ABSTRACT

Title of dissertation: DIRECTED GRAPHS: FIXED-PARAMETER TRACTABILITY & BEYOND

Rajesh Chitnis, Doctor of Philosophy, 2014

Dissertation directed by: Professor MohammadTaghi Hajiaghayi
Department of Computer Science

Most interesting optimization problems on graphs are NP-hard, implying that (unless P = NP) there is no polynomial time algorithm that solves all the instances of an NP-hard problem exactly. However, classical complexity measures the running time as a function of only the overall input size. The paradigm of *parameterized complexity* was introduced by Downey and Fellows to allow for a more refined multivariate analysis of the running time. In parameterized complexity, each problem comes along with a secondary measure k which is called the parameter. The goal of parameterized complexity is to design efficient algorithms for NP-hard problems when the parameter k is small, even if the input size is large. Formally, we say that a parameterized problem is fixed-parameter tractable (FPT) if instances of size n and parameter k can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function which does not depend on n. A parameterized problem belongs to the class XP if instances of size n and parameter k can be solved in $f(k) \cdot n^{O(g(k))}$ time, where f and g are both computable functions.

In this thesis we focus on the parameterized complexity of transversal and connectivity problems on directed graphs. This research direction has been hitherto relatively unexplored: usually the directed version of the problems require significantly different and more involved ideas than the ones for the undirected version. Furthermore, for directed graphs there are no known algorithmic meta-techniques: for example, there is no known algorithmic analogue of the Graph Minor Theory of Robertson and Seymour for directed graphs. As a result, the fixed-parameter tractability status of the directed versions of several fundamental problems such as MULTIWAY CUT, MULTICUT, SUBSET FEEDBACK VERTEX SET, ODD CYCLE TRANSVERSAL, etc. was open.

In the first part of the thesis, we develop the framework of *shadowless solutions* for a general class of transversal problems in directed graphs. For this class of problems, we reduce the problem of finding a solution in FPT time to that of finding a shadowless solution. Since shadowless solutions have a good (problem-specific) structure, this provides an important first step in the design of FPT algorithms for problems on directed graphs. By understanding the structure of shadowless solutions, we are able to design the first FPT algorithms for the DIRECTED MULTIWAY CUT problem and the SUBSET DIRECTED FEEDBACK VERTEX SET problem.

In the second part of the thesis, we present *tight bounds* on the parameterized complexity of well-studied directed connectivity problems such as STRONGLY CONNECTED STEINER SUBGRAPH and DIRECTED STEINER FOREST when parameterized by the number of terminals/terminal pairs. We design new optimal XP algorithms for the aforementioned problems, and also prove matching lower bounds for existing XP algorithms. Most of our hardness results hold even if the underlying undirected graph is planar.

Finally, we conclude with some open problems regarding the parameterized complexity of transversal and connectivity problems on directed graphs.

DIRECTED GRAPHS: FIXED-PARAMETER TRACTABILITY & BEYOND

by

Rajesh Chitnis

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2014

Dedication

To my sister Ishwari and my parents Sandhya and Hemant for all their love and support

Acknowledgments

First of all, I would like to thank my thesis advisor Prof. MohammadTaghi Hajiaghayi for his excellent scientific guidance over the last 4 years. He has been a wonderful source of problems and ideas. Looking back, I can see his imprint on every aspect of my scientific development. Mohammad always encouraged me to work on problems that I found interesting, even when sometimes they might not have been aligned with his research interests. He was always willing to fund me for conferences (even when I did not have a paper to present) and also for research visits - in Summer '12 he gave me an RA while I was away from Maryland the whole summer on research visits to Hungary and Norway! Thanks Mohammad for everything, and I look forward to our continued collaboration.

In grad school I was extremely fortunate to have had a chance to work with various researchers who were very generous with their time and ideas. Daniel Lokshtanov hosted me on a visit to San Diego which included a wonderful hike. I spent a month at Bergen, Norway visiting Fedor Fomin and Petr Golovach which led to our collaboration on the Anchored *k*-Core problem. I learned a lot about cut problems from Marek Cygan during the six months that he visited us at Maryland. I owe whatever little knowledge about streaming algorithms that I have to Morteza Monemizadeh. Two people have had a significant impact on me during my PhD studies - Daniel Marx and Saket Saurabh. Daniel has been a wonderful collaborator - I have learnt a lot from his intuition about important separators. Thanks also for hosting me for two months in Budapest. Saket has been a good friend and also the main reason why I began to work in FPT in the first place - in the last semester of my undergrad studies, I attended two courses by Saket on Parameterized Complexity and Advanced Graph Theory. These two courses and spending the ensuing summer working with Saket was enough for me to decide to pursue FPT in my PhD. I also spent a wonderful three months at TTI-Chicago working with Julia Chuzhoy.

