $D \setminus S$ is balanced. We call a set $S \subseteq A(D)$ a *co-connected balanced arc deletion set* if $D \setminus S$ is balanced and weakly connected.

Let (D, k) be an instance to Directed Eulerian Edge Deletion. A solution $S \subseteq A(D)$ of the problem should satisfy the following two properties, (a) S must be a balanced arc deletion set of D and, (b) $D \setminus S$ must be strongly connected. In fact, something stronger is known in the literature.

Proposition 11.10. [BJG08] *A digraph D is Eulerian if and only if D is weakly connected and balanced.*

Due to Proposition 11.10, we can relax the property (b) of the solution S and replace the requirement of having $D \setminus S$ as strongly connected with just requiring $D \setminus S$ to be be weakly connected. Now observe that solution S of Directed Eulerian Edge Deletion

is in fact a co-connected balanced arc deletion set of the directed graph D. Thus our goal is to compute a minimal co-connected balanced arc deletion set of D of size at most k.

We start with the following easy property of in-degrees and out-degrees of vertices in D. For a digraph D, define $\mathcal{T}^- = \{v \in V(D) \mid d_D^-(v) > d_D^+(v)\}$, $\mathcal{T}^= = \{v \in V(D) \mid d_D^-(v) = d_D^+(v)\}$ and $\mathcal{T}^+ = \{v \in V(D) \mid d_D^-(v) < d_D^+(v)\}$.

Proposition 11.11. *In a digraph D, $\sum_{v \in \mathcal{T}^+} d_D^+(v) - d_D^-(v) = \sum_{v \in \mathcal{T}^-} d_D^-(v) - d_D^+(v)$.*

Proof. In any digraph we have the following statement,

$$\begin{aligned} & \sum_{v \in V(D)} d^+(v) && = \sum_{v \in V(D)} d^-(v) \\ \iff & \sum_{v \in \mathcal{T}^+} d^+(v) + \sum_{v \in \mathcal{T}^-} d^+(v) && = \sum_{v \in \mathcal{T}^-} d^-(v) + \sum_{v \in \mathcal{T}^+} d^-(v) \\ \iff & \sum_{v \in \mathcal{T}^+} d_D^+(v) - d_D^-(v) && = \sum_{v \in \mathcal{T}^-} d_D^-(v) - d_D^+(v). \end{aligned}$$

This completes the proof. □

The following lemma characterizes the set of arcs which form a minimal solution S of the given instance (D, k). We then use this characterization to design a dynamic-programming algorithm for the problem.

Lemma 11.12. *Let D be a digraph, and $\ell = \sum_{v \in \mathcal{T}^+} d_D^+(v) - d_D^-(v)$. Let $S \subseteq A(D)$ be a minimal co-connected balanced arc deletion set. Then S is a union of ℓ arc disjoint paths $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ such that*

(1) For $i \in \{1, \ldots, \ell\}$, P_i starts at a vertex in $\mathcal{T}^+$ and ends at a vertex in $\mathcal{T}^-$.

(2) The number of paths in $\mathcal{P}$ that starts at $v \in \mathcal{T}^+$ is equal to $d_D^+(v) - d_D^-(v)$ and the number of paths in $\mathcal{P}$ that ends at $u \in \mathcal{T}^-$ is equal to $d_D^-(u) - d_D^+(u)$.

Proof. First we claim that $D(S)$ is a directed acyclic digraph. Suppose not, then let C be a directed cycle in $D(S)$. The in-degree and out-degree of any vertex of v of D in $D(S \setminus A(C))$ is either same as its in-degree and out-degree in the subgraph $D(S)$ or both in-degree and

out-degree of v is smaller by exactly one. So $S \setminus A(C)$ is a balanced arc deletion set of D. And since the subgraph $D \setminus S$ is connected by assumption, we get that the strictly larger subgraph $D \setminus (S \setminus A(C))$ is also connected. Thus $S \setminus A(C)$ is a co-connected balanced arc deletion set of D. This contradicts the fact that S is minimal, and hence we get that $D(S)$ a directed acyclic digraph.

We prove the lemma using induction on ℓ. When $\ell = 0$, the lemma holds vacuously. Now consider the induction step, i.e, when $\ell > 0$. Consider a *maximal* path P in $D(S)$. We claim that P starts at a vertex in $\mathcal{T}^+$. Suppose not, let P starts at $w \in V(D) \setminus \mathcal{T}^+$. Further, let (w, x) be the arc of P that is incident on w. By our assumption $w \in \mathcal{T}^- \cup \mathcal{T}^=$, which implies that if $(w, x) \in S$, then there must exist an arc $(y, w) \in S$ or else the vertex w cannot be balanced in $D \setminus S$. And since $D(S)$ is a directed acyclic digraph we have that $y \notin V(P)$. But this contradicts the assumption that P is a maximal path in $D(S)$. By similar arguments we can prove that P ends at a vertex in $\mathcal{T}^-$. Let P starts at t_1 and ends at t_2, where $t_1 \in \mathcal{T}^+$ and $t_2 \in \mathcal{T}^-$. Now consider the digraph $D' = D \setminus A(P)$. Clearly, $S \setminus A(P)$ is a minimal co-connected balanced arc deletion set for the digraph D'. We claim the following

$$\sum_{\{v \in V(D') | d^+_{D'}(v) > d^-_{D'}(v)\}} d^+_{D'}(v) - d^-_{D'}(v) = \ell - 1.$$

The correctness of this follows from the fact that the difference $d^+_{D'}(v) - d^-_{D'}(v) = d^+_D(v) - d^-_D(v)$ for all $v \in V(D) \setminus \{t_1, t_2\}$. And for t_1 we have that $d^+_{D'}(t_1) - d^-_{D'}(t_1) = d^+_D(t_1) - d^-_D(t_1) - 1$.

Now by applying induction hypothesis on D' with $\ell - 1$ we have that $S \setminus A(P)$ is a union of $\ell - 1$ arc disjoint paths $P_1, \ldots, P_{\ell-1}$ which satisfies properties (1) and (2) for the digraph D'. Now consider the path system $P, P_1, \ldots, P_{\ell-1}$ and observe that it indeed satisfies properties (1) and (2). This concludes the proof. □

Finally, we prove a kind of "converse" of Lemma 11.12.

Lemma 11.13. *Let D be a digraph, $\ell = \sum_{v \in \mathcal{T}^+} d^+_D(v) - d^-_D(v)$ and let $S \subseteq A(D)$. Furthermore, S is a union of ℓ arc disjoint paths $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ with the following properties.*

1. *The digraph $D \setminus S$ is weakly connected.*
2. *For $i \in \{1, \ldots, \ell\}$, P_i starts at a vertex in $\mathcal{T}^+$ and ends at a vertex in $\mathcal{T}^-$.*
3. *The number of paths in $\mathcal{P}$ that starts at $v \in \mathcal{T}^+$ is equal to $d^+_D(v) - d^-_D(v)$ and the number of paths in $\mathcal{P}$ that ends at $u \in \mathcal{T}^-$ is equal to $d^-_D(u) - d^+_D(u)$.*

Then S is a co-connected balanced arc deletion set.

Proof. By property (2), each vertex $v \in \mathcal{T}^=$ appears only as internal vertex of any path in $\mathcal{P}$. Therefore v is balanced in $D \setminus S$. By property (3), for every vertex $t \in \mathcal{T}^+$, exactly $d^+_D(t) - d^-_D(t)$ paths start at t in $\mathcal{P}$ and no path in $\mathcal{P}$ ends at t and thus t is balanced in $D \setminus S$. Similar arguments hold for all $t \in \mathcal{T}^-$. Hence $D \setminus S$ is balanced. Since $D \setminus S$ is weakly connected as well by property (1), we have that S is a co-connected balanced arc deletion set of D. □

Now we are ready to describe the algorithm for Directed Eulerian Edge Deletion. Let (D, k) be an instance of the problem. Lemma 11.12 and Lemma 11.13 imply that for a solution we can seek a path system $\mathcal{P}$ with properties mentioned in Lemma 11.13. Let $\mathcal{T}^+_m$ be the multiset of vertices in the graph G such that each vertex $v \in \mathcal{T}^+$ appears $d^+_D(v) - d^-_D(v)$ times in $\mathcal{T}^+_m$. Similarly, let $\mathcal{T}^-_m$ be the multiset of vertices in the graph D such that each vertex $v \in \mathcal{T}^-$ appears $d^-_D(v) - d^+_D(v)$ times in $\mathcal{T}^-_m$. Due to Proposition 11.11 we know that $|\mathcal{T}^+_m| = |\mathcal{T}^-_m|$. Observe that if $|\mathcal{T}^+_m| > k$, then any balanced arc deletion set must contain more than k arcs and thus the given instance is a No instance. So we assume that $|\mathcal{T}^+_m| \leq k$.

Lemma 11.12 implies that the solution can be thought of as a path system $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ connecting vertices from $\mathcal{T}^+_m$ to the vertices of $\mathcal{T}^-_m$ such that all the vertices of $\mathcal{T}^+_m \cup \mathcal{T}^-_m$ appear as end points exactly once and $D \setminus A(\mathcal{P})$ is weakly connected. Observe that the solution is a path system with properties which are similar to those in the undirected case of the problem. Indeed, the solution S corresponds to an independent set in the co-graphic matroid of the underlying (undirected) graph of D. After this the algorithm for Directed Eulerian Edge Deletion is identical to the algorithm for Co-Connected T-Join. Let $T = \mathcal{T}^+_m \cup \mathcal{T}^-_m$. We can define a notion of partial solutions analogous to $\mathcal{Q}[i, T', v]$. The

definition of *extender* remains the same except for the last item, where we now require that $\mathcal{P} \cup \mathcal{P}'$ is an arc disjoint path system connecting vertices from $\mathcal{T}_m^+$ to the vertices of $\mathcal{T}_m^-$ such that every vertex of $\mathcal{T}_m^+ \cup \mathcal{T}_m^-$ is an endpoint of exactly one path. Finally, we can define the recurrences for dynamic programming similar to those defined for $\mathcal{D}[i, T', v]$ in the case of Co-Connected T-Join. We then use these recurrences along with an algorithm to compute representative families to solve the given instance. The correctness of the algorithm follows via similar arguments as before. And by an analysis similar to the case of Co-Connected T-Join we can obtain the following bound on the running time of the algorithm.

Theorem 11.6. Directed Eulerian Edge Deletion *can be solved in time* $\mathcal{O}(2^{(2+\omega)k} \cdot n^2 m^3 k^6) + m^{\mathcal{O}(1)}$ *where where* $n = |V(D)|$ *and* $m = |A(D)|$.

Chapter 12

Fast Exact Algorithms for Survivable Network Design with Uniform Requirements

In this chapter we consider the problem of designing an exact algorithm for finding a minimum weight spanning subgraph of a given graph or digraph which is λ-edge connected. This is also called the *survivable network design* with *uniform requirements* problem.

Minimum Weight λ-connected Spanning Subgraph
Input: A graph G (or digraph D), an integer λ and a weight function w on the edges(or the arcs)
Question: Find a spanning λ-connected subgraph of minimum total weight.

Observe that such a subgraph contains at most $\lambda(n-1)$ edges (for digraphs, $2\lambda(n-1)$ arcs). Hence a solution can be obtained by enumerating all possible subgraphs with at most these many edges and testing if it is λ connected. However such an algorithm will take $2^{\mathcal{O}(\lambda n(\log n+\log \lambda))}$ time. One may try a more clever approach by observing that, we can enumerate every minimal λ-connected graph on n vertices in $2^{\mathcal{O}(\lambda n)}$ time. Then we test if any of these graph is isomorphic to a subgraph of the input graph. However, subgraph isomorphism requires $2^{\lambda n(\log n+\log \lambda)}$ unless the Exponential Time Hypothesis fails

[CFG+16], and hence this approach fails as well. In this chapter, we give the first single exponential algorithm for this problem that runs in time $2^{\mathcal{O}(\lambda n)}$. As a corollary, we also obtain single exponential time algorithm for the minimum augmentation problem.

Minimum Weight λ-connectivity Augmentation
Input: A graph G (or a digraph D), a set of links $L \subseteq V \times V$ (ordered pairs in case of digraphs), and a weight function $w : L \to \mathbb{N}$.
Question: Find a minimum weight subset L' of L such that $G \cup L$ (or $D \cup L$) is λ-connected.

The first exact algorithm for MEG and MSCSS, running in time $2^{\mathcal{O}(m)}$ time, where m is the number of edges in the graph, was given by Moyles and Thompson [MT69] in 1969. Only recently, Fomin et. al. gave the first single-exponential algorithm for MEG and MSCSS, i.e. with a running time of $2^{\mathcal{O}(n)}$ [FLS14]. For the special case of Hamiltonian Cycle, a $\mathcal{O}(2^n)$ algorithm is known [HK62, Bel62] for general digraphs from 1960s. It was recently improved to $\mathcal{O}(1.657^n)$ for graphs [Bjo14], and to $\mathcal{O}(1.888^n)$ for bipartite di-graphs [CKN13]. For other results and more details we refer to Chapter 12 of [BJG08] and the works mentioned above.

Our approach is similar to that of Fomin et. al. for finding a Minimum equivalent Graph [FLS14], although the algorithm becomes more involved. It is based on the following fact. A digraph D is λ-connected if and only if for any $r \in V(D)$, there is a collection $\mathbb{I}$ of λ arc-disjoint in-branchings rooted at r and a collection $\mathbb{O}$ of λ arc-disjoint out-branchings rooted at r. Then computing a $\mathbb{I}$ and a $\mathbb{O}$ with the largest possible intersection yields a minimum spanning subgraph which is λ-connected. We show that the solution is embedded in a linear matroid of rank $\mathcal{O}(\lambda n)$, and then compute the solution by a dynamic programming algorithm with representative sets over this matroid.

Theorem 12.1. *Let D be a λ edge connected digraph on n vertices and $w : A(D) \to \mathbb{N}$. Then we can find a minimum weight λ edge connected subgraph of D in $2^{\mathcal{O}(\lambda n)}$.*

For the case of undirected graphs, no equivalent characterization is known. However, we obtain a characterization by converting the graph to a digraph with labels on the arcs.

Then by a similar approach we obtain the following.

Theorem 12.2. *Let G be a λ edge connected graph on n vertices and $w : E(G) \to \mathbb{N}$. Then we can find a minimum weight λ edge connected subgraph of G in $2^{\mathcal{O}(\lambda n)}$ time.*

For the problem of augmenting a network to a given connectivity requirement at a minimum cost, we obtain the following results by applying the previous theorems with suitable weight functions.

Theorem 12.3. *Let D be a digraph (or a graph) on n vertices, $L \subseteq V(D) \times V(D)$ be a collection of links with weight function $w : L \to \mathbb{N}$. For any integer λ, we can find a $L' \subseteq L$ of minimum total weight such that $D' = (V(D), A(D) \cup L')$ is λ edge connected, in time $2^{\mathcal{O}(\lambda n)}$.*

12.1 Directed Graphs

In this section, we give a single exponential exact algorithm, that is of running time $2^{\mathcal{O}(\lambda n)}$, for computing a minimum spanning λ-connected subgraph of an input digraph on n vertices. We first consider the unweighted version of the problem and it will be clear that, essentially the same algorithm works for weighted version as well. We begin with the following characterization of λ connectivity in digraphs.

Lemma 12.1. *Let D be a digraph. Then D is λ connected if and only if for any $r \in V(D)$, there is a collection of λ arc-disjoint in-branchings rooted at r, and a collection of λ arc-disjoint out-branchings rooted at r.*

Proof. Fix an arbitrary vertex $r \in V(D)$. In the forward direction, by Edmond's disjoint out-branching theorem [Sch03, Corollary 53.1b], there is a collection of λ arc-disjoint out-branching rooted at a vertex r. To find the collection of in-branchings, we consider the graph D' which is the "reversed digraph" of D, i.e. $V(D') = V(D)$ and the arc set $A(D') = \{(v, u) | (u, v) \in A(D)\}$. Note that the digraph D' is also λ connected, and therefore there is a collection of λ arc-disjoint out-branchings $O'_1, O'_2, \ldots, O'_\lambda$ rooted at r in D'. From this collection, we construct the λ arc-disjoint in-branchings in D as follows. For

each $j \in [\lambda]$, let $I_j = \{(u,v)|(v,u) \in A(O'_j)\}$ and note that this is an in-branching rooted at r. This gives the required collection of in-branchings.

The reverse direction is easy to see, and it follows from [Sch03, Corollary 53.1b and Corollary 53.1d]. □

Let D be the input to our algorithm, which is a λ connected digraph on n vertices. Let us fix a vertex $r \in V(D)$. By Lemma 12.1, any minimal λ connected subgraph of D is a union of a collection $\mathbb{I}$ of λ arc-disjoint in-branchings and a collection $\mathbb{O}$ of λ arc-disjoint out-branchings which are all rooted at vertex r. The following lemma relates the size of such a minimal subgraph to the number of arcs common to both $\mathbb{I}$ and $\mathbb{O}$. It follows easily from Lemma 12.1. Here, $A(\mathbb{I})$ denotes the set of arcs which are present in some $I \in \mathbb{I}$ and $A(\mathbb{O})$ denotes the set of arcs which are present in some $O \in \mathbb{O}$. Observe that the set of arcs in $A(\mathbb{I})$, and the set of arcs in $A(\mathbb{O})$, that are incident on r, are disjoint. Hence the number of common arcs is at most $\lambda(n-2)$.

Lemma 12.2. *Let D be a λ-connected digraph, r be a vertex in $V(D)$ and $\ell \in [\lambda(n-2)]$. Then a subdigraph D' with at most $2\lambda(n-1) - \ell$ arcs, is a minimal λ-connected spanning subdigraph of D if and only if D' is a union of a collection $\mathbb{I}$ of arc-disjoint in-branchings rooted at r, and a collection $\mathbb{O}$ of arc-disjoint out-branchings rooted at r such that $|A(\mathbb{I}) \cap A(\mathbb{O})| \geq \ell$ (i.e. they have at least ℓ common arcs).*

By Lemma 12.2, a minimum λ connected subgraph of D' is $\mathbb{I} \cup \mathbb{O}$, where $\mathbb{O} = \{O_1, O_2, \ldots O_\lambda\}$ is a collection of λ arc disjoint out-branchings rooted at r, and $\mathbb{I} = \{I_1, I_2, \ldots I_\lambda\}$ is a collection of λ arc disjoint in-branchings rooted at r, such that $A(\mathbb{O}) \cap A(\mathbb{I})$ is maximized. Let us assume that the number of arcs in the subdigraph D' is $2\lambda(n-1) - \ell$ where $|A(\mathbb{O}) \cap A(\mathbb{I})| = \ell$. The first step of our algorithm is to construct the set $A(\mathbb{O}) \cap A(\mathbb{I})$, and then, given the intersection, we construct $\mathbb{I}$ and $\mathbb{O}$ in polynomial time. The main idea is to enumerate a subset of potential candidates for the intersection, via dynamic programming. But note that there could be as many as $n^{\mathcal{O}(\lambda n)}$ such candidates, and enumerating all of them will violate the claimed running time. So we try a different approach. We first observe that the solution, $\mathbb{I} \cup \mathbb{O}$, may be embedded into a linear matroid of rank $\mathcal{O}(\lambda n)$. Then we prove that it is enough to keep a representative family of the partial solutions, at

each step, in the dynamic programming table. Since, the size of the representative family is bounded by $2^{\mathcal{O}(\lambda n)}$, our algorithm runs in the claimed running time.

Let us delve into the details of the algorithm. Let D_r^- be the digraph obtained from D after removing the arcs in $\mathsf{Out}_D(r)$. Similarly, let D_r^+ be the digraph obtained from D after removing the arcs in $\mathsf{In}_D(r)$. Observe that the arc sets of D_r^- and D_r^+ can be partitioned as follows.

$$A(D_r^-) = \biguplus_{v \in V(D_r^-)} \mathsf{Out}_{D_r^-}(v) \qquad \text{and} \qquad A(D_r^+) = \biguplus_{v \in V(D_r^+)} \mathsf{In}_{D_r^+}(v)$$

We construct a pair of matroids corresponding to each of the λ in-branching in $\mathbb{I}$ and each of the λ out-branching in $\mathbb{O}$. For each in-branching $I_i \in \mathbb{I}$, we have a matroid $\mathcal{M}^i_{I,1} = (E^i_{I,1}, \mathcal{I}^i_{I,1})$ which is a graphic matroid in D and $E^i_{I,1}$ is a copy of the arc set of D. And similarly, for each out-branching $O_i \in \mathbb{O}$, we have a matroid $\mathcal{M}^i_{O,1} = (E^i_{O,1}, \mathcal{I}^i_{O,1})$ which is a graphic matroid in D and $E^i_{O,1}$ is again a copy of the arc set of D. Note that the rank of these graphic matroids is $n-1$. Next, for each I_i, we define matroid $\mathcal{M}^i_{I,2} = (E^i_{I,2}, \mathcal{I}^i_{I,2})$ which is a partition matroid where $E^i_{I,2}$ is a copy of the arc set of D_r^- and

$$\mathcal{I}^i_{I,2} = \{X \mid X \subseteq E^i_{I,2}, |X \cap \mathsf{Out}_{D_r^-}(v)| \leq 1, \textit{ for all } v \in V(D_r^-)\}$$ [1]

Since $\mathsf{Out}_{D_r^-}(r) = \emptyset$ and $|V(D_r^-)| = n$, we have that the rank of these partition matroids, $\mathcal{M}^i_{I,2}$, $i \in [\lambda]$, is $n-1$. And similarly, for each O_i, we define $\mathcal{M}^i_{O,2} = (E^i_{O,2}, \mathcal{I}^i_{O,2})$ as the partition matroid, where $E^i_{O,2}$ is a copy of the arc set of D_r^+ and

$$\mathcal{I}^i_{O,2} = \{X \mid X \subseteq E^i_{O,2}, |X \cap \mathsf{In}_{D_r^+}(v)| \leq 1, \textit{ for all } v \in V(D_r^+)\}$$

Since $\mathsf{In}_{D_r^+}(r) = \emptyset$ and $V(D_r^+) = n$, we have that the rank of these partition matroids, $\mathcal{M}^i_{O,2}$, $i \in [\lambda]$, is $n-1$. Next, we define two uniform matroids $\mathcal{M}_I$ and $\mathcal{M}_O$ of rank $\lambda(n-1)$, corresponding to $\mathbb{I}$ and $\mathbb{O}$, on the ground sets E_I and E_O, respectively, where E_i and E_O are copies of the arc set of D. We define matroid $\mathcal{M} = (E, \mathcal{I})$ as the direct sum of

[1] We slightly abuse notation for the sake of clarity, as strictly speaking X and $\mathsf{Out}_{D_G^r}(v)$ are disjoint, since they are subsets of two different copies of the arc set.

$\mathcal{M}_I, \mathcal{M}_O$, $\mathcal{M}^i_{I,j}, \mathcal{M}^i_{O,j}$, for $i \in [\lambda]$ and $j \in [2]$, That is,

$$\mathcal{M} = \Big(\bigoplus_{i\in[\lambda], j\in[2]} (\mathcal{M}^i_{I,j} \oplus \mathcal{M}^i_{O,j}) \Big) \oplus \mathcal{M}_I \oplus \mathcal{M}_O$$

Since the rank of $\mathcal{M}^i_{I,j}, \mathcal{M}^i_{O,j}$ where $i \in [\lambda]$ and $j \in [2]$, are $n-1$ each, and rank of $\mathcal{M}_I$ and $\mathcal{M}_O$ is $\lambda(n-1)$, we have that the rank of $\mathcal{M}$ is $6\lambda(n-1)$.

Let us briefly discuss the representation of these matroids. The matroids $\mathcal{M}^i_{I,1}, \mathcal{M}^i_{O,1}$ for $i \in [\lambda]$ are graphic matroids, which are representable over any field of size at least 2. And the matroids $\mathcal{M}^i_{I,2}, \mathcal{M}^i_{O,1}$ are partition matroids with partition size 1, and therefore they are representable over any field of at least 2 as well. Finally, the two uniform matroids, $\mathcal{M}_I$ and $\mathcal{M}_O$, are representable over any field with at least $|A(D)|+1$ elements. Hence, at the start of our algorithm, we choose a representation of all these matroids over a field $\mathbb{F}$ of size at least $|A(D)|+1$. So $\mathcal{M}$ is representable over any field of size at least $|A(D)|+1$, as it is a direct sum of all these matroids.

For an arc $e \in A(D)$ not incident to r there are $4\lambda+2$ copies of it in $\mathcal{M}$. Let $e^i_{J,j}$ denotes it's copy in $E^i_{J,j}$, where $i \in [\lambda]$, $j \in [2]$ and $J \in \{I, O\}$. An arc incident to r has only $3\lambda+2$ copies in $\mathcal{M}$. For an arc $e \in \mathsf{In}_D(r)$ we will denote its copies in $E^i_{I,1}, E^i_{O,1}, E^i_{I,2}$ by $e^i_{I,1}, e^i_{O,1}, e^i_{I,2}$, and similarly for an arc $e \in \mathsf{Out}_D(r)$ we will denote its copies in $E^i_{I,1}, E^i_{O,1}, E^i_{O,2}$ by $e^i_{I,1}, e^i_{O,1}, e^i_{O,2}$. And finally, for any arc $e \in A(D)$, let e_I and e_O denote it's copies in E_I and E_O, respectively. For $e \in A(D) \setminus \mathsf{Out}_D(r)$ and $i \in [\lambda]$, let $S^i_{I,e} = \{e^i_{I,1}, e^i_{I,2}\}$. Similarly for $e \in A(D) \setminus \mathsf{In}_D(r), i \in [\lambda]$, let $S^i_{O,e} = \{e^i_{O,1}, e^i_{O,2}\}$. Let $S_e = (\cup_{i=1}^{\lambda} S^i_{I,e}) \bigcup (\cup_{j=1}^{\lambda} S^j_{O,e}) \bigcup \{e_I, e_O\}$. For $X \in \mathcal{I}$, let A_X denote the set of arcs $e \in A(D)$ such that $S_e \cap X \neq \emptyset$.

Observation 12.3. *Let I be an in-branching in D rooted at r. Then for any $i \in [\lambda]$, $\{e^i_{I,1} \mid e \in A(I)\}$ is a basis in $\mathcal{M}^i_{I,1}$ and $\{e^i_{I,2} \mid e \in A(I)\}$ is a basis in $\mathcal{M}^i_{I,2}$. And conversely, Let X and Y be basis of $\mathcal{M}^i_{I,1}$ and $\mathcal{M}^i_{I,2}$, respectively, such that $A_X = A_Y$. Then A_X is an in-branching rooted at r in D.*

Observation 12.4. *Let O be an out-branching in D. Then for any $i \in [\lambda]$, $\{e^i_{O,1} \mid e \in A(O)\}$ is a basis in $\mathcal{M}^i_{O,1}$ and $\{e^i_{O,2} \mid e \in A(O)\}$ is a basis in $\mathcal{M}^i_{O,2}$. And conversely, Let*

X and Y be basis of $\mathcal{M}^i_{O,1}$ and $\mathcal{M}^i_{O,2}$, respectively, such that $A_X = A_Y$. Then A_X is an out-branching rooted at r in D.

Observe that any arc $e \in A(D)$ can belong to at most one in-branching in $\mathbb{I}$ and at most one out-branching in $\mathbb{O}$, because both $\mathbb{I}$ and $\mathbb{O}$ are collection of arc disjoint subgraphs of D. Because of Observation 12.3 and 12.4, if we consider that each $I_i \in \mathbb{I}$ is embedded into $\mathcal{M}^i_{I,1}$ and $\mathcal{M}^i_{I,2}$ and each $O_i \in \mathbb{O}$ is embedded into $\mathcal{M}^i_{O,1}$ and $\mathcal{M}^i_{O,2}$, then we obtain an independent set Z' of rank $4\lambda(n-1)$ corresponding to $\mathbb{I} \cup \mathbb{O}$ in the matroid $\mathcal{M}$. Further, since the collection $\mathbb{I}$ is arc-disjoint, $\{e_I \mid e \in A(\mathbb{I})\}$ is a basis of $\mathcal{M}_I$. And similarly, $\{e_O \mid e \in A(\mathbb{O})\}$ is a basis of $\mathcal{M}_O$. Therefore, $Z = Z' \cup \{e_I \mid e \in A(\mathbb{I})\} \cup \{e_O \mid e \in A(\mathbb{O})\}$ is a basis of $\mathcal{M}$.

Now observe that, each arc in the intersection $\mathbb{I} \cap \mathbb{O}$ has six copies in the independent set Z. The remaining arcs in $\mathbb{I} \cup \mathbb{O}$ have only three copies each, and this includes any arc which is incident on r. Now, we define a function $\phi : \mathcal{I} \times A(D) \to \{0,1\}$, where for $W \in \mathcal{I}$ and $e \in A(D)$, $\phi(W,e) = 1$ if and only if exactly one of the following holds.

- Either, $W \cap S_e = \emptyset$.
- Or, $\{e_I, e_O\} \subseteq W$ and there exists $t, t' \in [\lambda]$ such that $S^t_{I,e} \subseteq W$ and $S^{t'}_{O,e} \subseteq W$. And for each $i \in [\lambda] \setminus \{t\}$ and $j \in [\lambda] \setminus \{t'\}$, $S^i_{I,e} \cap W = \emptyset$ and $S^j_{O,e} \cap W = \emptyset$.

Using function ϕ we define the following collection of independent sets of $\mathcal{M}$.

$$\mathcal{B}^{6\ell} = \{W \mid W \in \mathcal{I}, |W| = 6\ell, \forall e \in A(D)\ \phi(W,e) = 1\}$$

By the definitions of ϕ, $\mathbb{I}$ and $\mathbb{O}$, $\bigcup_{e \in A(\mathbb{O}) \cap A(\mathbb{I})}(S_e \cup \{e_I, e_O\})$ is an independent set of $\mathcal{M}$, which is contained in $\mathcal{B}^{6\ell}$. In fact, for the optimal value of ℓ, the collection $\mathcal{B}^{6\ell}$ contains all possible candidates for the intersection of $\mathbb{O}'$ and $\mathbb{I}'$, where $\mathbb{O}'$ and $\mathbb{I}'$ are collections of arc disjoint in-branchings and arc disjoint out-branchings which form an optimum solution. Our goal is to find one such candidate partial solution from $\mathcal{B}^{6\ell}$. We are now ready to state the following lemma which shows that a representative family of $\mathcal{B}^{6\ell}$ always contains a candidate partial solution which can be extended to a complete solution.

Lemma 12.5. *Let D be a λ-connected digraph on n vertices, $r \in V(D)$ and $\ell \in [\lambda(n-2)]$. There exists a λ-connected spanning subdigraph D' of D with at most $2\lambda(n-1)-\ell$ arcs if and only if, there exists $\widehat{T} \in \widehat{\mathcal{B}}^{6\ell} \subseteq_{rep}^{n'-6\ell} \mathcal{B}^{6\ell}$, where $n' = 6\lambda(n-1)$, such that D has a of λ arc disjoint in-branchings containing $A_{\widehat{T}}$ and λ arc disjoint out-branchings containing $A_{\widehat{T}}$, which are all rooted at r.*

Proof. In the forward direction consider a λ-connected spanning subdigraph D' of D with at most $2\lambda(n-1)-\ell$ arcs. By Lemma 12.2, D' is union of a collection $\mathbb{I} = \{I_1, I_2, \ldots, I_\lambda\}$ of arc-disjoint in-branchings rooted at r, and a collection $\mathbb{O} = \{O_1, O_2, \ldots, O_\lambda\}$ of arc-disjoint out-branchings rooted at r such that $|A(\mathbb{I}) \cap A(\mathbb{O})| \geq \ell$. By Observation 12.3, for all $i \in [\lambda]$, $\{e^i_{I,1} \mid e \in A(I_i)\}$ is a basis in $\mathcal{M}^i_{I,1}$ and $\{e^i_{I,2} \mid e \in A(I_i)\}$ is a basis in $\mathcal{M}^i_{I,2}$. Similarly, by Observation 12.4, for all $i \in [\lambda]$, $\{e^i_{O,1} \mid e \in A(O_i)\}$ is a basis in $\mathcal{M}^i_{O,1}$ and $\{e^i_{O,2} \mid e \in A(O_i)\}$ is a basis in $\mathcal{M}^i_{O,2}$. Further $\{e_I \mid e \in A(\mathbb{I})\}$ and $\{e_O \mid e \in A(\mathbb{O})\}$ are bases of $\mathcal{M}_I$ and $\mathcal{M}_O$ respectively. Hence the set $Z_{D'} = \{e^i_{I,1}, e^i_{I,2} \mid e \in A(I_i),\ i \in [\lambda]\} \cup \{e^i_{O,1}, e^i_{O,2} \mid e \in A(O_i),\ i \in [\lambda]\} \cup \{e_I \mid e \in A(\mathbb{I})\} \cup \{e_O \mid e \in A(\mathbb{O})\}$ is an independent set in $\mathcal{M}$. Since $|Z_{D'}| = 6\lambda(n-1)$, $Z_{D'}$ is actually a basis in $\mathcal{M}$. Consider $T \subseteq A(\mathbb{I}) \cap A(\mathbb{O})$ with exactly ℓ arcs. Let $T' = \{e^i_{I,1}, e^i_{I,2} \mid e \in T \cap I_i,\ \textit{for some } i \in [\lambda]\} \cup \{e^i_{O,1}, e^i_{O,2} \mid e \in T \cap O_i,\ \textit{for some } i \in [\lambda]\} \cup \{e_I, e_O \mid e \in T\}$. Note that T' is a set of six copies of the ℓ arcs that are common to a pair of an in-branching in $\mathbb{I}$ and an out-branching in $\mathbb{O}$. Therefore, by the definition of $\mathcal{B}^{6\ell}$, $T' \in \mathcal{B}^{6\ell}$. Then, by the definition of representative family, there exists $\widehat{T} \in \widehat{\mathcal{B}}^{6\ell} \subseteq_{rep}^{n'-6\ell} \mathcal{B}^{6\ell}$, such that $\widehat{Z} = (Z_{D'} \setminus T') \cup \widehat{T}$ is an independent set in $\mathcal{M}$. Note that $|\widehat{Z}| = 6\lambda(n-1)$, and hence it is a basis in $\mathcal{M}$. Also note that $A_{\widehat{T}} \subseteq A_{\widehat{Z}}$.

Claim 1. *For any $i \in [\lambda]$ and $e \in A(D)$, either $\{e^i_{I,1}, e^i_{I,2}\} \subseteq \widehat{Z}$ or $\{e^i_{I,1}, e^i_{I,2}\} \cap \widehat{Z} = \emptyset$. And further for every $e \in A(D)$ such that $e^i_{I,1} \in \widehat{Z}$ for some $i \in [\lambda]$, $\widehat{Z}$ also contains e_I. Similarly, for any $i \in [\lambda]$ and $e \in A(D)$, either $\{e^i_{O,1}, e^i_{O,2}\} \subseteq \widehat{Z}$ or $\{e^i_{O,1}, e^i_{O,2}\} \cap \widehat{Z} = \emptyset$, and further, for every $e \in A(D)$ such that $e^i_{O,1} \in \widehat{Z}$ for some $i \in [\lambda]$, $\widehat{Z}$ also contains e_O.*

Proof. Let us consider the first statement. Recall that $\widehat{Z} = (Z_{D'} \setminus T') \cup \widehat{T}$. As discussed earlier, for any $e \in A_D$ and $i \in [\lambda]$, both $Z_{D'}$ and T' contain both the copy in $\{e^i_{I,1}, e^i_{I,2}\}$, or none of the copies from $\{e^i_{I,1}, e^i_{I,2}\}$. Now $\widehat{T} \in \mathcal{B}^{6\ell}$ also satisfies this condition for every $e \in A(D)$. Hence $\widehat{Z}$ satisfies this condition as well. Now let us consider the second

statement. By construction, $Z_{D'}$ contains e_I, e_O for every arc $e \in A(\mathbb{I}) \cup A(\mathbb{O})$. Similarly, T' contains e_I, e_O for every arc $e \in T$. Finally, $\widehat{T}$ contain both e_I, e_O for any arc $e \in A(D)$ if and only if it contains $e^j_{I,1}, e^j_{I,2}$ for some $j \in [\lambda]$. Hence, for any arc e, if $e^j_{I,1} \in \widehat{Z}$ for some $j \in [\lambda]$, then we have $e_I \in \widehat{Z}$ as well. We can similarly show the other two statements. □

Claim 2. *For any $i, j \in [\lambda]$, $i \neq j$, either $\{e^i_{I,1}, e^i_{I,2}\} \cap \widehat{Z} = \emptyset$ or $\{e^j_{I,1}, e^j_{I,2}\} \cap \widehat{Z} = \emptyset$. Similarly, for any $i, j \in [\lambda]$, $i \neq j$, either $\{e^i_{O,1}, e^i_{O,2}\} \cap \widehat{Z} = \emptyset$ or $\{e^i_{O,1}, e^i_{O,2}\} \cap \widehat{Z} = \emptyset$.*

Proof. Again let us consider the first statement and suppose that it is not true. So there are distinct $i, j \in [\lambda]$ such that, $\{e^i_{I,1}, e^i_{I,2}\} \cap \widehat{Z} \neq \emptyset$ and $\{e^j_{I,1}, e^j_{I,2}\} \cap \widehat{Z} \neq \emptyset$ for an arc $e \in A(D)$. Initially, for any arc f in the collection of arc-disjoint in-branchings $\mathbb{I}$, there is exactly one $k \in [\lambda]$ such that $f^k_{I,1}, f^k_{I,2} \in Z_{D'}$ and for any other $k' \in [\lambda]$, $f^{k'}_{I,1}, f^{k'}_{I,2} \notin Z_{D'}$. Further, we have $f_I \in Z_{D'}$. And for any arc not in $A(\mathbb{I})$, no copies of this arc from E_I and $E^k_{I,1}, E^k_{I,2}$ for all $k \in [\lambda]$, is present in $Z_{D'}$. Such a statement also holds true for T' and $\widehat{T}$ as well, as they are both in $\mathcal{B}^{6\ell}$. As $\widehat{Z} = (Z_{D'} \setminus T') \cup \widehat{T}$, it must be the case that $\{e^i_{I,1}, e^i_{I,2}\} \cap (Z_{D'} \setminus T') \neq \emptyset$ and $\{e^j_{I,1}, e^j_{I,2}\} \cap \widehat{T} \neq \emptyset$. But then, $Z_{D'} \setminus T'$ and $\widehat{T}$ have the element e_I in common. This contradicts the fact that $\widehat{T}$ is a representative of T' in $\widehat{\mathcal{B}}^{6\ell}$. Hence, no such pair i, j exists. We can show the second statement in a similar way. □

Since $\widehat{Z}$ is a basis in $\mathcal{M}$, for any $i \in [\lambda], j \in [2]$ and $k \in \{I, O\}$, we have that $\widehat{Z} \cap E^i_{k,j}$ is a basis in $\mathcal{M}^i_{k,j}$. For each $i \in [\lambda]$, let $\widehat{X}^i_1 = \widehat{Z} \cap E^i_{I,1}$ and $\widehat{X}^i_2 = \widehat{Z} \cap E^i_{I,2}$. By Claim 1, $A_{\widehat{X}^i_1} = A_{\widehat{X}^i_2}$ and hence, by Observation 12.3, $\widehat{I}_i = A_{\widehat{X}^i_1}$ forms an in-braching rooted at r. Because of Claim 2, $\{\widehat{I}_i \mid i \in \lambda\}$ are pairwise arc-disjoint as $\widehat{I}_i \cap \widehat{I}_j = \emptyset$ for every $i \neq j \in [\lambda]$. Further $A_{\widehat{T}}$ is covered in arc disjoint in-branchings $\{A_{I_{i,1}} \mid i \in \lambda\}$, as $\widehat{T} \cap E^i_{I,k} \subseteq \widehat{X}^i_j$ for $j \in [2]$. By similar arguments we can show that there exist a collection $\{\widehat{O}_i \mid i \in [\lambda]\}$ of λ out-branchings rooted at r containing $A_{\widehat{T}}$.

The reverse direction of the lemma follows from Lemma 12.2. □

Lemma 12.6. *Let D be a λ connected digraph on n vertices and $\ell \in [\lambda(n-2)]$. In time $2^{\mathcal{O}(\lambda n)}$ we can compute $\widehat{\mathcal{B}}^{6\ell} \subseteq^{n'-6\ell}_{rep} \mathcal{B}^{6\ell}$ such that $|\widehat{\mathcal{B}}^{6\ell}| \leq \binom{n'}{6\ell}$. Here $n' = 6\lambda(n-1)$.*

Proof. We give a dynamic programming based algorithm. Let $\mathcal{D}$ be an array of size $\ell + 1$.

For $i \in \{0, 1, \ldots, \ell\}$ the entry $\mathcal{D}[i]$ will store the family $\widehat{\mathcal{B}}^{6i} \subseteq_{rep}^{n'-6i} \mathcal{B}^{6i}$. We will fill the entries in array $\mathcal{D}$ according to the increasing order of index i, i.e. from $0, 1, \ldots, \ell$. For $i = 0$, we have $\widehat{\mathcal{B}}^0 = \{\emptyset\}$. Let $\mathcal{W} = \{\{e_I, e_O, e^i_{I,1}, e^i_{I,2}, e^j_{O,1}, e^j_{O,2}\} \mid i, j \in [\lambda], e \in A(D)\}$ and note that $|\mathcal{W}| = \lambda^2 m$, where $m = |A(D)|$. Given that we have filled all the entries $\mathcal{D}[i']$, where $i' < i+1$, we fill the entry $\mathcal{D}[i+1]$ at step $i+1$ as follows. Let $\mathcal{F}^{6(i+1)} = (\widehat{\mathcal{B}}^{6i} \bullet \mathcal{W}) \cap \mathcal{I}$.

Claim 1. *$\mathcal{F}^{6(i+1)} \subseteq_{rep}^{n'-6(i+1)} \widehat{\mathcal{B}}^{6(i+1)}$, for all $i \in \{0, 1, \ldots \ell - 1\}$*

Proof. Let $X \in \widehat{\mathcal{B}}^{6(i+1)}$ and Y be a set of size $n' - 6(i+1)$ such that $X \cup Y \in \mathcal{I}$ and $X \cap Y = \emptyset$. We will prove that there exists some $\widehat{X} \in \mathcal{F}^{6(i+1)}$ such that $\widehat{X} \cup Y \in \mathcal{I}$ and $\widehat{X} \cap Y = \emptyset$. This will prove the claim.

Let $e \in A(D)$ and $i, j \in [\lambda]$ such that $\{e_I, e_O, e^i_{I,1}, e^i_{I,2}, e^j_{O,1}, e^j_{O,2}\} \subseteq X$. Let $X' = X \setminus \{e_I, e_O, e^i_{I,1}, e^i_{I,2}, e^j_{O,1}, e^j_{O,2}\}$ and $Y' = Y \cup \{e_I, e_O, e^i_{I,1}, e^i_{I,2}, e^j_{O,1}, e^j_{O,2}\}$. Note that $X' \in \mathcal{I}$ and $Y' \in \mathcal{I}$, since $X \cup Y \in \mathcal{I}$. But, $X' \in \mathcal{B}^{6i}$, $X' \cup Y' \in \mathcal{I}$ and $|Y'| = n' - 6i$. This implies that there exists $\widehat{X}' \in \widehat{\mathcal{B}}^{6i}$ such that, $\widehat{X}' \cap Y' = \emptyset$ and $\widehat{X}' \cup Y' \in \mathcal{I}$. Therefore, $\widehat{X}' \cup \{e_I, e_O, e^i_{I,1}, e^i_{I,2}, e^j_{O,1}, e^j_{O,2}\} \in \mathcal{I}$ and also $\widehat{X}' \cup \{e_I, e_O, e^i_{I,1}, e^i_{I,2}, e^j_{O,1}, e^j_{O,2}\} \in (\widehat{\mathcal{B}}^{6i} \bullet \mathcal{W})$. This proves the claim. □

Now the entry for $\mathcal{D}[i+1]$ is $\widehat{\mathcal{F}}^{6(i+1)}$ which is $n' - 6(i+1)$ representative family for $\mathcal{F}^{6(i+1)}$, which is computed as follows. By Theorem 5.5 we know that $|\widehat{\mathcal{B}}^{6i}| \leq \binom{n'}{6i}$. Hence it follows that $|\mathcal{F}^{6(i+1)}| \leq \lambda^2 m \binom{n'}{6i}$ and moreover, we can compute $\mathcal{F}^{6(i+1)}$ in time $\mathcal{O}(\lambda^2 mn \binom{n'}{6i})$. We use Theorem 5.5 to compute $\widehat{\mathcal{F}}^{6(i+1)} \subseteq_{rep}^{n'-6(i+1)} \mathcal{F}^{6(i+1)}$ of size at most $\binom{n'}{6(i+1)}$. This step can be done in time $\mathcal{O}(\binom{n'}{6(i+1)} tp^\omega + t\binom{n'}{6(i+1)}^{\omega-1})$, where $t = |\mathcal{F}^{6(i+1)}| = \lambda^2 m \binom{n'}{6i}$. We know from Claim 1 that $\mathcal{F}^{6(i+1)} \subseteq_{rep}^{n'-6(i+1)} \mathcal{B}^{6(i+1)}$, and therefore by Lemma 5.7 we have $\widehat{\mathcal{B}}^{6(i+1)} = \widehat{\mathcal{F}}^{6(i+1)} \subseteq_{rep}^{n'-6(i+1)} \mathcal{B}^{6(i+1)}$. Finally, we assign the family $\widehat{\mathcal{B}}^{6(i+1)}$ to $\mathcal{D}[i+1]$. This completes the description of the algorithm and its correctness.

Since $\ell \leq n'/6$, we can bound the total running time of this algorithm as

$$\mathcal{O}\Big(\sum_{i=1}^{\ell} \Big(i^\omega \binom{n'}{6(i+1)} + \binom{n'}{6(i+1)}^{\omega-1}\Big) \lambda^2 m \binom{n'}{6i}\Big) \leq 2^{\mathcal{O}(\lambda n)}$$

□

We have the following algorithm for computing $\mathbb{I}$ and $\mathbb{O}$ given $A(\mathbb{I}) \cap A(\mathbb{O})$. This algorithm extends a given set of arcs to a minimum weight collection of k arc disjoint out-branchings. This is a simple corollary of [Sch03, Theorem 53.10] and it also follows from the results of Gabow [Gab95].

Lemma 12.7. *Let D be a digraph and w be a weight function on the arcs. For any subset X of arcs of D, a vertex r and an integer k, we can find a minimum weight collection $\mathbb{O}$ of k arc disjoint out-branchings rooted at r, such that $X \subseteq A(\mathbb{O})$, if it exists, in polynomial time.*

Proof. We define a new weight function w' which gives a weight 0 to any arc which is contained in X and it is same as w for all other arcs. We now apply an algorithm [Sch03, Theorem 53.10] with the weight function w', which returns a minimum weight collection $\mathbb{O}$ of k arc disjoint out-branchings rooted at r. If $X \subseteq A(\mathbb{O})$, then we return $\mathbb{O}$ as the required solution. Otherwise no such collection exists. □

Theorem 12.4. *Let D be a λ edge connected digraph on n vertices. Then we can find a minimum λ edge connected subgraph of D in $2^{\mathcal{O}(\lambda n)}$.*

Proof. Let $q = 6\lambda(n-1)$. By Lemma 12.2 we know that finding a minimum subdigraph D' of D is equivalent to finding a collection $\mathbb{I}$ of λ arc disjoint in-branchings and a collection $\mathbb{O}$ of λ arc disjoint out-branchings which are all rooted at a vertex $r \in V(D)$ such that $|A(\mathbb{I}) \cap A(\mathbb{O})|$ is maximized . We fix an arbitrary $r \in V(D)$ and for each choice of ℓ, the cardinality of $|A(\mathbb{I}) \cap A(\mathbb{O})|$, we attempt to construct a solution. By Lemma 12.5 we know that there exists a λ-connected spanning subdigraph D' of D with at most $2\lambda(n-1) - \ell$ arcs if and only if there exists $\widehat{T} \in \widehat{\mathcal{B}}^{6\ell} \subseteq_{rep}^{n'-6\ell} \mathcal{B}^{6\ell}$, where $n' = 6\lambda(n-1)$, such that D has a collection $\mathbb{I} = \{I_1, I_2, \ldots, I_\lambda\}$ of arc disjoint in-branchings rooted at r and a collection $\mathbb{O} = \{O_1, O_2, \ldots, O_\lambda\}$ of arc disjoint out-branchings rooted at r such that $A_{\widehat{T}} \subseteq A(\mathbb{I}) \cap A(\mathbb{O})$. Using Lemma 12.6 we compute $\widehat{\mathcal{B}}^{6l} \subseteq_{rep}^{n'-6\ell} \mathcal{B}^{6\ell}$ in time $2^{\mathcal{O}(\lambda n)}$, and for every $F \in \widehat{\mathcal{B}}^{6\ell}$ we check if A_F can be extended to a collection of λ arc disjoint out-branchings rooted at r and a collection of λ arc disjoint in-branchings rooted at r, using Lemma 12.7. Since $\ell \leq \lambda(n-2)$, the running time of the algorithm is bounded by $2^{\mathcal{O}(\lambda n)}$. □

An algorithm with the same running time for the weighted version of the problem can be obtained by using the notion of weighted representative sets in the above. This then proves Theorem 12.1.

12.2 Undirected Graphs

In this section, we give an algorithm for computing a minimum λ-connected subgraph of an undirected graph G. As before, we only consider the unweighted version of the problem. While there is no equivalent characterization of λ-connected graphs as there was in the case of digraphs, we show that we can obtain a characterization by converting the graph to a directed graph with labels on the arcs. Then, as in the previous section, we embed the solutions in a linear matroid and compute them by a dynamic programming algorithm with representative families.

Let D_G be the digraph with $V(D_G) = V(G)$ and for each edge $e = (u, v) \in E(G)$, we have two arcs $a_e = (u, v)$ and $a'_e = (v, u)$ in $A(D_G)$. We label that arcs a_e and a'_e by the edge e, which is called the *type* of these arcs. For $X \subseteq A(D_G)$ let $\mathsf{Typ}(X) = \{e \in E(G) \mid a_e \in X \text{ or } a'_e \in X\}$. The following two lemmata relate λ-connected subgraphs of G with collections of out-branchings in D_G.

Lemma 12.8. *Let G be an undirected graph and D_G be the digraph constructed from G as described above. Then G is λ-connected if and only if for any $r \in V(D_G)$, there are λ arc disjoint out-branchings rooted at r in D_G.*

Proof. In the forward direction, observe that since G is λ-connected, D_G is also λ-connected. By Lemma 12.1, for any $r \in V(D_G)$, there are λ arc disjoint out-branchings rooted at r in D_G.

In the reverse direction, suppose there are λ arc disjoint out-branchings rooted at r in D_G. Therefore for any vertex $v \in V(G)$, there is a collection of λ arc disjoint paths from r to v in D_G. If G is not λ connected, then there is a cut $(X, \overline{X})$ such that $|\delta_G(X)| \leq \lambda - 1$, and we may assume that $r \in X$. But then in D_G, there are at most $\lambda - 1$ arcs which go from

X to $\overline{X}$. Therefore, for any vertex $v \in \overline{X}$, there are at most $\lambda - 1$ arc disjoint paths from r to v in D_G. This is a contradiction. This completes the proof of this lemma. □

By Lemma 12.8 we know that G is λ-connected if and only if for any $r \in V(D)$, there is a collection $\mathbb{O}$ of λ arc disjoint out-branchings rooted at r in D_G. So given such a collection of out-branchings, we can obtain a λ-connected subgraph of G with at most $\lambda(n-1)$ edges. Observe that for an edge $e \in E(G)$ which is not incident on r, the two arcs corresponding to it in D_G may appear in two distinct out-branchings of $\mathbb{O}$, but for an edge e incident on r in G, only the corresponding outgoing arc of r may appear in $\mathbb{O}$. Since there are $\lambda(n-1)$ arcs in total that appear in $\mathbb{O}$ and at least λ of those are incident on r, the number of edges of G such that both the arcs corresponding to it appear in $\mathbb{O}$ is upper bounded by $\frac{\lambda(n-2)}{2}$. So any minimal λ-connected subgraph of G has $\lambda(n-1) - \ell$ edges where $\ell \in [\lfloor \frac{\lambda(n-2)}{2} \rfloor]$.