At Maryland one of the best things that happened to me was meeting people from various countries and getting to know their perspectives on life. I hope it has made me less narrow-minded than I was before. In the department, I would like to thank Samir Khuller and Aravind Srinivasan for their continued encouragement and sound advice throughout my PhD. Many thanks to Aravind Srinivasan, Erik Demaine, Dana Nau, Subramanian Raghavan for serving on my dissertation committee and informative comments on the thesis. I was able to remain oblivious to the intricacies of department paperwork thanks to Jenny Story, Fatima Bangura and Sharron McElroy. Many thanks to all my friends in the department - Ioana, Theo, Ben, Vahid, Reza, Kotaro, Aishwarya, Vikas, Abdul, Jayanta, Govind, Udayan, Kanthi, Manish, Amit, David, Hamid, Faezeh and some others whom I am surely forgetting right now. Doing a PhD becomes very monotonous if one does not have fun people around - many thanks to Varun, Raka, Prat Di, Puneet, Arvind and Vivek for various trips, tasty food and cold beer. Finally, thanks to my Persian group of friends for allowing me into their soccer team and also for the several intramural championships that we won.

I was very lucky to pursue my undergraduate studies at a unique institution like Chennai Mathematical Institute (CMI) which provided a lot of academic freedom. After misusing it for a while, I later was able to take advantage of it and attend various grad courses to see which subareas interest me more. Prof. K. Narayan Kumar gave me a second chance after I did not perform well in the first course with him, and I am glad for that. My first research interaction was with Prof. Samir Datta. Unfortunately our work together did not lead to a paper, but when my interests became a bit more focused Samir referred me to Prof. Sunil Chandran at IISc, Bangalore. Working with Sunil was a very good experience which resulted in my first publication. On the non-academic side, I want to thank all my friends at CMI (most of whom are part of the `bug07` mailing list).

Leaving the most important for the last, I want to thanks my parents, sister and my extended family. My extended family has always been very encouraging in all my endeavours, and provide important support to my family in my absence. Talking to friends from high school is always fun and very relaxing. Special mention goes to Harshad, Saurabh, Varun, Ambarish and Neeraj. My parents Sandhya and Hemant have been my role-models in my entire life. Everyday I understand more and more how truly awesome they are as parents. Talking to my sister Ishwari always gives me a different perspective of life. She is very enthusiastic about life and is always able to cheer me up. This thesis is a small token of my love and appreciation for my family - no words or actions can ever come close to thank you for all that you do for me.

I gratefully acknowledge the support of the following grants during my PhD studies: NSF CAREER award 1053605, NSF grant CCF-1161626, DARPA/AFOSR grant FA9550-12-1-0423, ONR YIP award N000141110662, DARPA/AFRL award FA8650-11-1-7162, NSF CAREER award 0844872, European Research Council (ERC) grant 280152 “PARAMTIGHT”, European Research Council (ERC) grant 267959 “Rigorous Theory of Preprocessing”, a Summer International Research Fellowship from University of Maryland, and a Simons Award for Graduate Students in Theoretical Computer Science.

Table of Contents

List of Figures

CHAPTER 1

Introduction and Overview

1.1 Notation

General Notation: $\mathbb{N}$ denotes the set of positive integers. $\mathbb{R}$ denotes the set of real numbers. We use $[n]$ to denote the set $\{1,2,3,\ldots,n\}$. A function $f : A \to B$ is said to be injective if and only if for all $a, a' \in A$ the condition $f(a) = f(a')$ implies $a = a'$. A (n,r,k)-splitter is a family of functions from $[n] \to [k]$ such that for every $M \subseteq [n]$ with $|M| = r$, at least one of the functions in the family is injective on M. A function $f : 2^U \to \mathbb{N} \cup \{0\}$ is *submodular* if for all $A, B \subseteq U$ we have $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. We use $\binom{n}{r}$ to denote the number of subsets of $[n]$ which have size r. We use $\binom{n}{\leq r}$ to denote the number of subsets of $[n]$ which have size at most r. A multiset is a generalization of a set where some elements can appear more than once. A set with a specific property is minimal (resp. maximal) if no proper subset (resp. superset) of the set satisfies the property.

We say that $f(n) = O(g(n))$ if there exist constants $c > 0$ and $n' \geq 0$ such that for each $n \geq n'$ we have $f(n) \leq c \cdot g(n)$. We say that $f(n) = \Omega(g(n))$ if there exist constants $c > 0$ and $n' \geq 0$ such that for each $n \geq n'$ we have $f(n) \geq c \cdot g(n)$. We say that $f(n) = \Theta(g(n))$ if both the conditions $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ hold true. We say that $f(n) = o(g(n))$ if for every constant $\varepsilon > 0$ there exists a constant N_ε such that for each $n \geq N_\varepsilon$ we have $f(n) \leq \varepsilon \cdot g(n)$. The O^* notation hides factors which are polynomial in the input size. For example, a function $g(k,n)$ can be written as $O^*(f(k))$ if and only if $g(k,n) = O(f(k) \cdot n^{O(1)})$. The $\tilde{O}$ notation hides factors which are polynomial in the logarithm of the input size. For example, a function $g(k,n)$ can be written as $\tilde{O}(f(k))$ if and only if $g(k,n) = O(f(k) \cdot (\log n)^{O(1)})$.