Lemma 12.9. *Let G be an undirected λ-connected graph on n vertices and $\ell \in [\lfloor \frac{\lambda(n-2)}{2} \rfloor]$. G has a λ-connected subgraph G' with at most $\lambda(n-1) - \ell$ edges if and only if for any $r \in V(D_G)$, $D_{G'}$ has λ arc disjoint out-branchings $\mathbb{O} = \{O_1, O_2, \ldots, O_\lambda\}$ rooted at r such that $|\mathsf{Typ}(A(\mathbb{O}))| \leq \lambda(n-1) - \ell$.*

Proof. In the forward direction let G' be a λ connected subgraph of G with at most $\lambda(n-1) - \ell$ edges. By Lemma 12.8, for any $r \in V(D_{G'})$, $D_{G'}$ has λ arc disjoint out-branchings rooted at r. Observe that there are at most $\lambda(n-1) - \ell$ edges in G', therefore the number of different types of edges possible in $D_{G'}$ is at most $\lambda(n-1) - \ell$.

In the reverse direction, consider a vertex $r \in V(D)$ with λ arc disjoint out-branchings $\mathbb{O} = \{O_1, O_2, \ldots, O_\lambda\}$ such that $|\mathsf{Typ}(A(\mathbb{O}))| \leq \lambda(n-1) - \ell$. Consider the graph $G' = (V(D), \mathsf{Typ}(A(\mathbb{O})))$. Observe that G' has at most $\lambda(n-1) - \ell$ edges and is λ connected subgraph of G. This concludes the proof. □

By Lemma 12.9, a collection $\mathbb{O}$ of out-branchings rooted at some vertex r, that minimizes $|\mathsf{Typ}(A(\mathbb{O}))|$ corresponds to a minimum λ-connected subgraph of G. In the rest of this section, we design an algorithm that find a collection of arc-disjoint out-branchings $\mathbb{O}$ in D_G such that $|\mathsf{Typ}(A(\mathbb{O}))|$ is minimized. The first step of our algorithm is to compute the

set of edges of G such that both the arcs corresponding to it appear in the collection $\mathbb{O}$, and then we can extend this to a full solution in polynomial time.

Fix a vertex r. Let D_G^r denote the digraph obtained from D_G by removing the arcs in $\mathsf{In}_{D_G}(r)$. Observe that $A(D_G^r)$ can be partitioned as follows.

$$A(D_G^r) = \biguplus_{v \in V(D_G^r)} \mathsf{In}_{D_G^r}(v)$$

We construct a pair of a graphic matroid and a partition matroid, corresponding to each of the λ out-branching that we want to find. For each $i \in [\lambda]$, we define a matroid $\mathcal{M}_1^i = (A_1^i, \mathcal{I}_1^i)$ which is a graphic matroid of D_G^r whose ground set A_1^i is a copy of the arc set $A(D_G)$. Similarly, for each $i \in [\lambda]$ we define matroid $\mathcal{M}_2^i = (A_2^i, \mathcal{I}_2^i)$, which is a partition matroid on the ground set A_2^i, which is a copy of the arc set $A(D_G^r)$, such that the following holds.

$$\mathcal{I}_2^i = \{I \mid I \subseteq A_2^i, |I \cap \mathsf{In}_{D_G^r}(v)| \leq 1, \textit{ for all } v \in V(D_G^r)\}$$

Next, let $\mathcal{M}_O$ be a uniform matroid of rank $\lambda(n-1)$ on the ground set A_O where A_O is also a copy of $A(D_G)$. Finally, we define the matroid $\mathcal{M} = (A_{\mathcal{M}}, \mathcal{I})$ as the direct sum of $\mathcal{M}_O$ and $\mathcal{M}_1^i, \mathcal{M}_2^i$, for $i \in [\lambda]$, i.e.

$$\mathcal{M} = \Big(\bigoplus_{i \in [\lambda]} (\mathcal{M}_1^i \oplus \mathcal{M}_2^i) \Big) \oplus \mathcal{M}_O$$

Note that the rank of this matroid is $3\lambda(n-1)$ and it is representable over any field of size at least $|A(D)| + 1$. For an arc $a \in A(D)$, we denote its copies in A_1^i, A_2^i and A_O by a_1^i, a_2^i and a_O respectively. For a collection $\mathbb{O}$ of λ out-branchings in D_G, by $A(\mathbb{O})$ we denote the set of arcs which is present in some $O \in \mathbb{O}$. For $X \in \mathcal{I}$, by A_X we denote the set of arcs $a \in A(D_G)$ such that $X \cap \bigcup_{i=1}^{\lambda} \{a_i^1, a_i^2\} \neq \emptyset$. For $e \in E(G)$ and $i \in [\lambda]$, we let $S_e^i = \{(a_e)_1^i, (a_e)_2^i, (a'_e)_1^i, (a'_e)_2^i\}$ and $S_e = \{(a_e)_O, (a'_e)_O\} \cup \big(\bigcup_{i=1}^{\lambda} S_e^i \big)$. We define a function $\psi : \mathcal{I} \times E(G) \to \{0, 1\}$, where for $W \in \mathcal{I}, e \in E(G)$, $\psi(W, e) = 1$ if and only if exactly one of the following holds.

- Either $W \cap S_e = \emptyset$.
- Or, there exists $t \neq t' \in [\lambda]$ such that,
  - (i) $(a_e)_O, (a'_e)_O \in W$
  - (ii) $S_e^t \cap W = \{(a_e)_1^t, (a_e)_2^t\}$
  - (iii) $S_e^{t'} \cap W = \{(a'_e)_1^{t'}, (a'_e)_2^{t'}\}$
  - (iv) $\forall i \in [\lambda] \setminus \{t, t'\}$, $S_e^i \cap W = \emptyset$.

Now for each $\ell \in [\lfloor \lambda(n-2)/2 \rfloor]$, we define the following set.

$$\mathcal{B}^{6\ell} = \{W \mid W \in \mathcal{I}, |W| = 6\ell \text{ and } \forall e \in E(G), \phi(W, e) = 1\}$$

Observe that for every $W \in \mathcal{B}^{6\ell}$, $|\mathsf{Typ}(A_W)| = \ell$ and, $a_e \in A_W$ if and only if $a'_e \in A_W$. Therefore any set in this collection corresponds to a potential candidate for the subset of arcs which appear in exactly two out-branchings in $\mathbb{O}$. We are now ready to state the following lemma, which relates the computation of λ out-branchings minimizing types and representative sets.

Lemma 12.10. *Let G be a λ-connected undirected graph on n vertices, D_G its corresponding digraph and $\ell \in [\lfloor \frac{\lambda(n-2)}{2} \rfloor]$. There exists a set $\mathbb{O} = \{O_1, O_2, \ldots O_\lambda\}$ of out-branchings rooted ar r, with $|\mathsf{Typ}(A(\mathbb{O}))| \leq \lambda(n-1) - \ell$ in D_G if and only if there exists $\widehat{T} \in \widehat{\mathcal{B}}^{6\ell} \subseteq_{rep}^{n'-6\ell} \mathcal{B}^{6\ell}$, where $n' = 3\lambda(n-1)$, such that D_G has a collection $\widehat{\mathbb{O}}$ of λ out-branchings rooted at at r, $A_{\widehat{T}} \subseteq A(\widehat{\mathbb{O}})$ and $|\mathsf{Typ}(\widehat{\mathbb{O}})| \leq \lambda(\mathsf{n}-1) - \ell$.*

Proof. In the forward direction, let $\mathbb{O}$ be a collection of λ out-branchings rooted at r in D_G, such that $|\mathsf{Typ}(A(\mathbb{O}))| \leq \lambda(n-1) - \ell$. For each $i \in [\lambda]$, let O_i^1 and O_i^2 be the independent sets in M_i^1 and M_i^2 respectively, corresponding to the out-branching $O_i \in \mathbb{O}$. Now consider the set $I = \{a_O, a_1^i, a_2^i \mid a \in A(O_i), i \in [\lambda]\}$ in the matroid $\mathcal{M}$. For each $i \in [\lambda]$, $\{a_1^i \mid a \in A(O_i)\}$ is an independent set in the graphic matroid $\mathcal{M}_1^i$ since O_i corresponds to a tree in the underlying graph. And similarly $\{a_2^i \mid a \in A(O_i)\}$ is an independent set in the partition matroid $\mathcal{M}_2^i$, since O_i is an out-branching. Finally $\{a_O \mid a \in A(\mathbb{O})\}$ is of cardinality $\lambda(n-1)$ and therefore a basis in $\mathcal{M}_O$. Therefore, I is

a basis in the matroid $\mathcal{M}$, and note that $|I| = 3\lambda(n-1)$. Let $P' = \{e \in E(G) \mid a_e, a'_e \in A(\mathbb{O})\}$ and clearly, $|P'| \geq \ell$. Fix an arbitrary subset P of P' with exactly ℓ edges. Let $T = \{(a_e)_O, (a'_e)_O), (a_e)^i_1, (a_e)^i_2, (a'_e)^j_1, (a'_e)^j_2 \mid e \in P\}$. Observe that $T \in \mathcal{B}^{6\ell}$, and therefore there is a $\widehat{T} \in \widehat{\mathcal{B}}^{6\ell} \subseteq^{n'-6\ell}_{rep} \mathcal{B}^{6\ell}$ such that $\widehat{I} = (I \setminus T) \cup \widehat{T}$ is an independent set in $\mathcal{M}$ of size $3\lambda(n-1)$, and note that $|\mathsf{Typ}(A_{\widehat{T}})| = \ell$.

We will now prove that there is a collection $\widehat{\mathbb{O}}$ of λ arc-disjoint out-branchings, such that $A_{\widehat{T}} \subseteq A(\widehat{\mathbb{O}})$ and $|\mathsf{Typ}(\widehat{\mathbb{O}})| \leq \lambda(n-1) - \ell$.

Claim 1. *For any arc $a \in A(D_G)$ and $i \in [\lambda]$, either $a^i_1, a^i_2 \in \widehat{I}$ or $a^i_1, a^i_2 \notin \widehat{I}$. Further, if $a^i_1 \in \widehat{I}$ for some $i \in [\lambda]$, then $a_O \in \widehat{I}$ as well.*

Proof. Let us consider the first statement. Initially, the statement hold for I by construction. And since $T, \widehat{T} \in \mathcal{B}^{6\ell}$, the statement holds for them as well. This implies that for $\widehat{I} = (I \setminus T) \cup \widehat{T}$ also satisfies this statement. Now we consider the second statement. Initially for any arc $a \in A(\mathbb{O})$ we have $a_O, a^i_1 \in I$. And T contains a_O for some arc a if and only if it also contains a^i_1 for some $i \in [\lambda]$, and a similar statement holds for $\widehat{T}$. Hence, the second statement also holds for $\widehat{I}$. □

Claim 2. *For any arc a, and any pair of $i \neq j \in [\lambda]$, either $a^i_1, a^i_2 \notin \widehat{I}$ or $a^j_1, a^j_2 \notin \widehat{I}$.*

Proof. Suppose that there were some $i \neq j \in [\lambda]$ and an arc $a \in A(D_G)$ such that $a^i_1, a^i_2, a^j_1, a^j_2 \in \widehat{I}$. Initially, for any arc b in the collection of arc-disjoint out-branchings $\mathbb{O}$, there is exactly one $k \in [\lambda]$ such that $b^k_1, b^k_2 \in I$ and for any other $k' \in [\lambda]$, $b^{k'}_1, b^{k'}_2 \notin I$. Further, we have $b_O \in I$. And for any arc not in $A(\mathbb{O})$, no copies of this arc from E_O and E^k_1, E^k_2 for all $k \in [\lambda]$, is present in I. Such a statement also holds true for T' and $\widehat{T}$ as well, as they are both in $\mathcal{B}^{6\ell}$. As $\widehat{I} = (I \setminus T') \cup \widehat{T}$, it must be the case that $\{a^i_1, a^i_2\} \cap (I \setminus T') \neq \emptyset$ and $\{a^j_1, a^j_2\} \cap \widehat{T} \neq \emptyset$. But then, $I \setminus T'$ and $\widehat{T}$ have the element e_I in common, which contradicts the fact that $\widehat{T}$ is a representative of T, in $\mathcal{B}^{6\ell}$. □

Now since $\widehat{I}$ is a basis of $\mathcal{M}$, we have that $X^i_j = \widehat{I} \cap A^i_j$ is a basis of $\mathcal{M}^i_j$, where $i \in [\lambda]$ and $j \in [2]$. Now by Claim 1, $A_{X^i_1} = A_{X^i_2}$, and hence $\widehat{O}_i = A_{X^i_1}$ is an out-branching rooted at r in D_G. Next, by Claim 2, the collection $\widehat{\mathbb{O}} = \{\widehat{O}_i \mid i \in [\lambda]\}$ is pairwise

arc-disjoint. And finally we bound the value of $|\mathsf{Typ}(\widehat{\mathbb{O}})|$. Initially $|\mathsf{Typ}(A_I)| = \lambda(n-1)-\ell$, and $|\mathsf{Typ}(A_T)| = |\mathsf{Typ}(A_{\widehat{T}})| = \ell$. Since $T \in \mathcal{B}^{6\ell}$, for any edge e of G, either both or neither of a_e, a'_e lie in T, and a similar statement holds for $\widehat{T}$ as well. Therefore we have $|\mathsf{Typ}(A_{I\setminus T})| = \lambda(n-1)-2\ell$, and hence $|\mathsf{Typ}(A_{\widehat{I}})| = |\mathsf{Typ}(\widehat{\mathbb{O}})| = \lambda(n-1)-\ell$.

The reverse direction follows trivially as $\widehat{\mathbb{O}}$ itself is the required collection of out-branchings.

□

Lemma 12.11. *Let G be a λ-connected undirected graph on n vertices, D_G its corresponding digraph and $\ell \in [\lfloor \lambda(n-2)/2 \rfloor]$. In time $2^{\mathcal{O}(\lambda n)}$ we can compute $\widehat{\mathcal{B}}^{6\ell} \subseteq^{n'-6\ell}_{rep} \mathcal{B}^{6\ell}$ such that $|\widehat{\mathcal{B}}^{6\ell}| \leq \binom{n'}{6\ell}$. Here $n' = 3\lambda(n-1)$.*

Proof. We give a dynamic programming based algorithm. Let $\mathcal{D}$ be an array of size $\ell+1$. For $i \in [\ell+1]$ the entry $\mathcal{D}[i]$ will store the family $\widehat{\mathcal{B}}^{6\ell} \subseteq^{n'}_{rep} \mathcal{B}^{6\ell}$. We will fill the entries in array $\mathcal{D}$ according to the increasing order of index i, where $i \in \{0,1,\ldots\ell\}$. For $i=0$, we have $\widehat{\mathcal{B}}^0 = \{\emptyset\}$. Let $\mathcal{W} = \big\{\{(a_e)_O, (a'_e)_O, (a_e)^i_1, (a_e)^i_2, (a'_e)^j_1, (a'_e)^j_2\} \mid i,j \in [\lambda],\ i \neq j,\ \text{and } e \in E(G) \text{ is not incident on } r\big\}$, and note that $|\mathcal{W}| \leq \binom{\lambda}{2}m$. Given that we have filled all the entries $\mathcal{D}[i']$ for every $i' < i+1$, we fill the entry at $\mathcal{D}[i+1]$ as described below.

Let $\mathcal{F}^{6(i+1)} = (\widehat{\mathcal{B}}^{6i} \bullet \mathcal{W}) \cap \mathcal{I}$. Observe that for any $X \in \mathcal{F}^{6(i+1)}$, $|\mathsf{Typ}(A_X)| = i+1$. We now have the following claim.

Claim 1. *$\mathcal{F}^{6(i+1)} \subseteq^{n'-6(i+1)}_{rep} \mathcal{B}^{6(i+1)}$, for all $i \in \{0,1,\ldots\ell-1\}$*

Proof. Let $X \in \mathcal{B}^{6(i+1)}$ and Y be a set of size $n'-6(i+1)$ such that $X \cup Y \in \mathcal{I}$ and $X \cap Y = \emptyset$. We will show that there exists some $X' \in \mathcal{F}^{6(i+1)}$ such that $X' \cup Y \in \mathcal{I}$ and $X' \cap Y = \emptyset$. This will prove the claim.

Let $e \in E(G)$ and $i \neq j \in [\lambda]$ such that $\{(a_e)_O, (a'_e)_O, (a_e)^i_1, (a_e)^i_2, (a'_e)^j_1, (a'_e)^j_2\} \subseteq X$. Let $X' = X \setminus \{(a_e)_O, (a'_e)_O, (a_e)^i_1, (a_e)^i_2, (a'_e)^j_1, (a'_e)^j_2\}$ and $Y' = Y \cup \{(a_e)_O, (a'_e)_O, (a_e)^i_1, (a_e)^i_2, (a'_e)^j_1, (a'_e)^j_2\}$. Note that $X' \in \mathcal{I}$ and $Y' \in \mathcal{I}$, since $X \cup Y \in \mathcal{I}$. But $X' \in \mathcal{B}^{6i}$, $X' \cup Y' \in \mathcal{I}$ and $|Y'| = n'-6i$, which implies that there exists $\widehat{X}' \in \widehat{\mathcal{B}}^{6i}$ such that, $\widehat{X}' \cup Y' \in \mathcal{I}$. Therefore, $\widehat{X}' \cup \{(a_e)_O, (a'_e)_O, (a_e)^i_1, (a_e)^i_2, (a'_e)^j_1, (a'_e)^j_2\} \in \mathcal{I}$ and observe that $\widehat{X}' \cup \{(a_e)_O, (a'_e)_O, (a_e)^i_1, (a_e)^i_2, (a'_e)^j_1, (a'_e)^j_2\} \in (\widehat{\mathcal{B}}^{6i} \bullet \mathcal{W})$. □

We fill the entry for $\mathcal{D}[i+1]$ as the following. By Theorem 5.5 we know that $|\widehat{\mathcal{B}}^{6i}| \leq \binom{n'}{6i}$, and hence it follows that $|\mathcal{F}^{6(i+1)}| \leq \binom{\lambda}{2} m \binom{n'}{6i}$. Moreover, we can compute $\mathcal{F}^{6(i+1)}$ in time $\mathcal{O}(\binom{\lambda}{2} mn \binom{n'}{6i})$. We use Theorem 5.5 to compute $\widehat{\mathcal{F}}^{6(i+1)} \subseteq_{rep}^{n'-6(i+1)} \mathcal{F}^{6(i+1)}$ of size at most $\binom{n'}{6(i+1)}$ in time $\mathcal{O}(\binom{n'}{6(i+1)} ti^{\omega} + t\binom{n'}{6(i+1)}^{\omega-1})$, where $t = |\mathcal{F}^{6(i+1)}|$. We know from Claim 1 that $\mathcal{F}^{6(i+1)} \subseteq_{rep}^{n'-6(i+1)} \mathcal{B}^{6(i+1)}$, and therefore $\widehat{\mathcal{B}}^{6(i+1)} = \widehat{\mathcal{F}}^{6(i+1)} \subseteq_{rep}^{n'-6(i+1)} \mathcal{B}^{6(i+1)}$. And finally, we assign the family $\widehat{\mathcal{B}}^{6(i+1)}$ to $\mathcal{D}[i+1]$. This completes the description of the algorithm and its correctness. Using the fact that $\ell < n'/6$, the total running time of the algorithm can bounded as follows.

$$\mathcal{O}\Big(\sum_{i=1}^{\ell} \left(\binom{n'}{6(i+1)} ti^{\omega} + t\binom{n'}{6(i+1)}^{\omega-1} \right) \lambda^2 m \binom{n'}{6i} \Big) \leq 2^{\mathcal{O}(\lambda n)}$$

□

Theorem 12.5. *Let G be a λ edge connected graph on n vertices. Then we can find a minimum λ edge connected subgraph of G in $2^{\mathcal{O}(\lambda n)}$ time.*

Proof. Let $n' = 2\lambda(n-1)$. By Lemma 12.9 we know that finding a minimum subgraph G' of G is equivalent to finding a collection $\mathbb{O}$ of λ arc disjoint out-branchings in D_G rooted at a fixed vertex $r \in V(D)$ such that $\mathsf{Typ}(\mathbb{O})$ is minimized. For each choice of $\ell \in [\lfloor \frac{\lambda(n-2)}{2} \rfloor]$, by Lemma 12.8 and Lemma 12.10, we know that there exists a λ-connected spanning subgraph G' of G with at most $\lambda(n-1) - \ell$ arcs if and only if there exists $\widehat{T} \in \widehat{\mathcal{B}}^{6\ell} \subseteq_{rep}^{n'-6\ell} \mathcal{B}^{6\ell}$, such that D_G has a collection $\mathbb{O} = \{O_1, O_2, \ldots, O_\lambda\}$ of out-branchings rooted at r and $A_{\widehat{T}} \subseteq A(\mathbb{O})$. So we apply Lemma 12.11 to compute $\widehat{\mathcal{B}}^{6\ell}$ in time $2^{\mathcal{O}(\lambda n)}$, and for every $F \in \widehat{\mathcal{B}}^{6\ell}$ we check if A_F can be extended to λ out-branchings rooted at r in D_G by using Lemma 12.7. We return the graph G' with the least number of edges, among all the graphs computed above, as our solution. Since $\ell \leq \lambda(n-2)/2$, the running time of the algorithm is bounded by $2^{\mathcal{O}(\lambda n)}$. This completes the proof. □

As before, an algorithm with the same running time for the weighted version of the problem may be obtained by using the notion of weighted representative sets in the above. This then proves Theorem 12.2.

12.3 Algorithms for Network Augmentation

The algorithms for Minimum Weight λ-connected Spanning Subgraph may be used to solve instances of Minimum Weight λ-connectivity Augmentation as well. Given an instance (D, L, w, λ) of the augmentation problem, we construct an instance (D', w', λ) of Minimum Weight λ-connected Spanning Subgraph, where $D' = G \cup L$ and w' is a weight function that gives a weight 0 to arcs in $E(G)$ and it is w for the arcs from L. It is easy to see that the solution returned by our algorithm contains a minimum weight augmenting set. A similar approach will work for augmenting graphs as well. This then proves Theorem 12.3.

Chapter 13

Parameterized Algorithms for Network Augmentation I

In connectivity augmentation problems, the input is a (multi) graph and the objective is to increase edge or vertex connectivity by adding the minimum number (weight) of additional edges, called links. This problem was first studied by Eswaran and Tarjan [ET76] who showed that increasing the edge connectivity of a given graph to 2 by adding minimum number of links (also called an augmenting set) is polynomial time solvable. Subsequent work in [WN87, Fra92b] showed that this problem is also polynomial time solvable for any given target value of edge connectivity to be achieved. However, if the set of links is restricted, that is, there are pairs of vertices in the graph which do not constitute a link, or if the links have (non-identical) weights on them, then the problem of computing the minimum size (or weight) augmenting set is NP-hard [ET76]. It is interesting to note that the vertex version of the problem is substantially more difficult even when the set of links which can be added is unrestricted, and is only known to be polynomial time for the special case when the connectivity of the graph is to be increased by 1, with the general case still open [Veg11].

In this chapter, we will focus on the parameterized complexity of the following problem, called Weighted Edge Connectivity Augmentation By One (w-Aug-One). As we discussed in chapter 8, this problems in NP-hard.

w-Aug-One	Parameter: k
Input: Graph $G = (V, E)$ which is λ-edge connected, set of links $\mathcal{L}$, integer k, weight function w on $\mathcal{L}$, $p \in \mathbb{R}$.	
Question: Is there a link set F such that $w(F) \leq p$, $\lvert F\rvert \leq k$ and $(V, E \cup F)$ is $\lambda + 1$-edge connected?	

Recently, Marx and Vegh [MV15] gave a kernel with $\mathcal{O}(k)$ vertices, $\mathcal{O}(k^3)$ links and weights of $(k^6 \log k)$ bit integers for this problem. This leads to an FPT algorithm running in time $2^{\mathcal{O}(k \log k)}|V|^{\mathcal{O}(1)}$. In this chapter, we obtain a single exponential FPT algorithm for this problem. In particular, we have the following theorem.

Theorem 13.1. w-Aug-One *is solvable in time* $9^k|V|^{\mathcal{O}(1)}$.

Our approach.

The result of Dinits et al. [DKL76] allows us to assume without loss of generality that either $\lambda = 1$ and G is a tree, or $\lambda = 2$ and G is a cactus graph. We then define an auxiliary graph which we call a *link-terminal intersection graph*, corresponding to a given instance of the problem and show that augmenting sets for the given instance exactly correspond to Steiner trees in the link-intersection graph for a specified set of terminals. This structural lemma forms the backbone of our algorithms. The set of vertices in the input graph corresponding to leaves or degree-2 vertices is chosen as the set of terminals for the auxiliary Steiner Tree instance. However, the Steiner Tree problem is not FPT when parameterized by the solution size. Thus, for our purposes we can only use those parameterization for which Steiner Tree is known to be FPT. In our context we use the following known results about Steiner Tree: an algorithm with running time $2^{|T|}|V(G)|^{O(1)}$ [DW71], where T is the set of input terminals. In the w-Aug-One problem, the number of links in any augmenting set is at least half the number of leaves or degree-2 vertices in the input graph. We thus obtain a Steiner Tree instance where the number of terminals is bounded linearly in the parameter, following which we can invoke known results to give an algorithm.

13.1 Tools and Techniques

We begin with a few definitions and notations that will be used in this chapter and the following chapter.

Definition 13.2. *Given a graph $G = (V, E)$, a spanning subgraph H of G is called a* **cactus** *if it is 2-edge connected and every edge belongs to exactly one cycle. Equivalently, H can be written as the union of a set $\mathcal{C}$=$\{C_1, \ldots, C_\ell\}$ where each C_i is a cycle, the graph induced on their union is connected and every edge in this graph belongs to exactly one cycle. Note that the subgraph H can have multi-edges. A cycle of length 2 in a cactus is called a* ***2-circuit****.*

Definition 13.3. *A set Z of t edges in a graph is called a* ***t-cut*** *if deleting the edges of Z disconnects the graph and for every subset $Z' \subset Z$, the graph remains connected after deleting the edges in Z'.*

Observation 13.1. *Let $\{e, f\}$ be a 2-cut of cactus $\mathcal{C}$ and let X be the vertices of degree 2 in the cactus. Each of the two components obtained by deleting the 2-cut $\{e, f\}$ contains a vertex from X.*

The following observation is a consequence of the definition of a t-cut :

Observation 13.2. *Every 1-cut of a tree is a bridge in a tree. The 2-cuts in a cactus are contained in a cycle, i.e., if $\{e, f\}$ is a 2-cut, then both the edges e and f belong to the same cycle C of the cactus $\mathcal{C}$.*

Definition 13.4. *We say that a link* covers *a bridge in a tree (2-cut in a cactus) if the end points of the link are in distinct connected components obtained by deleting the bridge (respectively 2-cut). Note that since every 2-cut is contained within a cycle, the vertices of the cycle will be in two distinct connected components.*

Definition 13.5. *Tree Decomposition. A* tree decomposition *of a graph $G = (V, E)$ is a pair (M, β) where M is a rooted tree and $\beta : V(M) \to 2^V$ such that,*

(i) $\bigcup_{t \in V(M)} \beta(t) = V$

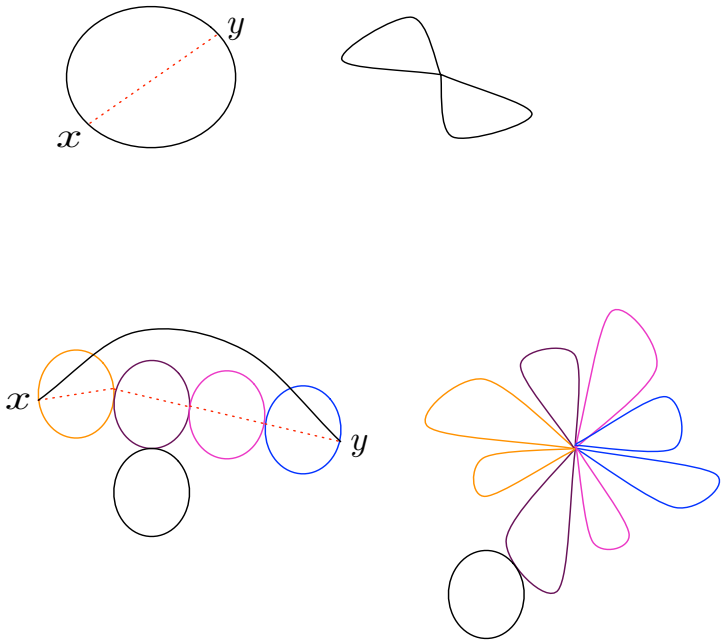

Figure 13.1: An illustration of projection of a link and the operation of refining a single link.

(ii) *For each edge* $(u, v) \in E$*, there is a* $t \in V(M)$ *such that both* u *and* v *belong to* $\beta(t)$*.*

(iii) *For each* $v \in V$*, the nodes in the set* $\{t \in V(M) \mid v \in \beta(t)\}$ *form a connected subtree of* M*.*

Let (M, β) *be a tree decomposition of a graph* G*. The* width *of* (M, β) *is* $min\{|\beta(t)| - 1 \mid t \in V(M)\}$*.*

13.1.1 Link-Intersection Graphs

In the first subsection, we introduce link-intersection graphs and prove certain properties of these graphs. In the following subsection, we prove our main structural result relating Steiner trees in these graphs and solutions for our problem.

Definition 13.6. (Projection of a link onto a cactus). *Consider a cactus* $\mathcal{C}$ *and a link* (u, v)*. Let* $a_1, \ldots, a_r$ *be the cut vertices of the cactus (other than* u *and* v*) which lie on every* u*-*v *path in the cactus and let* $C_1, \ldots, C_{r+1}$ *be the cycles which contain the segments* u*-*a_1*,* a_1*-*a_2*,...,* a_r*-*v *respectively of this path. Then, we say that the pair* $\langle u, a_1 \rangle$ *is the*

projection *of the link* (u, v) *onto the cycle* C_1*, the pair* $\langle a_r, v\rangle$ *is the projection of the link* (u, v) *onto the cycle* C_{r+1} *and the pair* $\angle a_i, a_{i+1}\rangle$ *is the projection of the link onto the cycle* C_{i+1}*. Note that if* u *and* v *lie in the same cycle of the cactus, then the projection is simply the pair* $\langle u, v\rangle$*. If the pair* $\langle x, y\rangle$ *is a projection of a link onto a cycle of the cactus, then we say that the link is* ***projectively incident*** *to* x*,*y *and this cycle and a link* (x, y) *is said to be* **properly incident** *on* x *and* y*. We say that the projection of a link* e *on to the cycle* C *is* non-trivial *if it is projectively incident to exactly two vertices of the cycle.*

Definition 13.7. (Crossing pairs). *Consider distinct vertices* x_1, x_2, y_1, y_2 *which lie on the same cycle in a cactus. We say that the pair* $\langle x_1, y_1\rangle$ crosses *the pair* $\langle x_2, y_2\rangle$ *if any path from* x_2 *to* y_2 *contains exactly one of vertices from* $\{x_1, y_1\}$ *as an internal vertex. Observe that the crossing relation is symmetric and hence we say that the pairs* $\langle x_1, y_1\rangle$ *and* $\langle x_2, y_2\rangle$ *are* **crossing***. Let* e *be a link that is incident (possibly projectively incident) on the vertices* x_1 *and* y_1 *of the cycle, then we say the link* $e = (x_1, y_1)$ *crosses the pair* $\langle x_2, y_2\rangle$*. If both the pairs correspond to links* $e = (x_1, y_1)$ *and* $f = (x_2, y_2)$*, then we say that the links* e *and* f *cross. A set* L *of links is said to be* **laminar** *if it does not contain a pair of links which cross.*

We now give an equivalent definition (to Definition 13.4) of covering 2-cuts using the notion of crossing pairs. For ease of description, we will be working with this definition from now on.

Definition 13.8. (Covering a 2-cut). *Let* $e = (a_1, b_1)$ *and* $f = (a_2, b_2)$ *be the edges of a cycle* C *such that the vertices appear as* a_1, b_1, a_2, b_2 *in a fixed ordering of the vertices of the cycle. Note that* b_1 *can be the same as* a_2 *and* a_1 *can be the same as* b_2*. We say that the link* $d = (x, y)$ *covers the 2-cut* $\{e, f\}$ *if it satisfies one of the following properties:* (i) $\{x, y\} = \{a_1, b_1\}$*,* (ii) $\{x, y\} = \{a_2, b_2\}$*,* (iii) *the link* d *crosses one of the pairs* $\langle a_1, b_2\rangle$ *or* $\langle b_1, a_2\rangle$*.*

Observation 13.3. *Let* $\mathcal{C}$ *be a cactus and let* e *be a link that covers the 2-cut formed by consecutive edges incident to a vertex* v *on a cycle of the cactus. Then, the link* e *is projectively incident to the vertex* v*.*

Definition 13.9. (Intersection graph with respect to a cactus.) *Given a graph*

G, a cactus $\mathcal{C}$ and the set $\mathcal{L}$ of links, we define the intersection graph $H_{\mathcal{C}}$ of the cactus as follows. We have a vertex for every link and the vertices corresponding to 2 links e_1 and e_2 are adjacent if

- *there is a vertex in the cactus to which both these links are projectively incident or*
- *there are crossing pairs $\langle x_1, y_1\rangle$ and $\langle x_2, y_2\rangle$ which are projections of e_1 and e_2 respectively on the same cycle of the cactus.*

We define the **link-terminal intersection graph** *$I_{\mathcal{C},\mathcal{L}}$ as the graph obtained from $H_{\mathcal{C}}$ by adding to it the vertex set of G (these new vertices are called* **terminals** *and the other vertices,* **link-vertices***) and making each vertex adjacent to the vertices of $H_{\mathcal{C}}$ which correspond to the links projectively incident to this vertex.*

Definition 13.10. (Refining a cactus by an link set). *We define the operation of* **refining** *a cactus by the link e as the operation of identifying the vertices which e is projectively incident on and deleting any resulting self-loops. We define the refining of a cactus by a link set as the process of iteratively refining the cactus by each link in the set in an arbitrary order and deleting any resulting self-loops.*

The following observation is a clear consequence of the definition of refinements.

Observation 13.4. *If a link has a non-trivial projection onto ℓ cycles, then the refinement of this link creates ℓ more cycles in the cactus $\mathcal{C}'$.*

We now prove certain properties of link-intersection graphs which will be used both in our main structural characterisation as well as directly in our algorithms.

Observation 13.5. *Let $\mathcal{C}$ and $\mathcal{L}$ be a cactus and a set of links respectively and let $I_{\mathcal{C},\mathcal{L}}$ be the link-terminal intersection graph defined on the cactus and link set.*
(a) Every terminal in the link-terminal intersection graph is simplicial, that is, the neighborhood of the vertex induces a clique.
(b) If B is a connected induced subgraph such that the terminal set $T \subset V(B)$, then the induced subgraph of B containing only the link-vertices, $B[V \setminus T]$, is also connected.

Proof. By the definition, the graph $I_{\mathcal{C},\mathcal{L}}$ contains an edge between every pair of link-vertices that are incident on the same vertex in the cactus. Therefore, for any terminal in $I_{\mathcal{C},\mathcal{L}}$, the link-vertices corresponding to the links incident on this vertex form a clique in the link-intersection graph. Furthermore, by definition, the terminal is not adjacent to any other vertices in the link-terminal intersection graph. Therefore, the terminal is simplicial.

For the proof of the second statement, consider a shortest path $P = \{e, v_1, v_2, \ldots, v_r, f\}$ between link-vertices e and f in B. Let $e = v_0$ and $f = v_{r+1}$. Suppose that there exists a terminal $v_i \in T$ in this path P. Since v_i is simplicial, there exists an edge between vertices v_{i-1} and v_{i+1}, resulting in a shorter path $P' = \{e, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_r, f\}$ between the link-vertices e and f, a contradiction. Therefore P does not contain terminals. Therefore, we conclude that a shortest path between any two link-vertices in B is disjoint from the terminals implying that the induced subgraph of B containing only the link-vertices, $B[V \setminus T]$ is also connected. □

Lemma 13.6. *Consider a cactus $\mathcal{C}$ and a set $\mathcal{L}$ of links on this cactus. Let u and v be the vertices of a cycle C in the cactus such that they are terminals in the same connected component Z_1 of the graph $I_{\mathcal{C},\mathcal{L}}$. Let e be a link with a non-trivial projection onto the cycle C. If the link e crosses the pair $\langle u, v\rangle$, then the link-vertex e is also in the same connected component Z_1 in $I_{\mathcal{C},\mathcal{L}}$.*

Proof. Let the link e be projectively incident on the vertices x and y of the cycle C. Since the link e crosses the pair $\langle u, v\rangle$, we have that the pairs $\langle x, y\rangle$ and $\langle u, v\rangle$ are crossing. If the link-vertex e is in Z_1, then we are done. Suppose that e is not in Z_1. By the definition of $I_{\mathcal{C},\mathcal{L}}$, we know that x and y are not in Z_1 and furthermore, e also does not cross any link from Z_1. Now consider the two internally vertex disjoint paths P_1 and P_2 between x and y in the cycle C. Let $V(P_1)$ and $V(P_2)$ be the set of internal vertices of the paths P_1 and P_2 respectively. Since the pairs $\langle x, y\rangle$ and $\langle u, v\rangle$ are crossing, without loss of generality, we can assume that $u \in V(P_1)$ and $v \in V(P_2)$.

Let L be the set of links in Z_1 which have a non-trivial projection onto the cycle C. For $i \in \{1, 2\}$, let $L_i \subseteq L$ be the set of links such that every link in L_i is projectively incident to a vertex on $V(P_i)$. Suppose that $L_1 \cap L_2 \neq \emptyset$. Let $f = (a, b) \in L_1 \cap L_2$. Without loss of

generality, let $a \in V(P_1)$ and $b \in V(P_2)$. Then the links $e = (x, y)$ and $f = (a, b)$ cross and hence the corresponding link-vertices are adjacent in $I_{\mathcal{C},\mathcal{L}}$, which implies that e is in the same component as f, that is Z_1, a contradiction to our assumption that e was not in Z_1. Therefore we have $L_1 \cap L_2 = \emptyset$. Therefore, we conclude that $L = L_1 \uplus L_2$. This implies that any link in L is projectively incident to two vertices contained completely in $V(P_1)$ or $V(P_2)$.

Now, consider a shortest path, P_{uv} in Z_1 from the terminal u to terminal v. Let L' be the set of link-vertices which form the internal vertices of the path from u to v. Let $L'_1 \subset L_1$ and $L'_2 \subset L_2$. We have that $L'_2 = L' \setminus L'_1$. Note that since P_{uv} exists, $L'_1 \neq \emptyset$ and $L'_2 \neq \emptyset$. Furthermore, there are links $e \in L'_1$ and $f \in L'_2$ which are consecutive in P_{uv}, and therefore adjacent in $I_{\mathcal{C},\mathcal{L}}$. However, by definition, they are neither incident on a common vertex in the cactus nor do they cross, contradicting their adjacency in $I_{\mathcal{C},\mathcal{L}}$. Therefore we infer that the link-vertex e is in the same component and the pair $\langle u, v \rangle$, it crosses. This concludes the proof of the lemma. □

Lemma 13.7. *Given a cactus $\mathcal{C}$ and a set $\mathcal{L}$ of links, let $\{xy, uv\}$ be a 2-cut in the cactus. Let a and b be any two vertices in the distinct connected components of $\mathcal{C} \setminus \{xy, uv\}$. Also let B be an induced subgraph of $I_{\mathcal{C},\mathcal{L}}$ such that the terminal set $T \subset V(B)$. If a and b are in the same connected component Z_1 of B, then there exists a link-vertex e in Z_1 such that the link e covers the 2-cut $\{xy, uv\}$.*

Proof. Suppose that there is no link-vertex e in Z_1 such that the link e covers the 2-cut $\{xy, uv\}$. Let A_1 and A_2 be the components of $\mathcal{C}$ obtained by deleting the 2-cut $\{xy, uv\}$. Without loss of generality let $a \in V(A_1)$ and $b \in V(A_2)$. Now, consider a shortest path, P_{ab} in Z_1 from the terminal a to terminal b. Let L be the set of link-vertices which form the internal vertices of the path from a to b. By assumption, none of the link-vertices in L cover the cut $\{xy, uv\}$. Therefore, every link of L has both its end points either in A_1 or in A_2. Let $L_1 \subset L$ be the set of links with both end points in A_1 and let $L_2 = L \setminus L_1$. Note that since P_{ab} exists, $L_1 \neq \emptyset$ and $L_2 \neq \emptyset$. Furthermore, there are links $e \in L_1$ and $f \in L_2$ which are consecutive in P_{ab}, and therefore adjacent in $I_{\mathcal{C},\mathcal{L}}$. However, by definition, they are neither incident on a common vertex in the cactus nor do they cross, contradicting

their adjacency in $I_{\mathcal{C},\mathcal{L}}$. This completes the proof of the lemma. □

13.2 Relating augmenting sets to Steiner trees.

Lemma 13.8. *Let S be a set of links which is an augmenting set for the cactus $\mathcal{C}$. Let C be any cycle in the cactus $\mathcal{C}$. Every pair of link-vertices in $I_{\mathcal{C},\mathcal{L}}$ which correspond to a pair of links with a non-trivial projection in the cycle C appear in the same connected component Z_1 of $I_{\mathcal{C},\mathcal{L}}[S \cup T]$. Furthermore, the terminals in $I_{\mathcal{C},\mathcal{L}}$ corresponding to the vertices of the cycle C appear in Z_1.*

Proof. Let e_1 and e_2 be a pair of link-vertices with projections $\langle x_1, y_1 \rangle$ and $\langle x_2, y_2 \rangle$ in the cycle C and suppose that they occur in different connected components of $I_{\mathcal{C},\mathcal{L}}[S \cup T]$. Since link-vertices e_1 and e_2 cannot be adjacent, we infer that the projections of links e_1 and e_2 do not cross and also the vertices x_1, y_1, x_2, y_2 of C are all distinct. Without loss of generality, we assume that these vertices appear in the order x_1, x_2, y_2, y_1 in a fixed ordering of the cycle vertices. Let P_1 be the path in the cycle between x_1 and x_2 disjoint from $\{y_1, y_2\}$ and let P_2 be the path in the cycle between y_1 and y_2 disjoint from $\{x_1, x_2\}$. Let e_1 belong to the component Z_1 and e_2 belong to the component Z_2. Let u be the vertex in $V(P_1)$ closest to x_2 such that $u \in Z_1$. Similarly let v be the vertex in $V(P_2)$ which is closest to y_2, such that $v \in Z_1$. Note that u and v exist since x_1 and y_1 themselves are candidates to be u and v respectively. Let u' be the neighbor of u in the subpath of P_1 from u to x_2 and let v' be the neighbor of v in the subpath of P_2 from v to y_2. Note that u' can be the same as x_2 and v' can be the same as y_2.

Consider the 2-cut formed by the edges (u, u') and (v, v') and consider a link e in S which covers this cut. By our assumptions regarding u' and v', we have that e is not neighbor to any vertex in Z_1, in particular e not projectively incident to u or v. However, since e covers this 2-cut, it must be the case that e crosses the pairs $\langle u, v \rangle$ and $\langle u', v' \rangle$ (by Definition 13.8).

By Lemma 13.6, since u and v are each incident on a link in Z_1, we have that e is in Z_1 in the graph $I_{\mathcal{C},\mathcal{L}}[S \cup T]$. If e also crosses the pair $\langle x_2, y_2 \rangle$, then e_2 is in Z_1, a contradiction. Otherwise one of the end points of e is either in the subpath of P_1 from u to x_2 or in the

subpath of P_2 from v to y_2. But this contradicts the selection of u or v as the vertices closest to y_1 or y_2 which are also in Z_1. Therefore, we conclude that if a pair of links in S have a non-trivial projection in the same cycle, they appear in the same connected component of $I_{\mathcal{C},\mathcal{L}}[S \cup T]$.

Note that since S is an augmenting set for $\mathcal{C}$, every vertex in the cycle C has at least one link from S projectively incident on it. This is because every link that covers the 2-cut formed by consecutive edges of the cycle has to be projectively incident on the common vertex (by Definition 13.8). Therefore, we infer that every terminal in $I_{\mathcal{C},\mathcal{L}}$ that corresponds to a vertex of C is also in the component Z_1. This completes the proof of the lemma. □

Lemma 13.9. *Given a cactus $\mathcal{C}$ and a set $\mathcal{L}$ of links, consider the link-terminal intersection graph $I_{\mathcal{C},\mathcal{L}}$ of the cactus. Let T be the terminals in $I_{\mathcal{C},\mathcal{L}}$ and let X be the vertices in $I_{\mathcal{C},\mathcal{L}}$ which correspond to degree-2 vertices in the cactus. Then, a set S of links is an augmenting set for this cactus if and only if $I_{\mathcal{C},\mathcal{L}}[S \cup X]$ is connected.*

Proof. Consider the forward direction and let S be a set of links which is an augmenting set for this cactus. By Lemma 13.8, we know that every pair of link-vertices in $I_{\mathcal{C},\mathcal{L}}$ which correspond to a pair of links with a non-trivial projection in a particular cycle C appear in the same connected component of $I_{\mathcal{C},\mathcal{L}}[S \cup T]$. Observe that every vertex of X lies in a unique cycle and also there must be a link in S properly incident on it. Since such a link must have a non-trivial projection in this cycle, using Lemma 13.8, we conclude that every pair of vertices in X which lie in the same cycle of $\mathcal{C}$ are in the same connected component of $I_{\mathcal{C},\mathcal{L}}[S \cup T]$.

Now, consider an arbitrary pair t_1, t_2 of vertices in X which are not in the same cycle of the cactus. Let $a_1, \ldots, a_r$ be the cut-vertices of the cactus which appear in the path from t_1 to t_2 other than (possibly) the vertices t_1 and t_2. Let $a_0 = t_1$ and $a_{r+1} = t_2$. Then, every a_i and a_{i+1} occur in the same cycle and by Lemma 13.8 the corresponding vertices in $I_{\mathcal{C},\mathcal{L}}$ are in the same connected component of $I_{\mathcal{C},\mathcal{L}}[S \cup T]$. Since connectivity is an equivalence relation, this implies that the vertices corresponding to a_0 and a_{r+1} are also in the same connected component of $I_{\mathcal{C},\mathcal{L}}[S \cup T]$. Therefore we conclude that $I_{\mathcal{C},\mathcal{L}}[S \cup T]$ is connected and by Observation 13.5, this implies that $I_{\mathcal{C},\mathcal{L}}[S \cup X]$ is connected.

For the proof of the converse direction, we first show that if $I_{\mathcal{C},\mathcal{L}}[S \cup X]$ is connected, then $I_{\mathcal{C},\mathcal{L}}[S \cup T]$ is also connected. Let v be any vertex in $T \setminus X$. We know that v is a cut vertex in the cactus. Let x and y be neighbors of v in a cycle C of the cactus. Now consider the 2-cut $\{vx, vy\}$. Let A_1 and A_2 be the components of the cactus $\mathcal{C}$ obtained by deleting the 2-cut $\{vx, vy\}$. From Observation 13.1 we infer that $V(A_1) \cap X \neq \emptyset$ and $V(A_2) \cap X \neq \emptyset$. Let $a \in V(A_1) \cap X$ and $b \in V(A_2) \cap X$. Since a and b are in same connected component B in $I_{\mathcal{C},\mathcal{L}}[S \cup T]$, by Lemma 13.7, there exists a link e corresponding to a link-vertex in B such that e covers the 2-cut $\{vx, vy\}$. By Observation 13.3, the link e is projectively incident to v which implies that the terminal v is in the same component as X. Therefore we conclude that all the vertices of $T \setminus X$ are in the same component of as X in $I_{\mathcal{C},\mathcal{L}}[S \cup T]$. Therefore if $I_{\mathcal{C},\mathcal{L}}[S \cup X]$ is connected, then $I_{\mathcal{C},\mathcal{L}}[S \cup T]$ is also connected.

Now, suppose that S is not an augmenting set for the cactus and let e_1, e_2 be a 2-cut in a cycle of the cactus which is not covered by S. Let A_1 and A_2 be the components of the cactus $\mathcal{C}$ obtained by deleting the 2-cut $\{e_1, e_2\}$. By Observation 13.1 we can infer that $V(A_1) \cap X \neq \emptyset$ and $V(A_2) \cap X \neq \emptyset$. Let $a \in V(A_1) \cap X$ and $b \in V(A_2) \cap X$. Since a and b are in same component in $I_{\mathcal{C},\mathcal{L}}[S \cup T]$ (the entire subgraph being connected by assumption), by Lemma 13.7, there exists a link e corresponding to a link-vertex in S such that e covers the 2-cut $\{e_1, e_2\}$. This contradicts our assumption that S is not an augmenting set for the cactus $\mathcal{C}$. This completes the proof of the lemma. □

13.3 An improved Algorithm for Cactus Augmentation

Theorem 13.11. *The* Weighted Edge Connectivity Augmentation of a Cactus *problem can be solved in time* $\mathcal{O}^*(9^k)$.

Proof. Given an instance $(G = (V, E), \mathcal{C}, \mathcal{L}, k)$, we construct the link-terminal intersection graph I_G. Let X be the vertices of I_G which correspond to degree-2 vertices of $\mathcal{C}$. By Lemma 13.9, we have that a set S of links is an augmenting set if and only if $I_G[S \cup X]$ has a connected component containing X. Therefore, we conclude that a min-cost augmenting set of size at most k exactly corresponds to a min-cost steiner tree of size at most $k + |X|$ in I_G

containing the set X. The dynamic programming algorithm of Dreyfuss and Wagner [DW71] can be modified to compute the min-cost steiner tree of size at most k in time $\mathcal{O}^*(3^{|X|})$. Since we already know that any augmenting set of links has size at least $|X|/2$, we return No if $|X| > 2k$. Hence, we may assume that $|X| \leq 2k$ and therefore, we have an algorithm for Min Cost augmentation of a cactus running in time $\mathcal{O}^*(9^k)$. □

Since the problem of augmenting a tree can be reduced to augmenting a cactus simply by replacing every tree edge with a pair of parallel edges, we have the following corollary.

Corollary 13.10. *The* Weighted Edge Connectivity Augmentation of a Tree *problem can be solved in time* $\mathcal{O}^*(9^k)$.

Once again, using the result of Dinits et al. referred to in earlier sections, w-Aug-One can, in polynomial time be reduced to Weighted Edge Connectivity Augmentation of a Tree or Weighted Edge Connectivity Augmentation of a Cactus depending on the parity of the connectivity of the given graph, hence proving Theorem 13.1.

Chapter 14

Parameterized Algorithms for Network Augmentation II

In this chapter, we will focus on the parameterized complexity of the following problem.

$m-k$ Augmentation by One	Parameter: k
Input: $(G, \mathcal{L}, k)$ where G is a λ-edge connected, $\mathcal{L}$ is a set of edges, called links, $G+\mathcal{L}$ is $(\lambda+1)$-edge connected, and k a positive integer.	
Question: Is there a set of k links in $\mathcal{L}$ whose deletion from $G+\mathcal{L}$ maintains $(\lambda+1)$-edge connectivity?	

Intuitively, $m-k$ Augmentation by One seems to be harder than w-Aug-by-One. For example, if we want to augment a tree T to a 2-edge connected graph, the number of links we have to add is at least half the number of leaves of T. By handling degree two vertices in appropriate way, this almost immediately brings us to a polynomial kernel. But when we want to delete links from $\mathcal{L}$ in order to keep a minimum 2-edge connected supergraph of T, there is no lower bounds on the number of deleted links, and this makes the problem much harder. Similar difference can be observed for graphs of larger connectivity. Not surprisingly, to solve $m-k$ Augmentation by One, we need completely different ideas, and we obtain the following theorem.

Theorem 14.1. *$m-k$ Augmentation by One is solvable in time $2^{\mathcal{O}(k)}|V|^{\mathcal{O}(1)}$.*

In other words, we establish that $m-k$ Augmentation by One is FPT. The next natural question following the establishment of parameterized tractability of a problem is if this problem admits a "polynomial kernel". We answer this question affirmatively.

Theorem 14.2. *$m-k$ Augmentation by One admits a kernel with $12k$ vertices and $3k$ links for odd λ and a kernel with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ links for even λ, where λ is the connectivity of the input graph G.*

As before, the result of Dinits et al. [DKL76] allows us to assume without loss of generality that either $\lambda = 1$ and G is a tree, or $\lambda = 2$ and G is a cactus graph. Then we design preprocessing rules based on several non-trivial structural lemmas regarding No instances of the problem. We then show that applying these preprocessing rules exhaustively results in an equivalent instance where the lengths of the cycles in the graph G is bounded linearly in the parameter. In our final step, we show that this property, along with those proved beforehand, gives a linear bound on the tree width of the link-terminal intersection graph, where it suffices to compute an optimum Steiner tree. Finally, we apply an algorithm for Steiner Tree, with a running time $2^{O(t)}|V(G)|^{O(1)}$ [BCKN13], where t is the treewidth of the input graph, to obtain the solution. The techniques we use for the kernels for $m-k$ Augmentation by One depend on the parity of the connectivity λ. In the case when λ is odd, we construct an alternate auxiliary graph with the set of links as the vertex set and prove two properties regarding this graph. The first is that this graph has a constant degeneracy and the second is that a vertex cover in this graph corresponds to an augmenting set in the input instance. Combining these two properties, we are able to conclude that if the size of the instance exceeds a certain constant factor of the parameter, then the input instance is a Yes instance, thus obtaining a linear kernel. For the case when λ is even, we introduce an additional preprocessing rule to the ones already used and show that these rules together suffice to bound the size of a given instance quadratically in the parameter. Our results are summarized in the following table.

Problem Name	Algorithm	Kernel
$m-k$ Augmentation by One (odd λ)	$O^*(8^k)$	$12k$ vertices and $3k$ links
$m-k$ Augmentation by One (even λ)	$O^*(2^{\mathcal{O}(k)})$	$\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ links

Table 14.1: The table gives a summary of our results, where the $\mathcal{O}^*(\cdots)$ notation hides polynomial factors. The constants hidden by the $\mathcal{O}(\cdots)$ notation in the kernel sizes are independent of λ.

14.1 Single Exponential FPT algorithm

In this section we give a single exponential FPT algorithm for $m-k$ Augmentation by One. As stated earlier, when λ is odd, we have a kernel with at most $3k$ links. Hence we can easily obtain a FPT algorithm with a running time of $\mathcal{O}^*(2^{8k})$ in this case. The case where λ is even, is more challenging. We show that we can preprocess the input instance in polynomial time, to reduce the treewidth of the associated link-intersection graph to $\mathcal{O}(k)$. Then by applying a single exponential FPT algorithm for Steiner Tree, parameterized by treewidth, we solve the instance in claimed running time.

We first consider the case of $\lambda = 2$ i.e. augmenting the connectivity of a cactus, which is called $m-k$ Cactus Augmentation. In this section, we give a set of preprocessing rules using which we will obtain a bound on the treewidth of the link-intersection graph. We begin by revisiting the operation of refining a cactus and proving certain properties of this operation which will play a crucial role in the statements as well as correctness of our preprocessing rules.

Lemma 14.1. *Let F be an augmenting set of links for the given cactus $\mathcal{C}$ and let $L \subseteq F$. Then, $F \setminus L$ is an augmenting set for the cactus obtained by refining $\mathcal{C}$ by the set L. Conversely, if refining $\mathcal{C}$ by a set L of links results in a cactus $\mathcal{C}'$, then for any augmenting set F of $\mathcal{C}'$, $F \cup L$ is an augmenting set of $\mathcal{C}$.*

Proof. We begin by proving the following claim.

Claim 1. *Let $\mathcal{C}$ be a cactus and $L \subseteq \mathcal{L}$ be a link-set. Let $\mathcal{C}'$ be the cactus obtained by refining the cactus $\mathcal{C}$ by the link-set L. Then the 2-cuts in the refined cactus $\mathcal{C}'$ are precisely those 2-cuts in $\mathcal{C}$ that are not covered by the link-set L.*

Proof. We begin by assuming that the cactus $\mathcal{C}$ is a single cycle C. Consider a link $e = (x, y)$ in the link-set. Since $\mathcal{C} = \{C\}$, we have that e is a chord of the cycle C. Let P_1 and P_2 be the components formed by deleting the vertices x and y from C. For $i \in \{1, 2\}$, let $C_i = C[V(P_i) \cup \{x, y\}]$. That is C_1 and C_2 are the paths xP_1y and xP_2y respectively. Note that that link $e = (x, y)$ covers all the 2-cuts of the form $\{g, h\}$, where $g \in E(C_1)$ and $h \in E(C_2)$ where $E(C_1)$ and $E(C_2)$ are the edges involved in the paths C_1 and C_2 respectively. Therefore, the only 2-cuts which are not covered by e are of the form $\{g, h\}$ where $g, h \in E(C_1)$ or $g, h \in E(C_2)$. However, observe that all pairs of such edges appear together in a cycle of the refined cactus and hence these 2-cuts are indeed preserved by the operation of refinement. Furthermore, observe that it follows from the definition of refinement that for every 2-cut $\{e_1, e_2\}$ covered by this chord, e_1 and e_2 are in distinct cycles of the refined cactus, and therefore no longer form a 2-cut. Hence, we conclude that the 2-cuts of the resulting cactus C' are exactly the 2-cuts not covered by e.

Now, consider the case when $\mathcal{C}$ is not a single cycle. Then, refining a link $e = (x, y)$ which does not lie within a single cycle is clearly the same as refining each cycle of the cactus by considering the projection of e onto this cycle as a chord of this cycle. Since any 2-cut in the cactus lies in a single cycle, if e covers this 2-cut, then the projection of e on this cycle covers this 2-cut. But we have already shown that the 2-cuts within each cycle which are not covered by a chord are preserved upon refining the cycle by the chord. Therefore, we conclude that refinement of the cactus by a link preserves the 2-cuts not covered by this link. It follows from the definition of refinement that the 2-cuts covered by this link are indeed absent in the refined cactus. Finally, applying this argument iteratively for every link in the given link-set L, the claim is proved. □

We now complete the proof of the lemma. Suppose that F is an augmenting set of links for $\mathcal{C}$. Since F covers all the 2-cuts in the cactus, by the above claim, we have that refining $\mathcal{C}$ by the link-set F results in a cactus $\mathcal{C}'$ with no 2-cuts, implying that $\mathcal{C}'$ is a singleton vertex. Now, consider $L \subseteq F$ and let $\mathcal{C}''$ be the cactus obtained from $\mathcal{C}$ by refining it by L. Since further refining $\mathcal{C}''$ by $F \setminus L$ results in a singleton vertex, we conclude that $F \setminus L$ is indeed an augmenting set for $\mathcal{C}''$.

Conversely, suppose that for some set L of links, $\mathcal{C}'$ is the cactus obtained from $\mathcal{C}$ by refining it with L and F is an augmenting set of $\mathcal{C}'$. Since F covers every 2-cut in $\mathcal{C}'$ by the above claim, F covers every 2-cut in $\mathcal{C}$ that is not covered by L. Therefore, we conclude that $F \cup L$ covers every 2-cut in $\mathcal{C}$. This completes the proof of the lemma. □

Observation 14.2. *A cycle of length ℓ requires at least $\ell/2$ links and can be augmented with at most $\ell - 1$ links. Furthermore, the upper bound is tight when the given set of links on the cycle is laminar.*

Proof. Since every vertex of the cycle needs a link in the augmenting set which is projectively incident on it and any link can only be projectively incident on 2 vertices of a cycle, the lower bound follows. For the upper bound, the proof is by induction on the length of the cycle. For $\ell = 2$, the statement holds. Suppose that $\ell > 2$. Consider a link e such that refining the cycle by e results in a cactus with 2 non-trivial cycles. If such a link does not exist, then every link is parallel to a cycle edge and we can pick any consecutive $\ell - 1$ of them. Let the 2 cycles be of length ℓ_1 and ℓ_2 with $\ell_1 + \ell_2 = \ell$. By Lemma 14.1, we have that the 2-cuts left uncovered by e are present in one of these 2 cycles and that any augmenting set for the resulting cactus, along with the link e is an augmenting set for the cycle. By the induction hypothesis, we have that the cycles can be augmented with $\ell_1 - 1$ and $\ell_2 - 1$ links respectively. Therefore, the cycle can be augmented with $\ell_1 - 1 + \ell_2 - 1 + 1 = \ell - 1$ links. Observe that the same argument shows that if the set of links on the cycle is laminar then the minimum augmenting set also has size at least $\ell - 1$. This completes the proof of the statement. □

We now move to the description of our polynomial time preprocessing rules. Since it will be clear from the statements that these rules can be implemented in polynomial time, we will ignore this part of the analysis.

Reduction Rule 14.3. *If there is a 2-cut in the cactus which is covered by a unique link e, then refine the cactus by this link.*

Observation 14.4. *If Reduction Rule 14.3 does not apply then every 2-cut in the cactus is covered by at least two links. Furthermore, if ℓ is the length of a cycle in the cactus, then*

it has at least ℓ links projectively incident on it.

Proof. Since the rule does not apply, every 2-cut in the cactus must be covered by at least 2 links. In particular, for any vertex v and any cycle C containing this vertex, the 2-cut formed by the 2 edges of C incident on v is also covered by at least 2 links. However, any link which covers this 2-cut must be projectively incident on v. Therefore, every vertex in a cycle has at least 2 links projectively incident on it, which implies that a cycle of length ℓ has at least ℓ links projectively incident on it. □

Reduction Rule 14.5. *Let x be a vertex in the cactus such that no link has x as an end point and the only edges of the cactus incident on it are the four edges of two 2-circuits xy and xz. Then contract the 2-circuit xy.*

The correctness of the first rule directly follows from Lemma 14.1 and the correctness of the second rule can be argued as follows. Since the links which cover the 2-cut formed by the 2-circuit xz are precisely the links which cover the 2-cut formed by the 2-circuit xy, it suffices to preserve exactly one of them in the instance. The following reduction rule is a generalization of the previous rule.

Reduction Rule 14.6. *Let x be a vertex of a cycle in the cactus such that the only edges of the cactus incident on it are the edges of the cycle and two edges of a 2-circuit xy. Furthermore, suppose that the only links projectively incident on x are those that are incident on y. Then contract the 2-circuit xy.*

Lemma 14.7. *Let $(\mathcal{C}, \mathcal{L}, k)$ be an instance on which Reduction Rule 14.3 does not apply. If there is a vertex in the cactus which has more than $4k$ links projectively incident on it, then the given instance is a* Yes *instance.*

Proof. Consider a vertex x and the set L of links projectively incident on x. We prove by induction on $|L|$ that there is a minimal augmenting set which contains at most $3/4|L|$ of the links incident on x. We now consider the base case of our induction, that is, $|L| = 4$. Let $e \in \mathcal{L}$ be a link incident on x. Observe that since Reduction Rule 14.3 does not apply, $\mathcal{L} \setminus \{e\}$ indeed covers all the 2-cuts in $\mathcal{C}$.

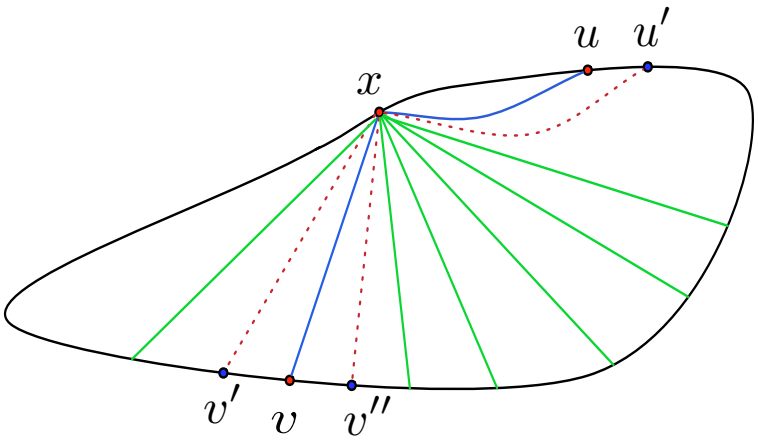

Figure 14.1: An illustration of the proof of Lemma 14.7.

For the induction step, consider the Eulerian tour of the cactus starting and ending at x. For any link $(u, v) \in L$ where $u, v \neq x$, i.e (u, v) is projectively incident on x, we represent it as 2 links in the Eulerian tour by adding a link between x and the first occurrence of u in the tour. and one between x and the first occurrence of v (see Figure 14.1). Similarly, for every link $(x, v) \in L$, we simply represent it as a link between the origin x and the first occurrence of v in the tour.

Consider the first vertex u in the tour which has a link to x in the representation. Suppose that this link represents the link $e = (u, v) \in L$ where $v \neq x$. Let v' be the first vertex on the tour after v which has a link projectively incident on itself and x, call this link $e(v')$ and v'' be the last vertex on the tour before v which has a link projectively incident on itself and x, call this link $e(v'')$. Similarly, let u' be the first vertex on the tour after u which has a link projectively incident on itself and x, call this link $e(u')$. If $u' = v$ or $v'' = u$ or $v' = x$, we let the corresponding links by undefined. Let $L' = \{e, e(u'), e(v'), e(v'')\}$. We claim that in the cactus $\mathcal{C}'$ obtained from $\mathcal{C}$ by refining the set $L' \setminus \{e\}$, none of the links in $L \setminus L'$ cover a 2-cut covered by e. We prove this by showing that every 2-cut covered by e and a link in $L \setminus L'$ is also covered by a link in $L' \setminus \{e\}$. Let P_1 be the part of the tour between the starting vertex and the first occurrence of u, P_2 be the part of the tour between the first occurrence of u and the first occurrence of v and let P_3 be the rest of the tour. Let Q_1 be part of the tour between u and u', Q_2 be the part of the tour between

v'' and v and let Q_3 be the part of the tour between v and v'. Observe that any 2-cut covered by e must have an edge in P_1 and an edge in P_2 or an edge in P_3 and an edge in P_2. Clearly, no link in $L \setminus L'$ can cover a 2-cut one of whose edges is in P_1 and the other in Q_1 or a 2-cut consisting of an edge in Q_2 and one in Q_3. Now, any 2-cut consisting of an edge in P_1 and one in $P_2 \setminus Q_1$ is covered by the link $e(u')$. Similarly, any 2-cut which consists of an edge in Q_3 and an edge in $P_2 \setminus Q_2$ is covered by $e(v'')$ and any 2-cut with an edge in Q_2 and one in $P_3 \setminus Q_3$ is covered by $e(v')$. We therefore, conclude that every 2-cut covered by e and a link in $L \setminus L'$ is also covered by a link in $L' \setminus \{e\}$, implying that in the cactus $\mathcal{C}'$, none of the links in $L \setminus L'$ cover a 2-cut covered by e. Therefore, e, along with an augmenting set for $\mathcal{C}'$ will be an augmenting set for $\mathcal{C}$.

By the induction hypothesis, we have that there is a minimal augmenting set F' for $\mathcal{C}'$ which contains $3/4|L \setminus L'|$ links from $L \setminus L$. Since none of these links in $L \setminus L'$ cover a 2-cut covered by e, $F' \cup \{e(u'), e(v'), e(v'')\}$ is indeed an augmenting set for $\mathcal{C}$ of size at most $3/4|L|$. Therefore, we conclude that if x has more than $4k$ links projectively incident on it, then the given instance is a Yes instance. □

Lemma 14.8. *Let C be a cycle of cactus. Let the number of links needed to augment the cycle C be ℓ. If the number of links projectively incident on the cycle C is greater than $\ell + 4k$, then the given instance is a Yes instance.*

Proof. Let L be a set of ℓ links which augment the cycle and consider the cactus refined by this link-set. Then, we get a vertex which is projectively incident to at least $4k$ links in the refined cactus, which is a Yes instance by Lemma 14.7. □

From Observation 14.2 and Lemma 14.8, we infer that if the number of links projectively incident on a cycle C is greater than $(t-1) + 4k$ where t is the length of the cycle, then the given instance is a Yes instance.

Greedy-augmenting-set($\mathcal{S}$, C):

1. Start with $\mathcal{T} = \emptyset$, $\mathcal{C} = C$ and $\mathsf{Cycles}(\mathcal{C}) = 1$.

   (Here $\mathsf{Cycles}(\mathcal{C})$ denotes the number of cycles in the cactus $\mathcal{C}$)

2. Repeat until there are no more links in $\mathcal{S} \setminus \mathcal{T}$ that can be added to $\mathcal{T}$ or until $\mathcal{T} = 8k$:

   (a) If there exists $e \in \mathcal{S} \setminus \mathcal{T}$ such that $\mathsf{Cycles}(\mathcal{C}') \geq \mathsf{Cycles}(\mathcal{C}) + 2$, where $\mathcal{C}'$ is the cactus obtained by refining $\mathcal{C}$ by the link e, then $\mathcal{T} = \mathcal{T} \cup \{e\}$ and set $\mathcal{C} = \mathcal{C}'$.

   (b) Else if there exist $e, f \in \mathcal{S} \setminus \mathcal{T}$, such that $\mathsf{Cycles}(\mathcal{C}') \geq \mathsf{Cycles}(\mathcal{C}) + 3$, where $\mathcal{C}'$ is the cactus obtained by refining $\mathcal{C}$ by the link-set $\{e, f\}$, then $\mathcal{T} = \mathcal{T} \cup \{e, f\}$ and set $\mathcal{C} = \mathcal{C}'$.

Reduction Rule 14.9. *Run the Greedy-augmenting-set algorithm with the set $\mathcal{S}$ of links projected on a cycle C. When the algorithm stops, if $|\mathcal{T}| = 8k$, then say* Yes.

Lemma 14.10. *Reduction Rule 14.9 is safe.*

Proof. Let C be a cycle of length ℓ on which the greedy algorithm was applied. Observe that in the first iteration of the algorithm, 3 new cycles are created after the refinement using some two links. Thus $\mathsf{Cycles}(\mathcal{C}) = 3+1 = 4$ after the first iteration. After that in each iteration at least $\frac{3}{2}x$ new cycles are created, where x is the number of links refined in that iteration. When the algorithm stops, let $t = \mathsf{Cycles}(\mathcal{C})$ be the number of cycles resulting from the refinement of the original cycle C. Therefore $t \geq \frac{3}{2}(|\mathcal{T}| - 2) + 4 = |\mathcal{T}| + \frac{|\mathcal{T}|+2}{2}$.

We know that any cycle of length ℓ_i requires at most $\ell_i - 1$ augmenting links (by Observation 14.2). Thus, we need $\sum_{i=1}^{t} \ell_i - 1 = \sum_{i=1}^{t} \ell_i - t \leq \sum_{i=1}^{t} \ell_i - (|\mathcal{T}| + \frac{|\mathcal{T}|+2}{2})$ links to augment all of the new cycles. Since we have used $\mathcal{T}$ links for the refinement in the Greedy Algorithm, the total number of links needed to augment the original cycle C is at most $\sum_{i=1}^{t} \ell_i - (|\mathcal{T}| + \frac{|\mathcal{T}|+2}{2}) + |\mathcal{T}| = \sum_{i=1}^{t} \ell_i - (\frac{|\mathcal{T}|+2}{2})$. But $\sum_{i=1}^{t} \ell_i = \ell$. Therefore we have an upper bound of $\ell - (\frac{|\mathcal{T}|+2}{2})$ on the number of links needed to completely augment this cycle C. Since there are at least ℓ edges projectively incident on the cycle (By Observation 14.4),

once the cycle is augmented by refining these $\ell - (\frac{|\mathcal{T}|+2}{2})$ links, the cycle will be contracted to a single vertex which will have at least $|\mathcal{T}|/2 = 8k/2 = 4k$ links projectively incident on it. Therefore, by Lemma 14.7, the given instance is indeed a Yes instance. □

14.1.1 Reducing Laminar Strips

Definition 14.3. *Given a cactus $\mathcal{C}$ and a set of links $\mathcal{L}$ on the cactus, let C be a cycle in the cactus. Consider a path $P = v_0, v_1, v_2 \dots v_t, v_{t+1}$ of the cycle C where $v_0 \neq v_{t+1}$. Let $\mathcal{L}[P] \subseteq \mathcal{L}$ be the set of links which have both endpoints in $V(P)$ and let $\mathcal{L}[P_1] \subseteq \mathcal{L}$ be the set of links projectively incident on the vertex set $V(P_1) = V(P) \setminus \{v_0, v_{t+1}\} = \{v_1, v_2, \dots, v_t\}$, that is the internal vertices of the path P. We call P a **laminar strip** if the following conditions are satisfied (see Figure 14.2).*

- *$\mathcal{L}[P]$ is laminar.*
- *$\mathcal{L}[P_1] \subseteq \mathcal{L}[P]$.*
- *Every edge in the path P is covered by at least one link in $\mathcal{L}[P]$. That is for every edge (v_i, v_{i+1}), there is a link in $\mathcal{L}[P]$ with endpoints v_j and v_r where $j \leq i < r$.*

We call $\mathcal{L}[P]$ the set of links associated with the laminar strip P and define the length of a laminar strip P as the number of edges in the path P.

Lemma 14.11. *Given a cactus $\mathcal{C}$ and a set of links $\mathcal{L}$, let P be a laminar strip of length $t+1$ with endpoints x and y. Let C_1 be the cycle obtained from P by adding an edge (x, y) and let C_2 be the cycle obtained from P by identifying x and y. Let F be any minimal augmenting set and let $L = \mathcal{L}[P] \cap F$. Then, either $|L| \geq t+1$ and L is an augmenting set for C_1 or $|L| \geq t$ and L is an augmenting set for C_2.*

Proof. Observe that since P is a laminar strip, the set $\mathcal{L}[P]$ is laminar in both C_1 and C_2. Suppose that L is not an augmenting set for C_2. Consider a 2-cut $\{e_1, e_2\}$ in C_2 not covered by L. Then, the 2-cut $\{e_1, e_2\}$ is also not covered by L in the cactus $\mathcal{C}$. Since P is a laminar strip, no link disjoint from $\mathcal{L}[P]$ can cover this 2-cut, implying that F is

not an augmenting set for $\mathcal{C}$, a contradiction. Therefore, we conclude that L is indeed an augmenting set for C_2. Since the length of C_2 is $t+1$, by Observation 14.2, $|L| \geq t$. Furthermore, if L is also an augmenting set for C_1, then Observation 14.2 implies that $|L| \geq t+1$. □

Given the above lemma, we define an operation we refer to as *shrinking a laminar strip*. The intuition behind this operation is as follows. Since the intersection of any minimal augmenting set with the laminar strip has one of only 2 possible "types" according to the previous lemma, we replace each laminar strip by an "equivalent" laminar strip of constant size. More precisely,

Definition 14.4. ***Shrinking a laminar strip.*** *Given an instance $(\mathcal{C}, \mathcal{L}, k)$ of $m-k$* Cactus Augmentation*, let P be a laminar strip and $\mathcal{L}[P]$ be the associated link-set. We define the process of* shrinking *the laminar strip P as follows. We contract the path P in the cycle to the edge xy (see Figure 14.2), which we refer to as the path P' and change the link-set $\mathcal{L}$ to $\mathcal{L}' = (\mathcal{L} \setminus \mathcal{L}[P]) \cup \{(x,y)\}$. If $|\mathcal{L}[P]| = \ell$, then set $k' = k - \ell + t + 1$. Finally, return $(\mathcal{C}', \mathcal{L}', k')$ where $\mathcal{C}'$ and $\mathcal{L}'$ are the resulting cactus and link-set respectively.*

Reduction Rule 14.12. *Given an instance $(\mathcal{C}, \mathcal{L}, k)$ of $m-k$* Cactus Augmentation*, if a cycle of the cactus contains a laminar strip P whose endpoints do not coincide, then shrink it.*

Lemma 14.13. *Reduction Rule 14.12 is safe.*

Proof. Let $(\mathcal{C}', \mathcal{L}', k')$ be the instance obtained by applying Reduction rule 14.12 to the instance $(\mathcal{C}, \mathcal{L}, k)$. Let P be the laminar strip on which the rule was applied, x and y be its endpoints, let $|P| = t+1$ and $|\mathcal{L}[P]| = \ell$. Then, we have that $k' = k - \ell + t + 1$ and $|\mathcal{L}'| = |\mathcal{L}| - \ell + 1$. We now show that $(\mathcal{C}, \mathcal{L}, k)$ is a Yes instance if and only if $(\mathcal{C}', \mathcal{L}', k')$ is a Yes instance.

Suppose $(\mathcal{C}, \mathcal{L}, k)$ is a Yes instance, F be a minimum augmenting set for $\mathcal{C}$, and $L = F \cap \mathcal{L}[P]$. Then $|F| \leq |\mathcal{L}| - k$. Consider the cycle C_1 obtained from P by adding an edge (x,y) and the cycle C_2 obtained from P by identifying x and y. By lemma 14.11, we have that either $|L| \geq t+1$ and L augments C_1 or $|L| \geq t$ and L augments C_2.

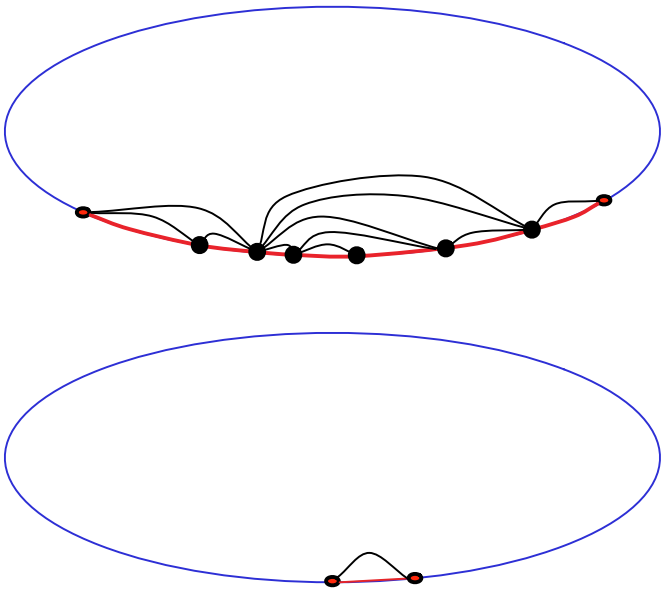

Figure 14.2: An illustration of a laminar strip (the red path) and the operation of shrinking it.

In the former case, we define the set $F' = (F \setminus L) \cup \{(x, y)\}$ and otherwise, we define set $F' = F \setminus L$. We claim that $|F'| \leq |\mathcal{L}'| - k'$ and that it is an augmenting set for $\mathcal{C}'$. Clearly, $|F'| \leq |F| - t$. Therefore, in order to prove that $|F'| \leq |\mathcal{L}'| - k'$, it suffices to prove that $|\mathcal{L}| - k - t \leq |\mathcal{L}'| - k'$, which follows from the fact that $k' = k - \ell + t + 1$ and $|\mathcal{L}'| = |\mathcal{L}| - \ell + 1$.

If F' is not an augmenting set for $\mathcal{C}'$, then there is a 2-cut in $\mathcal{C}'$ which is not covered by F'. Since the only links of F not present in F' are those in L, it must be case that any such uncovered 2-cut in $\mathcal{C}'$ contains the edge (x, y) and a second edge, say e. Since this 2-cut is not covered by F', we infer that F' does not contain the link (x, y). Therefore, by the definition of F' it must have been the case that L is *not* an augmenting set for C_1. This implies that any 2-cut comprising an edge in P and out outside P in $\mathcal{C}$, must therefore, be covered by a link in $F \setminus L$. However, any such link is present in F' and will cover the 2-cut $\{e, (x, y)\}$ in $\mathcal{C}'$, a contradiction. Hence, we conclude that F' is indeed an augmenting set for $\mathcal{C}'$ and this completes the forward direction.

For the converse direction, suppose $(\mathcal{C}', \mathcal{L}', k')$ is a Yes instance, F' be a minimum augment-

ing set for $\mathcal{C}'$, and $L' = F' \cap \mathcal{L}'[P]$. Then $|F'| \leq |\mathcal{L}'| - k$. Furthermore, let $L_1, L_2 \subseteq \mathcal{L}[P]$ be minimum augmenting sets for C_1 and C_2 respectively. By Observation 14.2 and the fact that $\mathcal{L}[P]$ is laminar in C_1 and C_2, we have that $|L_1| = t + 1$ and $|L_2| = t$.

We now define the set F as follows. If F' contains the link (x, y) then we define $F = F' \cup L_1$ and $F = F' \cup L_2$ otherwise. We now claim that $|F| \leq |\mathcal{L}| - k$ and that it is an augmenting set for $\mathcal{C}$. Clearly, $|F| = |F'| + t$. Therefore, in order to prove that $|F| \leq |\mathcal{L}| - k$, it suffices to prove that $|F'| + t \leq |\mathcal{L}'| + \ell - 1 - k$, which follows from the fact that $k' = k - \ell + t + 1$.

If F is not an augmenting set for $\mathcal{C}$, then there is a 2-cut in $\mathcal{C}$ which is not covered by F. Since every 2-cut with both edges disjoint from $E(P)$ is also present in $\mathcal{C}'$ and F' covers these 2-cuts, we conclude that F covers all 2-cuts with both edges disjoint from $E(P)$. Therefore, any 2-cut not covered by F in $\mathcal{C}$ must contain an edge, say e' in $E(P)$. Since both L_1 and L_2 cover every 2-cut with both edges in $E(P)$, it must be the case that exactly one of the edges of the 2-cut, say e'', lies outside P. Since L_1 does cover such 2-cuts, we can conclude that L_1 is not contained in F, implying that L_2 is contained in F. By the definition of F this implies that F' does not contain the link (x, y). Let l be a link in F' which covers the cut $\{e'', (x, y)\}$. Since l is also present in F and it covers 2-cuts formed by e'' and any edge in $E(P)$, F is an augmenting set for $\mathcal{C}$, a contradiction. Therefore, we conclude that F is indeed an augmenting set for $\mathcal{C}$ and this completes the proof of the lemma. □

Now we can assume that there are no laminar strips of length greater than one in any cycle of the cactus.

14.1.2 Bounding the cycle lengths and tree-width of the link-intersection graph

Lemma 14.14. *If the Reduction Rules 1-5 do not apply and there is a cycle in the cactus which has length $> 67k$, then the given instance is a* Yes *instance.*

Proof. Let C be a cycle in the cactus, $\mathcal{L}[C]$ be the set of links which are projectively incident on C. We use $deg_{\mathcal{L}}(v)$ to denote the number of links projectively incident on v.

We partition the vertices of $V(C)$ as follows and proceed to establish a bound on each of them.

$V_3 = \{v \in V(C) | deg_{\mathcal{L}}(v) \geq 3\}$

$V_2^a = \{v \in V(C) | deg_{\mathcal{L}}(v) = 2$ and v is a cut vertex in the cactus$\}$

$V_2^b = \{v \in V(C) | deg_{\mathcal{L}}(v) = 2$ and v is not a cut vertex$\}$

1. First, suppose that $|V_3| \geq 8k$. Since Reduction Rule 14.3 does not apply, every vertex in the cactus is projectively incident on at least 2 links. Then, the sum of the number of links projectively incident on each vertex in the cycle C is at least $2(\ell - 8k) + 3 \cdot 8k$ where ℓ is the length of the cycle. Therefore, the number of links which are projectively incident on this cycle is at least half this number, that is $\ell - 8k + 12k = \ell + 4k$, and by Lemma 14.8, we have that the given instance is a Yes instance. Therefore, we may assume that $|V_3| < 8k$.

2. Now, suppose that $|V_2^a| \geq k$. Consider a vertex $v \in V_2^a$. The vertex v is a cut-vertex in the cactus. Let a and b be the neighbors of v in the cycle C. Let $\mathcal{C}'$ be a sub-cactus of $\mathcal{C}$ formed by the component containing the vertex v, obtained by deleting the edges (v, a) and (v, b) in the cactus $\mathcal{C}$. We claim that there exists a link which is properly incident on two vertices of $\mathcal{C}'$. Suppose not. Then every link that is properly incident on a vertex of $\mathcal{C}'$ has another endpoint outside it. This implies that every link has a non-trivial projection onto the cycle C. But we know that only two links are projectively incident on v in C. Thus only two links are incident on the vertices of $\mathcal{C}'$. But note that by Observation 13.1, $\mathcal{C}'$ contains a vertex u of degree 2 in $\mathcal{C}$. We also know that at least two links are properly incident on u. This implies that every link incident on u is also projectively incident on v and v does not have any other link incident on it. Furthermore, no other vertex in $\mathcal{C}'$ have a link properly incident on it. Therefore u and v would be part of a 2-circuit or a chain of 2-circuits with no links incident on any of the intermediate vertices. Then in this situation, either Reduction Rule 14.5 or Reduction Rule 14.6 would have applied. Therefore we infer that there is a link which is properly incident on two vertices of $\mathcal{C}'$. Let us

call this link e_v. Observe that we have a distinct such link corresponding to every vertex in V_2^a and furthermore, there is no 2-cut in the cactus covered by a pair these links and hence we can still augment the cactus using the links disjoint from this set. This implies that the given instance is a Yes instance if $|V_2^a| \geq k$. Therefore, we may assume that $|V_2^a| < k$.

3. We now proceed to bound the set V_2^b which requires more involved arguments. Since Reduction Rule 14.9 does not apply, when the greedy algorithm terminated, we have a link-set $\mathcal{T}$ of at most $8k$ links. Since vertices of V_2^b, call it X which appear as endpoints of these links are bounded by $16k$, it suffices for us to consider those vertices which are not the endpoints of links in $\mathcal{T}$ and therefore, we consider the cactus $\mathcal{C}'$ which is obtained by refining the cycle C with the set $\mathcal{T}$. Consider a vertex $v \in V_2^b \setminus X$. By the definition of V_2^b, the vertex v has exactly 2 links $e_1 = (v, z)$ and $e_2 = (v, w)$ incident on it and other end points of both the links, z and w are in $V(C)$.

   We claim that it cannot be the case that both z and w are in $V_2^b \setminus X$. Suppose that z and w are both indeed in $V_2^b \setminus X$. Clearly, there is a cycle $C' \in \mathcal{C}'$ which contains the vertices v, z and w and furthermore, none of these vertices are cut vertices in the cactus $\mathcal{C}'$. Now, consider the set $\mathcal{L}[C']$ of links with non-trivial projections on C'. Since the greedy algorithm terminated at this point, it must be the case that no 2 links in $\mathcal{L}[C']$ *cross* (although they may be incident on the same vertex of C').

   Observe that if z and w were the 2 neighbors of v in the cactus then the path z, v, w would form a laminar strip of length 2, which contradicts our assumption that Reduction Rule 14.12 does not apply. Therefore, at least one z or w is not adjacent to v on the cycle C'. Without loss of generality we assume that z is not adjacent to v in the cycle. Let P be the subpath of C' between v and z that does not contain w. Let v' be the vertex on the path adjacent to v and let z' be the vertex on this path adjacent to z. It could be the case that $v' = z'$. Now, consider the 2-cut formed by the edges (z, z') and (v, v'). Since (b, a) is a link and no pair of links in $\mathcal{L}[C']$ cross, any link which covers this cut must have either z or v as one of its points. Since

by assumption, w cannot be equal to z' and v only has 2 links incident on it, the only possible link in $\mathcal{L}[C']$ and therefore in $\mathcal{L}$ which can cover this 2-cut is the link (b, v'). However, this contradicts our assumption that Reduction Rule 14.3 cannot be applied. Therefore, we conclude that either z or w is not in $V_2^b \setminus X$. That is, every vertex in $V_2^b \setminus X$ is incident on a link whose other endpoint lies in $V_3 \cup V_2^a \cup X$. We now bound the size of $V_2^b \setminus X$ as follows.

The number of vertices of $V_2^b \setminus X$ which can be certified by X is bounded by $|X| \leq 16k$. Those which can be certified by the vertices in V_2^a is bounded by $2|V_2^a| \leq 2k$. Therefore it remains to bound the number of vertices of $V_2^b \setminus X$ which can be certified by V_3.

By the argument in case 1, at most $24k$ links are projectively incident on the vertices of V_3. This is because, otherwise, the sum of the number of links projectively incident on each vertex in the cycle C is at least $2(\ell - 8k) + 24k$, implying that there are at least $\ell - 8k + 12k = \ell + 4k$ links projectively incident on the cycle, implying that the given instance is a Yes instance. Thus at most $24k$ vertices of $V_2^b \setminus X$ are certified by the vertices of V_3. Thus $|V_2^b \setminus X| \leq 16k + 2k + 24k = 42k$. Hence the total number of vertices in the cycle is $|V_3| + |V_2^a| + |X| + |V_2^b \setminus X| < 8k + k + 16k + 42k = 67k$.

□

We now bound the treewidth of the link-intersection graph constructed from instances on which Reduction Rules 1-5 do not apply.

Lemma 14.15. *Consider an instance $(\mathcal{C}, \mathcal{L}, k)$ on which Reduction Rules 1-5 do not apply. Then, there is an algorithm that in polynomial time either returns a tree-decomposition of $I_{\mathcal{C},\mathcal{L}}$ of width $O(k)$ or correctly concludes that the given instance is a Yes instance.*

Proof. Let $I_{\mathcal{C},\mathcal{L}} = (V, E)$ and consider the rooted block tree of the given cactus and let M be the tree whose vertices correspond to the cycles of the cactus and 2 vertices are adjacent if the block corresponding to one is the parent of the block corresponding to the other in the rooted block tree. Let V_M be the vertices of M and for every $v \in V_M$ let C_v be the corresponding cycle in the cactus. We now define the bags $\beta : V_M \to V$ as follows.

$$\beta(v) = \{u \in V | u \in C_v \text{ or } u \in \mathcal{L} \text{ and } u \text{ is projectively incident on } C_v\}$$

We claim that (M, β) is indeed a tree-decomposition of $I_{\mathcal{C},\mathcal{L}}$. Clearly, every link and every terminal appear in some bag. Furthermore, for every terminal, there is a bag which contains this terminal and all links which are projectively incident on this terminal. Now, consider an edge in $I_{\mathcal{C},\mathcal{L}}$ between 2 links. They are adjacent because they are both projectively incident on some vertex in which case there is a bag which contains both these links or they cross, which implies that they are both projectively incident on some cycle, in which case the corresponding bag contains both these vertices. Hence we conclude that every edge of $I_{\mathcal{C},\mathcal{L}}$ is contained in some bag. Finally, observe that any terminal appears only in those bags whose corresponding cycles contain this terminal and they are by definition connected. Similarly, a link only appears in those bags whose corresponding cycles have a non-trivial projection of this link, which by definition form a path in the tree M. This concludes the proof that (M, β) is indeed a tree decomposition of $I_{\mathcal{C},\mathcal{L}}$.

By Lemma 14.14, we have that every cycle in the cactus has length at most $67k$ and by Lemma 14.8, we have that if the number of links projectively incident on a cycle exceeds the length of the cycle by more than $4k$, then we have a Yes instance. Therefore, if the number of links projectively incident on any cycle exceeds $71k$, then we say Yes. Otherwise, every cycle has at most $71k$ links incident on it and therefore every bag in the tree-decomposition has size at most $138k$, thus concluding the proof of the lemma. □

We are now have the following lemma.

Lemma 14.16. *There is an algorithm for* $m - k$ Cactus Augmentation *running in time* $\mathcal{O}^*(2^{\mathcal{O}(k)})$.

Proof. We first apply Reduction Rules 1-5 on the given instance and then use the algorithm of Lemma 14.15 to either conclude that the given instance is a Yes instance or obtain a tree decomposition for the link-intersection graph, whose width is bounded linearly in the parameter. By Lemma 13.9, it suffices to compute a minimum Steiner tree in $I_{\mathcal{C},\mathcal{L}}$ with the set of degree-2 vertices of the cactus as the terminals. We then solve the problem by

using the algorithm for Steiner Tree given by [BCKN13] running in time $O^*(2^{O(tw)})$, which translates to an $O^*(2^{O(k)})$ algorithm for $m-k$ Cactus Augmentation. □

The proof of the following theorem follows easily from Lemma 14.16, Lemma 14.18 (proved in the next section) and the results of Dinits et al. [DKL76].

Theorem 14.1. *$m-k$ Augmentation by One is solvable in time $2^{\mathcal{O}(k)}|V|^{\mathcal{O}(1)}$.*

Proof. Dinits et al. [DKL76] showed that the general problem of $m-k$-augmentation by one reduces in polynomial time either to $m-k$ Cactus Augmentation or to $m-k$ Tree Augmentation without an increase in the parameter. Therefore, by Lemma 14.16 and Lemma 14.18, we have the claimed algorithm. □

14.2 Polynomial Kernels for $m-k$ Cactus Augmentation

In this section we show that $m-k$ Augmentation by One admits a polynomial kernel. We first consider the cases of tree augmentation and cactus augmentation, separately.

14.2.1 A linear kernel for $m-k$ Tree Augmentation problem

Here we give a linear kernel for $m-k$ Tree Augmentation. Note that, we obtain an FPT algorithm for this problem which runs in time $2^{\mathcal{O}(k)}n^{\mathcal{P}(1)}$, as a corollary.

Definition 14.5. *In a rooted tree, the depth of any node in a tree is the length of the shortest path from the root the node. The depth of the root is 0 and the depth of any other node is one greater than the depth of its parent. The depth of any edge $f=ab\in E(T)$ is defined as $depth(f)=\mathsf{min}(depth(a),depth(b))$. The depth of a link $e=xy\in\mathcal{L}$ is the depth of the least common ancestor of x and y.*

Definition 14.6. *A graph is called 2-degenerate if all of its induced subgraphs have a vertex of degree at most 2. An ordering π of the vertices of a graph H is called a 2-degenerate ordering if for any vertex $v=\pi(i)$, the degree of v in $H[V_i]$ is at most 2.*

Lemma 14.17. *Let (G, T, k) be the given instance of the problem, with the additional property that every cut is covered by at least two links from $\mathcal{L}$. Then there exists an augmenting set for the tree T using at most $\lfloor 2|\mathcal{L}|/3 \rfloor$ links.*

Proof. We construct an auxiliary graph H whose vertex set is the set of links. And the depth of a vertex in H is the same as the depth of the corresponding link. The edge set is defined as follows. Let σ_E be an ordering of the edges of the tree in the non-increasing order of their depths. We process the edges according to this ordering. For an edge (a, b) in the tree, let V_{ab} be the set of vertices corresponding to the links covering the edge (a, b). If there exists an edge between a pair of vertices in V_{ab}, we do nothing. Otherwise V_{ab} is an independent set at this stage. Let ψ_{ab} be an ordering of V_{ab} in increasing order of their depths. We put an edge between the first and the second vertex of the list, i.e., $(\psi_{ab}(1), \psi_{ab}(2)) \in E(H)$. We do this for all the edges of the tree, processing each according to the list σ_E and obtain the graph H.

Note that for every edge(cut) in T, there exists an edge in H. Therefore it is easy to see that a vertex cover of H corresponds to a set of links that covers all the edges(cuts) of T. Thus if there exists an independent set of size k in H, then there exists an augmenting set of size $|\mathcal{L}| - k$ for T. Our objective is to show that the graph H we constructed is 2-degenerate, which implies the presence of an independent size of size at least $\lceil |\mathcal{L}|/3 \rceil$ in H which in turn implies an augmenting set of size at most $|\mathcal{L}| - \lceil |\mathcal{L}|/3 \rceil = \lfloor 2|\mathcal{L}|/3 \rfloor$ for T.

Claim 1. *The graph H is 2-degenerate.*

Proof. Let ϕ_H be the ordering of the vertices of H in the increasing order of their depths. We claim that this ordering is a 2-degenerate ordering. Suppose not and let $H[V_i]$ (The graph induced by the first i vertices in the ordering ϕ_H) be an induced subgraph which has minimum degree greater than 2. Let e the link corresponding to the vertex $\phi_H(i)$. Let the path covered by this link $e = v$ in T be $P = \{x, v_1, v_2, \ldots, v_r, y\}$. Let z be the least common ancestor of x and y. Let some three of the links corresponding to the neighbours of $\phi_H(i)$ in $H[V_i]$ be f_1, f_2, f_3. Note that all these links have depth at most as that of e since they appeared before e in the ordering ϕ_H. Since all three links are adjacent to e

in H, they cover an edge in the path P. Combining the observations in the previous 2 sentences, we conclude that the least common ancestor of the endpoints of any of these links is either z or is an ancestor of z and all three of them cover an edge incident on z. Since there are only two edges of the path P incident on z, at least two of the links in f_1, f_2, f_3 have to cover one of these edges. Without loss of generality assume that f_1 and f_2 cover the edge (z, v_i) where v_i is in the subpath P' of P between x and z. Since there are edges (e, f_1) and (e, f_2) in H, there are two distinct edges, say (a, b) and (c, d) in the path P' that are responsible for the edges (e, f_1) and (e, f_2) in H respectively. Let the depth of (a, b) be greater than that of (c, d). Then when we were building the graph H, we would have added the edge (e, f_1) to H when processing the edge (a, b). Observe that the edge (c, d) is covered by both e and f_1. Therefore, the edge (c, d) could not have been responsible for the addition of the edge (e, f_2) in H, a contradiction. Hence we conclude that the graph H is 2-degenerate. □

□

Given Lemma 14.17, if $|\mathcal{L}| \geq 3k$, then the given instance is a Yes instance. Therefore, we may assume that $|\mathcal{L}| \leq 3k$.

Observe that the number of vertices of the tree with a link incident on it is bounded by $2|\mathcal{L}|$. This set includes all the leaves of the tree. Furthermore, the implies that the number of vertices of the tree with degree at least 3 is also bounded by $2|\mathcal{L}|$. By short-circuiting every degree-2 path in the tree with no links incident on the internal vertices (adding an edge between the endpoints of the path and removing the internal vertices of the path), we may assume that every degree-2 vertex in the tree has a link incident on it. Therefore, we have that $n \leq 4|\mathcal{L}| = 12k$.

To conclude the proof of the kernel, we need to convert any given instance into one with the property assumed by the statement of Lemma 14.17. Observe that if a single link covers a bridge in the tree, then this link must be part of every augmenting set and hence we get an equivalent instance when we contract all edges covered by any such link and removing this link, with no change in the number of links to be excluded, k. Therefore, given an

instance $(T, \mathcal{L}, k)$, we obtain an equivalent instance $(T', \mathcal{L}', k)$ by repeatedly contracting an edge which is the only edge covering a bridge in the tree T. It is easy to see that the resulting instance (G', T', k) is equivalent to the original instance and furthermore every edge in T' is covered by at least 2 links.

We also note the following corollary of the upper bound of $3k$ on the set of links.

Lemma 14.18. *There is an algorithm for $m - k$* Tree Augmentation *running in time $\mathcal{O}^*(8^k)$.*

14.2.2 A quadratic kernel for $m - k$ Cactus Augmentation

Definition 14.7. *In a cactus, cycles of length 2 are called* **trivial** *cycles and the others,* **non-trivial** *cycles. A cycle is called* **empty** *if none of the links projectively incident on it are properly incident on it. Otherwise the cycle is a* **non-empty** *cycle. A* **leaf-cycle** *in a cactus is a cycle which has exactly one cut vertex incident on it.*

We will also need the following observation which follows from Lemma 14.8.

Observation 14.19. *Consider a non-trivial cycle C in the cactus. Then, the number of trivial leaf-cycles which share a vertex with C is bounded by $\ell + 4k$ where ℓ is the length of C.*

Definition 14.8. ***Reforming an empty non-trivial leaf-cycle.*** *Consider the (arbitrarily rooted) block tree of the cactus $\mathcal{C}$ and let C be an* empty *cycle which corresponds to a non-leaf vertex whose children are all trivial leaf-cycles. Let v be the cut vertex on C separating it from its parent cycle and let $U = \{u_1, \ldots, u_r\}$ be the other cut-vertices on C. Suppose that every link projectively incident on C is also projectively incident on v.*

We then define the operation of **reforming** *this cycle as follows.*

- C is non-trivial or C is trivial and u_1 has a link properly incident on it: *First subdivide exactly once every edge except those incident on v. Then, replace every link of the form (u_i, w) where w is any vertex, with the link (v, w) and identify all the subdivision vertices with v*

- C is trivial and u_1 has no link properly incident on it: *Identify v and u_1.*

We now give a reduction rule for empty non-trivial leaf cycles.

Reduction Rule 14.20. *Suppose the (arbitrarily rooted) block tree of the cactus $\mathcal{C}$ has a cycle C which is* empty *and corresponds to a non-leaf vertex whose children are all trivial leaf-cycles. Furthermore, suppose that every link projectively incident on C is also projectively incident on the cut-vertex separating C from its parent cycle in the block tree. Then reform the cycle C.*

Lemma 14.21. *Reduction Rule 14.20 is safe.*

Proof. Let $U = \{u_1, \ldots, u_r\}$ be the set of cut vertices on C other than v, which is the cut vertex separating C from its parent in the block tree. The correctness of the rule in the second case, that is, when C is a trivial cycle and there is no link incident on u_1 is easy to see and therefore, we only concentrate on the application of the rule in the first case. Consider an augmenting set F for the original instance. Consider the corresponding link set F' in the reduced instance given by the definition of the reformation operation. We claim that F' is an augmenting set for $\mathcal{C}'$. Suppose that this is not the case. Clearly any uncovered 2-cut must be one of the new 2-circuits obtained by the reformation of the cycle C. Let the 2-circuit be $\{a, v\}$ where a is a vertex in C. Since there was a link F projectively incident on a and v in $\mathcal{C}$ and there is none in F', it must be the case that $a \in U$. However, since a is a cut-vertex in $\mathcal{C}$ there was a 2-circuit $\{a, b\}$ in $\mathcal{C}$. Since F contains a link projectively incident on b and v, the corresponding link in F' must also be projectively incident on a, a contradiction.

Conversely, consider an augmenting set F' for the reduced instance. Construct the set F of links in the original instance by simply replacing each link F' by the corresponding link in the original instance. We claim that F is an augmenting set for $\mathcal{C}$. If this were not the case, it must be the case that any uncovered cut must be from C. However, since v now forms a 2-circuit with every vertex in C and for each such 2-circuit, there is a link in F' which covers this 2-circuit, we have that for every vertex $a \in C$, there is a link in F which is projectively incident on both a and v. This implies that F covers every 2-circuit in C, a

contradiction. Hence, we conclude that the reduced instance is equivalent to the original instance and therefore, Reduction Rule 14.20 is safe. □

We define an irreducible instance as an instance on which none of the Reduction Rules 1-6 apply.

Lemma 14.22. *Let $(\mathcal{C}, \mathcal{L}, k)$ be an irreducible* No *instance of mkcactusaug. Then, the size of this instance is $\mathcal{O}(k^2)$.*

Proof. Consider the rooted block tree (where the root is not a leaf) associated with this cactus, say T. Let T' be the tree resulting from T by removing all leaves of this tree which correspond to trivial cycles. Since Reduction Rule 14.20 does not apply, it must be the case that any cycle corresponding to a leaf of T' must be either non-empty or there is a link projectively incident on C which is not projectively incident on the cut-vertex separating C from its parent cycle. Therefore, if number of leaves in T' is at least k, then it implies the presence of k links such that the corresponding sets of 2-cuts they cover are pairwise disjoint and hence we would have a Yes instance. Therefore, we conclude that the number of leaves of T' is at most $k-1$. This also gives us a bound on the number of vertices of T' of degree at least 3.

Now, consider the degree-2 paths in T' where the endpoints are also degree-2 vertices. Let this set of paths be Z. We now define a packing Z' of degree-2 paths as follows. Consider a path $P = \{v_1, \ldots, v_r\} \in Z$ with endpoints where v_1 is a descendant of v_r in the tree T'. Let i be the least index such that v_i satisfies the following property. The sub-cactus $\mathcal{C}'$ corresponding to the path $v_1, \ldots, v_i$ (including the vertices in the trivial cycles attached to the cycles corresponding to these vertices in T') has a link with both endpoints inside this sub-cactus referred to as an *internal link*. We then add the path $v_1, \ldots, v_i$ to Z' and replace the path P in Z with the path $v_{i+1}, \ldots, v_r$ and continue this process. We now have the following claim.

Claim 1. *When the process described above can no longer continue, each degree-2 path in Z and Z' has at length most $8k+3$ and the number of paths in the packing Z' is bounded by $k-1$.*

Proof. Suppose that there is a degree-2 path in Z' of length greater than $8k+3$. Then, the 2 endpoints of this path correspond to 2-cut vertices in the cactus such that every link projectively incident on the corresponding sub-cactus is also projectively incident on one of the 2 vertices. Since there is a link projectively incident on every vertex in the sub-cactus and there are at least $8k+1$ vertices other than the cut-vertices themselves in the sub-cactus, one of the 2-cut vertices must be projectively incident on at least $4k+1$ links, which by Lemma 14.7 implies that the given instance is a Yes instance, a contradiction to our assumption that the given instance is a No instance. Observe that by the same argument, no path left in Z can correspond to a sub-cactus with more than $8k+3$ vertices as otherwise it would contradict our assumption that the construction of Z' is complete.

For the last part of the statement, observe that there is a set of $|Z'|$ links one corresponding to each path in Z' such that each of them is internal to the corresponding sub-cactus. That is, no two of these links cover the same cut. Therefore, a packing of size k implies that the given instance is a Yes instance. This completes the proof of the claim.

□

The number of degree-2 paths in Z was originally bounded by twice the number of leaves of T', that is, $2k$. Furthermore, after the construction of Z', the number of paths in Z cannot increase since in each step since we only replace a path in Z with another. Therefore, the number of paths in Z is always bounded by $2k$.

We now bound the number of vertices in the sub-cactus corresponding to each degree-2 path in the packings Z and Z'. Observe that the argument in the above claim also implies that the number of vertices in each such sub-cactus excluding those lying on the end point cycles is also bounded by $8k+3$. By Lemma 14.14 and Observation 14.19, we have that the number of vertices involved in trivial cycles attached to the same cycle is bounded by $\mathcal{O}(k)$. Therefore, the number of vertices involved in the sub-cactus corresponding to each degree-2 path can exceed $8k+3$ by only $\mathcal{O}(k)$ (due to the vertices and trivial cycles attached to the 2 endpoint cycles) and hence the total number of vertices involved in the sub-cactus corresponding to any degree-2 path in Z or Z' is bounded by $\mathcal{O}(k)$. Since the number of paths in Z and Z' is $\mathcal{O}(k)$, we have that the number of vertices which occur in

the sub-cacti corresponding to the paths in the packing is $\mathcal{O}(k^2)$.

At this point, the only vertices left to be bounded are those that form a 2-circuit with a cycle which appears a leaf of T'. However, by Observation 14.19, the number of such 2-circuits corresponding to each cycle is $\mathcal{O}(k)$ and since the number of such cycles is already bounded by $\mathcal{O}(k)$, we have that these vertices are also bounded by $\mathcal{O}(k^2)$. We have thus bounded the vertex set of the cactus by $\mathcal{O}(k^2)$. Since the number of links needed to augment the cactus is also bounded linearly in the size of the cactus, we can assume that the size of the link-set in the instance does not exceed this lower bound by more than k, implying a bound of $\mathcal{O}(k^2)$ on the link-set as well. This completes the proof of the lemma.

□

14.2.3 Kernels for general $m - k$ Augmentation by One.

Given an instance of $m - k$ Augmentation by One, we apply the algorithm of [DKL76] and reduce it in polynomial time to an equivalent instance of either $m - k$ Tree Augmentation or $m - k$ Cactus Augmentation. We then apply the apply the appropriate kernelization algorithm and obtain an equivalent kernelized instance $(G, \mathcal{L}, k)$ of $m - k$ Tree Augmentation or $m - k$ Cactus Augmentation. In the former case, make λ copies of each tree edge and in the latter case, make $\lambda/2$ copies of each cactus edge and return the instance $(G', \mathcal{L}, k)$ where G' is the graph thus constructed. This is an equivalent instance of $m - k$ Augmentation by One which has a bounded number of vertices and links. This proves the following theorem.

Theorem 14.2. *$m - k$ Augmentation by One admits a kernel with $12k$ vertices and $3k$ links for odd λ and a kernel with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ links for even λ, where λ is the connectivity of the input graph G.*

Chapter 15

Minimum Equivalent Digraph is Fixed-Parameter Tractable

In this chapter we consider a classical network design problem in digraphs, namely, the Minimum Equivalent Digraph (MEG) problem. Two digraphs G and H on the same vertex set, $V(G) = V(H)$, are said to be *equivalent*, and denoted by $G \equiv H$, if for any pair of vertices u, v, there is a path from u to v in G if and only if there is a path from u to v in H. That is, two vertices in G are reachable if and only if they are reachable in H. A decision version of MEG consists of a digraph G and a positive integer ℓ, and the objective is to decide whether there is a sub-digraph H of G on at most ℓ arcs such that $G \equiv H$. This problem is easily seen to be NP-complete, by a reduction from the Hamiltonian Cycle problem [GJ79]. It has been extensively studied in the realm of approximation algorithms and exact algorithms (see the subsection on related work for details on this). However, in the realm of parameterized complexity this problem is still not well understood.

Let (G, ℓ) be an input instance to MEG on n vertices and m arcs. It is well known that MEG can be reduced to an input G' which is strongly connected (that is, there is a directed path between every pair of vertices in G'). The following proposition is due to Moyles and Thompson [MT69], see also [BJG08, Sections 2.3], reduces the problem of finding a minimum equivalent sub-digraph of an arbitrary G to a strong digraph.

Proposition 15.1. *Let G be a digraph on n vertices with strongly connected components*

$C_1, \ldots, C_r$. *Given a minimum equivalent subdigraph C'_i for each C_i, $i \in [r]$, one can obtain a minimum equivalent subdigraph G' of G containing each of C'_i in $\mathcal{O}(n^\omega)$ time.*

Proposition 15.1 allows us to reduce an instance of MEG on a general digraph to instances where the graph is strongly connected, in polynomial time. Observe that for a strong digraph G any equivalent sub-digraph is also strong. This implies that since G has n vertices, the number of arcs in H must be at least n. The way we have a lower bound on ℓ, there is also an upper bound on ℓ when the input digraph is strongly connected.

A digraph T is an *out-tree* (an *in-tree*) if T is an oriented tree with just one vertex s of in-degree zero (out-degree zero). The vertex s is the root of T. If an out-tree (in-tree) T is a spanning subdigraph of D, T is called an *out-branching* (an *in-branching*). It is known that a digraph is strong if and only if it contain an out-branching and an in-branching rooted at some vertex $v \in V(D)$ [BJG08, Proposition 12.1.1]. This implies that $\ell \leq 2n - 2$. Thus, a natural question is whether one can obtain an equivalent sub-digraph H of size $\ell \leq 2n - 2 - k$ with k being the parameter. However as we have discussed in chapter 8, this parameter is not helpful in all instances. Hence, we parameterize the problem by the maximum number of arcs which may be safely removed from the graph. In particular we study the following problem.

Minimum Equivalent Digraph	Parameter: k
Input: A digraph G and an integer k	
Question: Is there a set of arcs F in G of size at least k, such that G and $H = G \setminus F$ are equivalent ?	

The parameterized complexity of Minimum Equivalent Digraph as stated above has remained open until now. In this chapter we show that the problem is FPT by showing that it admits a polynomial kernel of size $\mathcal{O}(k^4)$.

Theorem 15.1. Minimum Equivalent Digraph *admits a kernel with $\mathcal{O}(k^4)$ vertices and arcs.*

As a corollary of Theorem 15.1, we obtain the following theorem.

Theorem 15.2. Minimum Equivalent Digraph *has an algorithm running in time*

$2^{\mathcal{O}(k \log k)} + n^{\mathcal{O}(1)}$.

By Proposition 15.1, MEG reduces to the following problem.

Minimum Strongly Connected Spanning Subgraph	Parameter: k
Input: A strongly connected digraph G and an integer k	
Question: Is there a set of arcs F in G of size at least k, such that G and $H = G \setminus F$ are equivalent ?	

There appears to be an absence of consensus in the literature on how to refer to these problems. MEG sometimes is also referred as Minimum Equivalent Digraph and Minimum Equivalent Subdigraph, while Minimum Strongly Connected Spanning Subgraph (MSCSS) is also called Minimum Spanning Strong Subdigraph (MSSS). For MSCSS we obtain a FPT algorithm with linear time dependence on the input size. In particular, we get the following.

Theorem 15.3. MSCSS *has an algorithm running in time* $2^{\mathcal{O}(k \log k)}\mathcal{O}(n + m)$.

Our Methods. An arc in a digraph is deletable, if removing it doesn't alter the reachability relations in the graph. At the heart of our algorithm is the following combinatorial result. If a digraph contains more than $\mathcal{O}(k^2)$ deletable arcs then there is a set of k arcs which can be removed from the graph without altering the reachability relations. To prove this result we consider the structure of a maximal set of arcs whose removal results in an equivalent digraph. We delete the arcs of such a set iteratively, and note how deletable arcs in the graph turn into undeletable arcs. The core of our argument is that if at any step a large number of arcs turn undeletable, then we may remove this large set of arcs from the graph without altering the reachability relations in the graph. Then using this structural result and another reduction rule we obtain a polynomial kernel for MEG. We also use the above structural result, along with a result of Italiano et. al. [ILS12], to obtain a linear time FPTalgorithm MSCSS.

Related Work. The algorithmic study of MEG can be traced to the work of Moyles and Thompson [MT69], and Hsu [Hsu75], who showed that this problem can be solved

exactly in $\mathcal{O}(n!)$ time. This was improved to an algorithm with a running time of $\mathcal{O}(2^m)$ [Mar79, MT82], where m is the number of arcs in the graph. Recently, an algorithm with running time $2^{\mathcal{O}(n)}$ was obtained by Fomin et. al [FLS14], where n is the number of the vertices in the digraph.

It was shown by Aho, Garey and Ullman [AGU72] that when the input graph is acyclic, the problem can be solved in polynomial time. The problem of minimizing the size of the graph H has been well studied in the realm of approximation algorithms. A factor 2 approximation algorithm was given by Fredricson and Jájá [FJ81]. This was improved to a factor 1.617 approximation algorithm by Khullar, Ragavachari and Young [KRY95, KRY96]. Subsequently, this was improved to a factor 1.5 approximation algorithm by Vetta [Vet01]. A different factor 1.5 approximation algorithm for this problem was presented by Berman, Dasgupta and Karpinski [BDK09]. The problem of computing a maximum sized deletion set of arcs has also been studied, and Berman, Dasgupta and Karpinski [BDK09] have shown a factor $\frac{1}{2}$ approximation algorithm for this problem. We refer to chapter 12 of the book of Bang-Jensen and Gutin [BJG08], for more combinatorial and algorithmic results on MEG.

15.1 Preliminaries

Equivalence of digraphs. Two digraphs G and H on the same vertex set, $V(G) = V(H)$, are said to be *equivalent*, if for any pair of vertices u, v, there is a path from u to v in G if and only if there is a path from u to v in H. Note that the notion of equivalence of digraphs is transitive, i.e. if G_1 is equivalent to G_2 and G_2 is equivalent to G_3 then G_1 is equivalent to G_3. We slightly generalize the notion of equivalence as follows. Let G and H be two digraphs such that $V(G) \subseteq V(H)$. And suppose we have that, for any pair of vertices u and v in $V(G)$, there is a path from u to v in G if and only if there is a path from u to v in H. Then we say that G and H are *equivalent with respect to* $V(G)$. This definition allows us to make a useful observation.

Observation 15.2. *Let G be a graph and let H be obtained from G by subdividing some of its arcs. Then G and H are equivalent with respect to $V(G)$.*

15.2 Minimum Equivalent Digraph

In this section we will show that Minimum Equivalent Digraph admits a polynomial kernel. We begin by deducing a combinatorial structure on the arcs of the digraph. Let (G, k) be an instance of Minimum Equivalent Digraph.

Definition 15.4. *Let $e = (u, v) \in E(G)$. We say that e is **deletable** (in G) if there is a path in $G - e$ from u to v. All arcs that are not deletable are called **undeletable**.*

We also make the following simple observation.

Observation 15.3. *Let G be a digraph and e be a deletable arc in G. Then G and $G - e$ are equivalent digraphs.*

Clearly, any solution to the instance (G, k) (if one exists) must be a subset of the set of deletable arcs. We now consider the set of all deletable arcs in G and prove certain properties of these arcs. Before moving ahead, we note that the set of deletable arcs can be computed in polynomial time.

Definition 15.5. *Let G be a digraph and $e = (u, v) \in E(G)$ be a deletable arc. Let P be a path from u to v which does not contain e. Then P is called an **alternate path** of e. Furthermore, let e_1 and e_2 be two deletable arcs in G such that e_2 occurs in every alternate path of e_1. Then, we say that e_1 **requires** e_2 or e_2 is required for e_1.*

The following lemma shows that the above relation among the arcs is symmetric.

Lemma 15.4. *For any two deletable arcs, e_1 requires e_2 if and only if e_2 requires e_1.*

Proof. Let $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$. Suppose that e_1 requires e_2 and e_2 does not require e_1. Let P_1 be an alternate path of e_1. Then by definition, P_1 contains e_2. Let P_A be the sub-path of P_1 from u_1 to u_2, and let P_B be the sub-path of P_1 from v_2 to v_1.

Let P_2 be an alternate path of e_2 such that,P_2 doesn't contain e_1. Now, observe that the walk $P_A + P_2 + P_B$ goes from u_1 to u_2 to v_2 to v_1. Furthermore, it does not contain e_1 or e_2. This implies the presence of an alternate path for e_1 which is disjoint from e_2, a

contradiction to the fact that e_1 requires e_2. The argument for the converse is analogous. This completes the proof of the lemma. □

The above lemma motivates the following definition.

Definition 15.6. *Let e_1 and e_2 be a pair of deletable arcs, such that e_1 is required for e_2. Then we call (e_1, e_2) a **critical pair**.*

Further note the following corollary.

Corollary 15.5. *Let (e_1, e_2) be a critical pair. Then e_1 is undeletable in $G - e_2$, and vice versa.*

Next, we consider some properties of a set of critical pairs such that they all have one arc in common.

Lemma 15.6. *Let $e = (s, t)$ be an arc and let (e, e_i) be a critical pair for $i \in \{1 \dots \ell\}$, where $e_i = (u_i, v_i)$. Let P and Q be two alternate paths for e. Then P and Q visit the arcs $e_1, \dots, e_\ell$ in the same order.*

Proof. Since P and Q are alternate paths for e and (e, e_i) form a critical pair, we have that P and Q contain all of $\{e_1, e_2, \dots e_\ell\}$. Suppose that P visits the arcs in the order $e_1, e_2, \dots, e_\ell$. Now, suppose that Q visits these arcs in a different order and let i be the first position, where the orderings of P and Q differ. That is, the path P goes from e_{i-1} to e_i while the path Q goes from e_{i-1} to e_j, where without loss of generality we assume $j > i$.

Now consider the walk $P[s, v_{i-1}] + Q[v_{i-1}, v_j] + P[v_j, t]$. Observe that this walk goes from s to t, but it doesn't contain $e = (s, t)$ or $e_i = (u_i, v_i)$. This implies the presence of an alternate path for e which is disjoint from e_i. But this is a contradiction, as (e, e_i) is a critical pair. Hence, we conclude that P and Q must visit these arcs in the same order, completing the proof of the lemma. □

We note down the following corollaries and variants of the above lemma.

Corollary 15.7. *Let $e = (s, t)$ be an arc and let (e, e_i) be a critical pair for $i \in \{1 \dots \ell\}$, where $e_i = (u_i, v_i)$. Then the following hold.*

1. *Any path from s to t (that excludes e) visits the endpoints of the arcs in the order, $s, u_1, v_1, u_2, v_2, \ldots, u_\ell, v_\ell, t$.*

2. *For $j > i$, there is no path from u_i to u_j or v_j, which is disjoint from both v_i and the arc e.*

3. *For $j > i$, there is no path from v_i to v_j or u_{j+1} which is disjoint from both u_j and the arc e.*

Proof. The first statement follows from Lemma 15.6. The proof of the second and the third statement are also very similar to the proof of the above lemma. Consider the second statement, and suppose that it is not true. Let $Q_{i,j}$ be a path from u_i to u_j which avoids both e and v_i. Suppose P is an alternate path for e. Then consider the walk $P[s, u_i] + Q_{i,j} + P[u_j, t]$, which avoids the arcs $e_i = (u_i, v_i)$ and e. This walk implies the existence of an alternate path for e which avoids e_i, a contradiction. The arguments for the case of v_j are analogous.

Now suppose that the third statement is false. Let $Q_{i,j}$ be a path from v_i to v_j which avoids both e and u_j. As before, let P be an alternate path for e. Then consider the walk $P[s, v_i] + Q_{i,j} + P[v_j, t]$, which avoids the arcs $e_j = (u_j, v_j)$ and e. This walk implies the existence of an alternate path for e which avoids e_j, a contradiction. The arguments for the case of u_{j+1} are analogous. □

Next we look at the structure of alternate paths for each of the arcs e_i. We assume that the collection of arcs $e_1, e_2, \ldots, e_\ell$ is ordered according to the sequence in which an alternate path P for e visits them.

Lemma 15.8. *Let $e = (s, t)$ be an arc and let (e, e_i) be a critical pair for $i \in \{1 \ldots \ell\}$, where $e_i = (u_i, v_i)$. Then for all u_i, there is a path from u_i to s that avoids all of $\{e_1, \ldots, e_\ell\}$.*

Proof. Let P_i be an alternate path for e_i. Since (e, e_i) is a critical pair, e occurs in every alternate path of e_i. Thus P_i contains a subpath Q_i that goes from u_i to s. By definition, Q_i avoids e_i. Suppose Q_i contained some arc e_j for $j > i$ and let e_j be the first such arc

in Q_i. But then the subpath $Q_i[u_i, u_j]$ contradicts Corollary 15.7. Therefore, no such e_j occurs in Q_i.

Next we claim that for all $j < i$, e_j does not occur in Q_i. We use induction on i. For the base case $i = 1$, the claim trivially holds as there is no arc e_j such that $j < 1$. Now by the induction hypothesis, the claim holds for all e_j where $j < i$. Now let $e_{j'}$ be the first arc in Q_i such that $j' < i$. Let $Q_{j'}$ be a path from $u_{j'}$ to s which avoids all of $e_1, \ldots, e_\ell$. Then consider the walk $Q_i[u_i, u_{j'}] + Q_{j'}$, which goes from u_i to s while avoiding all of $e_1, \ldots, e_\ell$. We can find a path contained in this walk which goes from u_i to s. This completes the proof of this lemma. □

Lemma 15.9. *There is an alternate path for e_i, which avoids every e_j for $j < i$.*

Proof. Let P_i be an alternate path for e_i. We can apply Lemma 15.8 to P_i to ensure that the subpath $P_i[u_i, s]$ avoids all of $e_1, \ldots, e_\ell$. Now if P_i contains some $e_j = (u_j, v_j)$ for $j < i$, then consider the subpath $P_i[v_j, v_i]$. Observe that this subpath avoids e as well as $e_i = (u_i, v_i)$. But this contradicts Corollary 15.7. Hence P_i avoids e_j for all $j < i$. □

Next, we have the following observation.

Observation 15.10. *Let $e, e_1, \ldots, e_\ell$ be deletable arcs in G such that in $G - e$, the arcs $e_1, \ldots, e_\ell$ are undeletable. Then $(e, e_1), \ldots, (e, e_\ell)$ are critical pairs.*

Proof. For every i, since e_i is undeletable in $G - e_i$ it must be the case that e is present in every alternate path for e_i. Therefore e_i requires e and by Lemma 15.4, (e, e_i) form a critical pair. □

The following is a corollary of Lemma 15.9 and Observation 15.10.

Corollary 15.11. *Let G be a digraph and $e, e_1, \ldots, e_\ell$ be a collection of deletable arcs in G such that, $e_1, e_2 \ldots, e_\ell$ are undeletable in $G - e$. Then $G \setminus \{e_1, \ldots, e_\ell\}$ and G are equivalent digraphs.*

Proof. By Observation 15.10 we have that $e_1, e_2, \ldots, e_\ell$ occur in every alternate path of e, and we assume that they occur in the above sequence. Let G_i be the graph $G \setminus \{e_1, e_2, \ldots, e_i\}$.

We claim that G and G_i are equivalent for all $i \in \{1, 2, \ldots, \ell\}$. We prove this by induction on i. For the base case $i = 1$, we know that e_1 is deletable in the graph G, i.e. there is an alternate path for e_1 in G. Therefore G and G_1 are equivalent. Now by induction hypothesis, we assume that the claim is true for all $j = \{1, 2, \ldots i-1\}$. Next we consider the arc e_i and show that it is deletable in the graph G_{i-1}. By Lemma 15.9, e_i has an alternate path which avoids all of $e_1, e_2, \ldots, e_{i-1}$ in the graph G. Then this alternate path is also present in the graph G_{i-1}. Thus e_i is deletable in $G \setminus \{e_1, e_2, \ldots, e_{i-1}\}$. By Observation 15.3, G_{i-1} and G_i are equivalent digraphs. And since G_{i-1} and G are equivalent, therefore G_i is equivalent to G. This completes the proof of this lemma. □

The following is our main combinatorial lemma.

Lemma 15.12. *Let (G, k) be an instance of* Minimum Equivalent Digraph. *If G has more than $(k-1)^2$ deletable arcs, then this instance has a solution of size k which can be computed in polynomial time.*

Proof. Let $F = \{f_1, f_2, \ldots, f_p\}$ be an arbitrary maximal set of arcs such that $G - F$ is equivalent to G. We can construct F greedily. If $|F| = p \geq k$, then we already have the required solution. Therefore, we assume that $p \leq k - 1$.

Now, consider the graphs $G_i = G \setminus \{f_1, \ldots f_i\}$. Therefore $G_{i+1} = G_i - f_{i+1}$ and $G_p = G \setminus F$. Observe that each of the graph G_i is equivalent to G, by the definition of F. Let C_i be the set of deletable arcs in G_i which are undeletable in G_{i+1}. Then by Corollary 15.11, we have that G_i and $G_i \setminus C_i$ are equivalent. Since G_i is a subgraph of G, $G \setminus C_i$ is also equivalent to G. Therefore, if $|C_i| \geq k$, then we have the required solution. Otherwise we have $|C_i| < k$.

Now consider any deletable arc of G. It is either picked in F, or there is some i such that it is deletable in G_i but undeletable in G_{i+1}. In other words, $F \cup C_1 \ldots C_p$ cover all the deletable arcs of G. Since $p \leq k - 1$ and all of $F, C_1, \ldots, C_p$ have at most $k - 1$ arcs in them, the total number of deletable arcs in G is bounded by $(k-1)^2$. □

The above lemma is restated as the following reduction rule.

Reduction Rule 15.13. *If G has more than $(k-1)^2$ deletable arcs, then return a trivial* Yes *instance.*

As noted before, the above Reduction Rule immediately implies an FPT algorithm for Minimum Equivalent Digraph that runs in time $2^{\mathcal{O}(k \log k)} + n^{\mathcal{O}(1)}$. We now move ahead and complete the proof of Theorem 15.1 by giving a polynomial kernel for this problem. Going forward, we assume that Reduction Rule 15.13 does not apply on the instance of Minimum Equivalent Digraph we are dealing with. We then have the following observation.

Observation 15.14. *The number of vertices with a deletable arc incident on it is bounded by $2(k-1)^2$.*

Observe that bounding the number of deletable arcs alone does not imply a kernel for this problem, and our goal now is to bound the total number of arcs and vertices in the graph. We need a few additional preprocessing rules to obtain an equivalent instance whose size is polynomially bounded in the parameter. To this end, first we consider a colored version of the problem, where arcs are colored red and green and only the green arcs are allowed in any solution. We show that we can reduce the number of vertices in such a colored instance. Finally, we shall reduce the colored instance to a uncolored instance.

We begin by coloring all the undeletable arcs of G red, and the remaining arcs green. We have the following lemma, which is easy to see.

Lemma 15.15. *Let G' be the graph obtained from G by coloring its arcs as above. Then the colored instance and the uncolored instances are equivalent.*

We now apply the following reduction rule. We further have the following observation from Reduction Rule 15.13.

Observation 15.16. *The number of green arcs in a colored instance (G, k) is bounded by $(k-1)^2$.*

Reduction Rule 15.17. *Let v be a vertex in G such that only red arcs are incident on it, and let $N^-(v)$ and $N^+(v)$ be the in-neighbourhood and out-neighbourhood of v. Then*

remove v from G, and for each $x \in N^-(v)$ and $y \in N^+(v)$ add an arc from x to y, if it is not present, and color it red.

Lemma 15.18. *Reduction rule 15.17 is safe.*

Proof. Let G' be the graph obtained from G, by the above process. Let F be any solution of G (which contains only green arcs). So the arcs in F are also present in G', and are colored green. Consider a pair of vertices, x and y, in G' such that y is reachable from x. Then by construction of G', y is reachable from x in G as well. Consider any path P in $G \setminus F$, from x to y. If P doesn't contain v, then this path is also present in $G' \setminus F$. Otherwise let u and w be the in and out-neighbour of v respectively in P. Since there is a red arc from u to w in G', by modifying P, we can obtain a path P' in $G' \setminus F$ which goes from x to y. Therefore, $G' \setminus F$ and G are equivalent.

In the reverse direction, let F' be any solution in G'. Again observe that all the arcs in F' are also present in G, and are colored green. We will show that $G \setminus F'$ is equivalent to G. Now, consider a pair of vertices x and y in G, such that y is reachable from x. If x is v then set x to some arbitrary out-neighbour of v, and similarly if y is v, then set y to some arbitrary in-neighbour of v. Now, by construction of G', y is reachable from x in G'. Since F' is a solution in G', therefore there is a path P' path from x to y in $G' \setminus F'$. If P' doesn't contain any of the newly introduced arcs in G', then it is also present in $G \setminus F'$. Otherwise, by replacing each new arc (u, w) in P' with the length 2 path (u, v, w), we obtain a walk in $G \setminus F$ from x to y. Therefore $G \setminus F$ and G are equivalent. □

The next two lemmas follow easily from the above reduction rule.

Lemma 15.19. *Each application of Reduction Rule 15.17 reduces the number of vertices in the instance by* 1.

Lemma 15.20. *Let (G, k) be an instance on which Reduction Rule 15.17 cannot be applied. Then the number of vertices of (G, k) is bounded by $2(k-1)^2$.*

Proof. If the reduction rule doesn't apply, then every vertex has a green arc incident on it. By Observation 15.16, the number of green arcs is bounded by $(k-1)^2$. Therefore the

number of vertices of (G, k) is bounded by $2(k-1)^2$. □

We have the following corollary.

Corollary 15.21. *The number of red arcs in a reduced instance is bounded by* $4k^4$.

Thus, we have a colored instance with a bounded number of vertices and arcs. The next lemma shows that we can "uncolor" the instance, at a small cost.

Lemma 15.22. *Let* G' *be obtained from the colored graph* G *by replacing each red arc with a new path of length* 2*, and then removing colors from all the arcs. Then the colored instance* (G, k) *and the uncolored instance* (G', k) *are equivalent. Further* G' *has* $\mathcal{O}(k^4)$ *vertices and* (k^4) *arcs.*

Proof. Observe that, G' is obtained by subdividing some of the arcs of G. And therefore G' and G are equivalent with respect to $V(G)$.

Suppose F is a solution in the colored graph G, and therefore F is a subset of the set of green arcs in G, which are also present in G'. Now observe that $G' \setminus F$ is obtained from $G \setminus F$ by subdividing all the red arcs. Therefore by Observation 15.2, $G' \setminus F$ and $G \setminus F$ are equivalent with respect to $V(G)$. Now consider any pair of vertices in u and v in G'. Let x be the unique out neighbour of u if u is a newly added vertex, otherwise $x = u$. And similarly, let y be the unique in neighbour of v if v is a newly added vertex, otherwise $y = v$. Then observe that v is reachable from u in G' if and only if y is reachable from x in G'. Since the arcs incident on the newly added vertices of G' are disjoint from F, therefore G' and $G' \setminus F$ are equivalent. Thus, F is a solution in G'.

In the reverse direction, let F' be a solution in G'. Observe that every arc in F' was a green arc in G, as the red-arcs have been replaced by length 2 paths, and these new arcs are undeletable. We will show that F' is also a solution in G. Let u and v be two vertices in $V(G)$, such that v is reachable from u. Since G and G' are equivalent with respect to $V(G)$, and G' and $G' \setminus F'$ are also equivalent, therefore v is reachable from u in $G \setminus F'$. So consider a path P' in $G' \setminus F'$ from u to v. By replacing each of the newly added length 2 paths, which are present in P', with the corresponding red-arcs in G, we obtain a path

P in $G \setminus F'$ which goes from u to v. Therefore, $G \setminus F'$ and G are equivalent. Thus F' is a solution in G.

Next, we bound the size of G'. Recall that G has at most $\mathcal{O}(k^2)$ vertices, and there may be at most $\mathcal{O}(k^4)$ red arcs and $\mathcal{O}(k^2)$ green arcs in G. Therefore, the number of newly added vertices in G' is at most $\mathcal{O}(k^4)$, which also bounds the total number of vertices in G'. Further, for each red arc we add exactly 2 arcs to G' and for each green arc we have exactly one arc in G'. Therefore the total number of arcs in G' is also bounded by $\mathcal{O}(k^4)$. □

Combining the above results, have shown the following theorem.

Theorem 15.1. MINIMUM EQUIVALENT DIGRAPH *admits a kernel with* $\mathcal{O}(k^4)$ *vertices and arcs.*

15.2.1 A Linear time FPT algorithm for MSCSS

In this subsection, we prove Theorem 15.3. The main ingredient of our algorithm is a result by Italiano et al. [ILS12] which computes the set of all undeletable arcs in a strongly connected digraph in $\mathcal{O}(m+n)$ time. As noted by Italiano et. al., for acyclic digraphs, computing the set of undeletable arcs is essentially equivalent to boolean matrix multiplication[AGU72, Mun71, Fur70, FM71], and the best known algorithm for this problem runs in time $\mathcal{O}(n^\omega)$. Italiano et. al. use the term "strong bridge" to denote those arcs in a digraph whose removal increases the number of strongly connected components. Observe that the set of all strong bridges is precisely the set of all undeletable arcs in a strongly connected digraph.

Theorem 15.7 ([ILS12], Theorem 4.4)**.** *The set of undeletable arcs of a strongly connected directed graph* G *can be computed in* $\mathcal{O}(n+m)$ *time.*

We have the following lemma for finding a solution in an instance with a large number of deletable arcs.

Lemma 15.23. *Let* (G,k) *be an instance of* MSCSS *and let* X *be the collection of all deletable arcs in* G*, such that* $|X| > (k-1)^2$*. Let* Y *be a subset of* X*, of size* $(k-1)^2+1$*. Then there is a subset* Z *of* Y *of size at least* k*, such that* $G \setminus Z$ *is strongly connected.*

Proof. Consider the graph G' obtained from G by subdividing all the arcs in $E(G) \setminus Y$. By Observation 15.2, G' and G are equivalent with respect to $V(G)$. Then the instance (G', k) has Y as its set of deletable arcs. By Lemma 15.12, there is some subset of Z of size at least k which is a solution to the instance (G', k). It is easy to see that this set Z forms a solution for the original instance (G, k). □

Now given the set of deletable arcs in G, we have the following theorem.

Theorem 15.3. MSCSS *has an algorithm running in time* $2^{\mathcal{O}(k \log k)}\mathcal{O}(n + m)$.

Proof. Let (G, k) be an instance of MSCSS. Recall that G is strongly connected. Let X be the set of detable arcs in G. Using Theorem 15.7, we can compute X in $\mathcal{O}(n + m)$ time. If $|X| > (k - 1)^2$, then let Y be an arbitrary subset of X of size $(k - 1)^2 + 1$. Then by Lemma 15.23, there is a subset Z of Y which is the required solution. Otherwise we have that $|X| \leq (k - 1)^2$ and let $Y = X$. In both cases, we may find a solution by iterating over each subset Z of Y of size k, and testing if $G \setminus Z$ is strongly connected. To test if $G \setminus Z$ is strongly connected we may use the algorithm of Tarjan [Tar72], which runs in time $\mathcal{O}(m + n)$. Therefore we can find a solution, if it exists, in $2^{\mathcal{O}(k \log k)}(n + m)$ time. This completes the proof of this theorem. □

Conclusion

Chapter 16

Conclusion and Future Directions.

In this thesis, we gave improved and new FPT algorithms, kernels and exact algorithms for a number of problems related to network design, including

- the first single exponential FPT algorithm for Euler Edge Deletion and Undirected Connected Odd Edge Deletion,
- the first single exponential exact algorithm for Survivable Network Design with uniform requirements,
- the first single exponential FPT algorithm for Augmentation by One,
- new FPT algorithm and kernels for $m - k$ augmentation by One,
- and the first FPT and kernelization algorithm for Minimum Equivalent Digraph.

Furthermore, we gave new deterministic algorithms for two matroid theory problems, and their applications in network design problems.

We conclude with the following questions.

- *Editing to a graph class while preserving high connectivity.* Many network design problems (as well as other problems) have efficient algorithms and heuristics on certain graph classes. Can we extend those algorithms to other graphs which are "close" to this graph class ?

This leads to the following problem, which lies at the intersection of graph modification and network design problems.

> Let λ be a fixed constant and Π be a graph class. Given a graph or digraph G which is λ connected, edit (or delete) at most k edges (or vertices) to obtain a graph H, such that H is λ edge-connected (or vertex connected) and in Π.

For example, when $\lambda = 2$ and Π is the class of undirected eulerian graphs, we showed that this problem in FPT. We may ask this question for many other classes of graphs as well. We may also ask the augmentation version of the problem.

> Let λ be a fixed constant and Π be a graph class. Given a graph or digraph G and a set links $L \subseteq V \times V$, find $F \subseteq L$ containing at most k links such that $H = G \cup F$ is λ edge-connected (or vertex connected) and in Π.

- FPT *algorithms and kernels for other network design problems.* While we saw kernelization, FPT and exact algorithms for some of the problems mentioned in chapter 8, many more remain unresolved. Here we mention, two which we are very interested in.
  - Is Strong Connectivity Augmentation fixed parameter tractable ?
  - Is λ-connected Steiner Subgraph fixed parameter tractable, when parameterized by the size of the maximum edge deletion set ?

  The last two questions are unresolved for both graphs and digraphs and for all values of λ.

- *Deterministic Polynomial time representation of Gammoids and Transversal Matroids.* These matroids are central to several recent kernelization results which are randomized. Hence, a deterministic algorithm for computing their representation would derandomize all these kernels. This might be a difficult problem though, as this has connections to several other complexity theory questions.(See [Vad12] for more details.) However, many of the above algorithms can still be derandomized if one can deterministically compute representative sets over these matroids. Hence

arises the question of efficient computation of representative sets over these matroids, *deterministically*.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009. 7, 8, 20, 22

[ACN05] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *STOC*, pages 684–693, 2005. 34

[AGU72] Alfred V. Aho, Michael R Garey, and Jeffrey D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972. 208, 217

[AJ86] Leonard M. Adleman and Hendrik W. Lenstra Jr. Finding irreducible polynomials over finite fields. In Juris Hartmanis, editor, *STOC*, pages 350–355. ACM, 1986. 93

[AK10] F. N. Abu-Khzam. A kernelization algorithm for d-hitting set. *Journal of Computer and System Sciences*, 76(7):524 – 531, 2010. 34

[Alo06] N. Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, 20(1):137–142, 2006. 34

[ALS09] N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *ICALP*, volume 5555 of *LNCS*, pages 49–58, 2009. 34

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995. 57

[BCKN13] Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems

parameterized by treewidth. In *ICALP (1)*, pages 196–207, 2013. 99, 104, 180, 196

[BD10] Alin Bostan and Philippe Dumas. Wronskians and linear independence. *The American Mathematical Monthly*, 117(8):722–727, 2010. 84

[BDK09] Piotr Berman, Bhaskar Dasgupta, and Marek Karpinski. Approximating transitive reductions for directed networks. In *Proceedings of the 11th International Symposium on Algorithms and Data Structures*, pages 74–85. Springer-Verlag, 2009. 10, 67, 69, 208

[Bel62] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM (JACM)*, 9(1):61–63, 1962. 148

[BFG+09] Stéphane Bessy, Fedor V. Fomin, Serge Gaspers, Christophe Paul, Anthony Perez, Saket Saurabh, and Stéphan Thomassé. Kernels for feedback arc set in tournaments. In *FSTTCS*, pages 37–47, 2009. 34, 35

[BHKK10] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *arXiv preprint arXiv:1007.1161*, 2010. 57

[BJG08] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008. 27, 35, 69, 73, 142, 148, 205, 206, 208

[Bjo14] Andreas Bjorklund. Determinant sums for undirected hamiltonicity. *SIAM Journal on Computing*, 43(1):280–299, 2014. 148

[BJT92] Jørgen Bang-Jensen and Carsten Thomassen. A polynomial algorithm for the 2-path problem for semicomplete digraphs. *SIAM Journal on Discrete Mathematics*, 5(3):366–376, 1992. 34

[BJY08] Jørgen Bang-Jensen and Anders Yeo. The minimum spanning strong subdigraph problem is fixed parameter tractable. *Discrete Applied Mathematics*, 156(15):2924–2929, 2008. 73

[Bol65] B. Bollobás. On generalized graphs. *Acta Math. Acad. Sci. Hungar*, 16:447–452, 1965. 47

[Bor81] J. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1781. 33

[CCM92] BS Carlson, CYR Chan, and DS Meliksetian. An efficient algorithm for the identification of dual eulerian graphs and its application to cell layout. In *Circuits and Systems, 1992. ISCAS'92. Proceedings., 1992 IEEE International Symposium on*, volume 5, pages 2248–2251. IEEE, 1992. 9

[CDZ02] Mao-Cheng Cai, Xiaotie Deng, and Wenan Zang. A min-max theorem on feedback vertex sets. *Mathematics of Operations Research*, 27(2):361–371, 2002. 34

[CFG+16] Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socala. Tight bounds for graph homomorphism and subgraph isomorphism. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1643–1649, 2016. 148

[CFJ+] Marek Cygan, Fedor Fomin, Bart M.P. Jansen, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Open Problems for FPT School 2014, Będlewo, Poland. `http://fptschool.mimuw.edu.pl/opl.pdf` 128

[CFK+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Science & Business Media, to appear in 2015. 7, 20, 23

[CKN13] Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 301–310. ACM, 2013. 148

[CMP+14] Marek Cygan, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Ildikó Schlotter. Parameterized complexity of eulerian deletion problems. *Algorithmica*, 68(1):41–61, 2014. 9, 127, 128, 129, 131

[Con85] M. Condorcet. Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix, 1785. 33

[CSS97] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. In *NIPS*, pages 451–457, 1997. 33

[CT00] Joseph Cheriyan and Ramakrishna Thurimella. Approximating minimum-size k-connected spanning subgraphs via matching. *SIAM Journal on Computing*, 30(2):528–560, 2000. 71

[CTY07] P. Charbit, S. Thomassé, and A. Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Combinatorics, Probability and Computing*, 16(1):1–4, 2007. 34

[CY11] Leizhen Cai and Boting Yang. Parameterized complexity of even/odd subgraph problems. *J. Discrete Algorithms*, 9(3):231–240, 2011. 127, 128

[DGH+10] Michael Dom, Jiong Guo, Falk Hüffner, Rolf Niedermeier, and Anke Truß. Fixed-parameter tractability results for feedback set problems in tournaments. *Journal of Discrete Algorithms*, 8(1):76–86, 2010. 34, 37

[DGvHP14] Konrad K Dabrowski, Petr A Golovach, Pim vant Hof, and Daniël Paulusma. Editing to eulerian graphs. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science(FSTTCS)*, 2014. 127

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. 27, 31

[DKL76] E. Dinits, A. Karzanov, and M. Lomonosov. On the structure of a family of minimal weighted cuts in graphs. *In: Fridman, A. (ed.) Studies in Discrete Mathematics, Nauka, Moscow*, pages 290–306, 1976. 168, 180, 196, 203

[DKNS01] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001. 33

[DKSS09] Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the method of multiplicities, with applications to kakeya sets and mergers. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 181–190. IEEE, 2009. 81

[DW71] S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971. 168, 178

[EJ73] Jack Edmonds and Ellis L. Johnson. Matching, euler tours and the Chinese postman. *Mathematical Programming*, 5(1):88–124, 1973. 129

[ET76] K. Eswaran and R. Tarjan. Augmentation problems. *SIAM Journal on Computing*, 5(4):653–665, 1976. 167

[FC70] H Frank and W Chou. Connectivity considerations in the design of survivable networks. *Circuit Theory, IEEE Transactions on*, 17(4):486–490, 1970. 68

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006. 23

[FG14] Fedor V. Fomin and Petr A. Golovach. Long circuits and large euler subgraphs. *SIAM J. Discrete Math.*, 28(2):878–892, 2014. 127, 128

[FGPS] Fedor V. Fomin, Petr Golovach, Fahad Panolan, and Saket Saurabh. Editing to connected f-degree graph. *To appear in STACS 2016*. 120

[FJ81] Greg N. Frederickson and Joseph JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981. 208

[FK11] Fedor V Fomin and Dieter Kratsch. *Exact exponential algorithms*. Springer, 2011. 7, 20, 22

[FLPS14] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative sets of product families. In *Algorithms - ESA 2014 - 22th Annual*

European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings, volume 8737, pages 443–454, 2014. 11, 47, 57

[FLS14] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 142–151, 2014. 10, 11, 47, 53, 54, 55, 57, 58, 77, 102, 105, 106, 107, 112, 130, 148, 208

[FM71] Michael J Fischer and Albert R Meyer. Boolean matrix multiplication and transitive closure. In *Switching and Automata Theory, 1971., 12th Annual Symposium on*, pages 129–131. IEEE, 1971. 217

[Fra82] P. Frankl. An extremal problem for two families of sets. *European J. Combin.*, 3(2):125–127, 1982. 47

[Fra92a] A. Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM Journal on Discrete Mathematics*, 5(1):25–53, 1992. 8, 9, 69

[Fra92b] András Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM Journal on Discrete Mathematics*, 5(1):25–53, 1992. 167

[Fra93] András Frank. A survey on T-joins, T-cuts, and conservative weightings. In *Combinatorics, Paul Erdös is eighty*, volume 2, pages 213–252. János Bolyai Mathematical Society, 1993. 132

[FS12] Michael A. Forbes and Amir Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 163–172, 2012. 85

[Fur70] ME Furman. Application of a method of fast multiplication of matrices to problem of finding graph transitive closure. *Doklady Akademii Nauk SSSR*, 194(3):524, 1970. 217

[Gab95] Harold N Gabow. A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences*, 50(2):259–273, 1995. 157

[GGRW06] Jim Geelen, Bert Gerards, Neil Robertson, and Geoff Whittle. Obstructions to branch-decomposition of matroids. *Journal of Combinatorial Theory, Series B*, 96(4):560–570, 2006. 64

[GGW07] Jim Geelen, Bert Gerards, and Geoff Whittle. Excluding a planar graph from gf (q)-representable matroids. *Journal of Combinatorial Theory, Series B*, 97(6):971–998, 2007. 63

[GHM07] Jiong Guo, Falk Hüffner, and Hannes Moser. Feedback arc set in bipartite tournaments is np-complete. *Information Processing Letters*, 102(2-3):62–65, 2007. 34

[GJ79] Michael R Garey and David S Johnson. *Computers and intractability: a guide to NP-completeness*. WH Freeman New York, 1979. 205

[GJ02] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002. 7, 8, 22

[GK11] Anupam Gupta and Jochen Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3–20, 2011. 8, 67, 74

[GK13] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. In *FOCS*, pages 608–617, 2013. 84, 85

[GMP13] Prachi Goyal, Neeldhara Misra, and Fahad Panolan. Faster deterministic algorithms for r-dimensional matching using representative sets. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, December 12-14, 2013, Guwahati, India*, volume 24, pages 237–248, 2013. 47

[GMP+15] Prachi Goyal, Pranabendu Misra, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. Finding even subgraphs even faster. In *35th IARCS Annual*

Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India, pages 434–447, 2015. 120

[Gol03] David Goldschmidt. *Algebraic functions and projective curves*, volume 215. Springer, 2003. 81

[GU10] Jiong Guo and Johannes Uhlmann. Kernelization and complexity results for connectivity augmentation problems. *Networks*, 56(2):131–142, 2010. 70

[Guo09] Jiong Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8-10):718–726, 2009. 35

[Gup08] Sushmita Gupta. Feedback arc set problem in bipartite tournaments. *Information Processing Letters*, 105(4):150–154, 2008. 34

[Gus88] Dan Gusfield. A graph theoretic approach to statistical data security. *SIAM Journal on Computing*, 17(3):552–571, 1988. 68

[GV87] Arnaldo Garcia and Jose Felipe Voloch. Wronskians and linear independence in fields of prime characteristic. *Manuscripta Mathematica*, 59(4):457–469, 1987. 84

[GY12] Gregory Gutin and Anders Yeo. Constraint satisfaction problems parameterized above or below tight bounds: A survey. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond*, volume 7370 of *Lecture Notes in Computer Science*, pages 257–286. Springer, 2012. 73

[HK62] Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962. 148

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2:225–231, 1973. 119

[HP10] Michel Habib and Christophe Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010. 37

[Hsu75] Harry T Hsu. An algorithm for finding a minimal equivalent graph of a digraph. *Journal of the ACM (JACM)*, 22(1):11–16, 1975. 207

[ILS12] Giuseppe F Italiano, Luigi Laura, and Federico Santaroni. Finding strong bridges and strong articulation points in linear time. *Theoretical Computer Science*, 447:74–84, 2012. 207, 217

[IP73] A.W Ingleton and M.J Piff. Gammoids and transversal matroids. *Journal of Combinatorial Theory, Series B*, 15(1):51 – 68, 1973. 62

[JG86] SP Jain and Krishna Gopal. On network augmentation. *Reliability, IEEE Transactions on*, 35(5):541–543, 1986. 68

[Juk11] Stasys Jukna. *Extremal combinatorics*. Springer Verlag Berlin Heidelberg, 2011. 48

[Kao96] Ming-Yang Kao. Data security equals graph connectivity. *SIAM Journal on Discrete Mathematics*, 9(1):87–100, 1996. 68

[Kem59] J. Kemeny. Mathematics without numbers. *Daedalus*, 88:571–591, 1959. 33

[Khu97] Samir Khuller. Approximation algorithms for finding highly connected subgraphs. *Vertex*, 2:2, 1997. 8, 69

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. 113

[KMS07] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *STOC*, pages 95–103, 2007. 34

[KN10] Guy Kortsarz and Zeev Nutov. Approximating minimum cost connectivity problems. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2010. 8, 9, 10, 67, 69

[KRY95] Samir Khuller, Balaji Raghavachari, and Neal Young. Approximating the minimum equivalent digraph. *SIAM Journal on Computing*, 24(4):859–872, 1995. 208

[KRY96] Samir Khuller, Balaji Raghavachari, and Neal Young. On strongly connected digraphs with bounded cycle length. *Discrete Applied Mathematics*, 69(3):281–289, 1996. 208

[KS62] J. Kemeny and J. Snell. *Mathematical models in the social sciences*. Blaisdell, 1962. 33

[KS10] M. Karpinski and W. Schudy. Faster algorithms for feedback arc set tournament, kemeny rank aggregation and betweenness tournament. *CoRR*, abs/1006.4396, 2010. 34

[KV94] Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *Journal of the ACM (JACM)*, 41(2):214–235, 1994. 69, 71

[KW09] Ioannis Koutis and Ryan Williams. Limits and applications of group algebras for parameterized problems. In *Automata, languages and programming*, pages 653–664. Springer, 2009. 57

[KW12] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS 2012)*, pages 450–459. IEEE, 2012. 11, 47, 62, 112, 130

[KW14] Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms*, 10(4):20:1–20:15, 2014. 47, 112

[LMPS15] Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 922–934, 2015. 120

[Lov77] L. Lovász. Flats in matroids and geometric graphs. In *In Combinatorial surveys (Proc. Sixth British Combinatorial Conf., Royal Holloway Coll., Egham)*, pages 45–86. Academic Press, London, 1977. 47

[Mar79] S Martello. An algorithm for finding a minimal equivalent graph of a strongly connected digraph. *Computing*, 21(3):183–194, 1979. 208

[Mar09] Dániel Marx. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.*, 410(44):4471–4479, 2009. 11, 47, 50, 61, 62, 77, 78, 82, 109, 112, 113, 130

[Mon85] B. Monien. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*, volume 109 of *North-Holland Math. Stud.*, pages 239–254. North-Holland, Amsterdam, 1985. 130

[MR10] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010. 7, 24

[MRS09] Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing above or below guaranteed values. *Journal of Computer and System Sciences*, 75(2):137–153, 2009. 73

[MS03] Thom Mulders and Arne Storjohann. On lattice reduction for polynomial matrices. *J. Symb. Comput.*, 35(4):377–401, 2003. 105, 107

[MT69] Dennis M Moyles and Gerald L Thompson. An algorithm for finding a minimum equivalent graph of a digraph. *Journal of the ACM (JACM)*, 16(3):455–460, 1969. 148, 205, 207

[MT82] Silvano Martello and Paolo Toth. Finding a minimum equivalent graph of a digraph. *Networks*, 12(2):89–100, 1982. 208

[Mui82] Thomas Muir. *A Treatise on the Theory of Determinants*. Dover Publications, 1882. 84

[Mun71] Ian Munro. Efficient determination of the transitive closure of a directed graph. *Information Processing Letters*, 1(2):56–58, 1971. 217

[Mur00] Kazuo Murota. *Matrices and matroids for systems analysis*, volume 20. Springer, 2000. 82, 101

[MV15] Dániel Marx and László A Végh. Fixed-parameter algorithms for minimum-cost edge-connectivity augmentation. *ACM Transactions on Algorithms (TALG)*, 11(4):27, 2015. 10, 70, 168

[Nag03] Hiroshi Nagamochi. An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree. *Discrete Applied Mathematics*, 126(1):83–113, 2003. 70

[NMN01] Jaroslav Nešetřil, Eva Milková, and Helena Nešetřilová. Otakar borůvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1):3–36, 2001. 68

[Oxl06] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, 2006. 11, 47, 49, 51, 52, 61, 62, 111

[PTW01] Pavel A Pevzner, Haixu Tang, and Michael S Waterman. An eulerian path approach to dna fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001. 9

[RS06] V. Raman and S. Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoretical Computer Science*, 351(3):446–458, 2006. 34

[Sch03] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003. 149, 150, 157

[Sho88] Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. In *FOCS*, pages 283–290. IEEE Computer Society, 1988. 92, 93

[Shp90] Igor Evgen'evich Shparlinski. On primitive elements in finite fields and on elliptic curves. *Matematicheskii Sbornik*, 181(9):1196–1206, 1990. 93

[SHSL02] Arunabha Sen, Bin Hao, Bao Hong Shen, and Guohui Lin. Survivable routing in wdm networks-logical ring in arbitrary physical topology. In *Communica-*

tions, 2002. ICC 2002. IEEE International Conference on, volume 5, pages 2771–2775. IEEE, 2002. 9

[SKH10] Bhavesh Sanghvi, Neeraj Koul, and Vasant Honavar. Identifying and eliminating inconsistencies in mappings across hierarchical ontologies. In *OTM Conferences (2)*, volume 6427 of *LNCS*, pages 999–1008, 2010. 33

[Spe89] E. Speckenmeyer. On feedback problems in digraphs. In *WG*, volume 411 of *LNCS*, pages 218–231, 1989. 34

[SZ14] Hadas Shachnai and Meirav Zehavi. Representative families: A unified tradeoff-based approach. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737, pages 786–797, 2014. 47

[Tar72] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972. 218

[TCHP08] Marc Tedder, Derek G. Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *ICALP*, LNCS, pages 634–645, 2008. 36

[Tut65] William T Tutte. Mengers theorem for matroids. *J. Res. Nat. Bur. Standards Sect. B*, 69:49–53, 1965. 63

[Tuz94] Zs. Tuza. Applications of the set-pair method in extremal hypergraph theory. In *Extremal problems for finite sets (Visegrád, 1991)*, volume 3 of *Bolyai Soc. Math. Stud.*, pages 479–514. János Bolyai Math. Soc., Budapest, 1994. 48

[Tuz96] Zs. Tuza. Applications of the set-pair method in extremal problems. II. In *Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993)*, volume 2 of *Bolyai Soc. Math. Stud.*, pages 459–490. János Bolyai Math. Soc., Budapest, 1996. 48

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. 222

[Veg11] L. Vegh. Augmenting undirected node-connectivity by one. *SIAM Journal on Discrete Mathematics*, 25(2):695–718, 2011. 167

[Vet01] Adrian Vetta. Approximating the minimum strongly connected subgraph via a matching lower bound. In *SODA*, volume 1, pages 417–426, 2001. 10, 208

[Vic90] Shoup Victor. Searching for primitive roots in finite fields. In Harriet Ortiz, editor, *STOC*, pages 546–554. ACM, 1990. 93

[vZ11] Anke van Zuylen. Linear programming based approximation algorithms for feedback set problems in bipartite tournaments. *Theoretical Computer Science*, 412(23):2556–2561, 2011. 34

[vZHJW07] A. van Zuylen, R. Hegde, K. Jain, and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. In *SODA*, pages 405–414, 2007. 34

[Whi35] Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, 1935. 47

[Wil09] Ryan Williams. Finding paths of length k in o(k2) time. *Information Processing Letters*, 109(6):315–318, 2009. 57

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC 2012)*, pages 887–898. ACM, 2012. 27

[WN87] Toshimasa Watanabe and Akira Nakamura. Edge-connectivity augmentation problems. *Journal of Computer and System Sciences*, 35(1):96 – 144, 1987. 167

[WNN89] Toshimasa Watanabe, Takanori Narita, and Akira Nakamura. 3-edge-connectivity augmentation problems. In *Circuits and Systems, 1989., IEEE International Symposium on*, pages 335–338. IEEE, 1989. 69

[XG12] Mingyu Xiao and Jiong Guo. A quadratic vertex kernel for feedback arc set in bipartite tournaments. In *MFCS*, volume 7464 of *LNCS*, pages 825–835. Springer, 2012. 45

[ZA12] Daniel Zelazo and Frank Allgöwer. Eulerian consensus networks. In *CDC 2012. 51st IEEE Conference on Decision and Control*, pages 4715–4720, 2012. 9

[Zeh15] Meirav Zehavi. Mixing color coding-related techniques. In *Algorithms-ESA 2015*, pages 1037–1049. Springer, 2015. 58