Undirected Graphs: An undirected graph is a pair $G = (V,E)$ where V is called as the vertex set and E is called as the edge set. An edge between vertices v and w is denoted by $\{v,w\}$ or vw. The edge vw is said to be incident with the vertices v and w. If vw is an edge then v and w are said to be adjacent or neighbors. All graphs considered in this thesis are simple and finite (unless otherwise stated explicitly). We do not allow loops (edges of the type vv) or more than one edge between two vertices. The *open neighborhood* of v is given by $N(v) = \{w : vw \in E\}$. The degree of a vertex v is the number of its neighbors, i.e, $|N(v)|$. The *closed neighborhood* of v is given by $N[v] = N(v) \cup \{v\}$. A *walk* is a sequence of vertices $v_1, v_2, \ldots, v_r$ such that $v_i v_{i+1} \in E$ for each $i \in [r-1]$. A walk that does not repeat a vertex is called as a *path*. A *closed walk* is a walk that starts and ends at the same vertex. A closed walk that repeats exactly one vertex is called a *cycle*.

A graph $H = (V_H, E_H)$ is said to be a *subgraph* of $G = (V_G, E_G)$ if $V_H \subseteq V_G$ and

$E_H \subseteq E_G$. Given a set $S \subseteq V$ the subgraph induced on S is denoted by $G[S]$: its vertex set is S and the edge set is the set of all those edges in G which have both endpoints in S. For $S \subseteq V$ we use $G \setminus S$ to denote the graph $G[V \setminus S]$. For $S \subseteq E$ we use $G \setminus S$ to denote the graph $(V, E \setminus S)$. A graph $G = (V,E)$ is *bipartite* if there exists a set $V' \subseteq V$ such that both the graphs $G[V']$ and $G[V \setminus V']$ do not have any edges. A graph is *planar* if it can be drawn in the plane or the sphere without crossings.

Directed Graphs: In directed graphs, the edge set contains ordered pairs of vertices. An edge from v to w is represented as $v \to w$ or (v,w). In this case, we call v as the *tail* and w as the *head*. Again, we only consider simple, finite directed graphs, i.e., no edges of the type (v,v) or no multiple edges between two vertices. The *out-neighborhood* of v is given by $N^+(v) = \{w : (v,w) \in E\}$. The out-degree of a vertex v is the number of its out-neighbors, i.e, $|N^+(v)|$. The *in-neighborhood* of v is given by $N^-(v) = \{w : (w,v) \in E\}$. The in-degree of a vertex v is the number of its in-neighbors, i.e, $|N^-(v)|$. A *walk* is a sequence of vertices $v_1, v_2, \ldots, v_r$ such that $(v_i, v_{i+1}) \in E$ for each $i \in [r-1]$. A walk that does not repeat a vertex is called as a *path*. We denote a path from v to w by $v \rightsquigarrow w$. A *closed walk* is a walk that starts and ends at the same vertex. A closed walk that repeats exactly one vertex is called a *cycle*. A directed graph without a cycle is called as a directed acyclic graph (DAG). Every directed acyclic graphs has a *topological ordering*: an ordering such that for each edge the tail precedes the head in the ordering.

Minors and Treewidth: Given an edge $e = \{u,v\}$ in an undirected graph G, the contraction of e in G is the result of identifying the vertices u and v in G and removing all loops and duplicate edges (the resulting graph is denoted by G/e). A graph H obtained by a sequence of such edge contractions starting from G is said to be a *contraction* of G. A graph H is a *minor* of G if H is a subgraph of some contraction of G. A graph class $\mathcal{G}$ is *minor-closed* if any minor of any graph in $\mathcal{G}$ is also a member of $\mathcal{G}$. A minor-closed graph class $\mathcal{G}$ is *H-minor-free* if $H \notin \mathcal{G}$. We use the term *H-minor-free* to refer to any minor-closed graph class that excludes some fixed graph H.

A *tree decomposition* of an undirected graph G is a tree $\mathcal{T}$ in which every vertex $x \in V(\mathcal{T})$ has an assigned set of vertices $B_x \in V(G)$ (called a bag) such that the following properties are satised:

- $\bigcup_{x \in V(\mathcal{T})} B_x = V(G)$.
- For each $\{u,v\} \in E(G)$, there exists an $x \in V(\mathcal{T})$ such that $u,v \in B_x$.
- For each $v \in V(G)$, the set of vertices of $\mathcal{T}$ whose bags contain v induce a connected subtree of $\mathcal{T}$.

The *width* of a tree decomposition $\mathcal{T}$ is $\max_{x \in V(\mathcal{T})} |B_x| - 1$. The treewidth of a graph G, usually denoted by $\mathrm{tw}(G)$, is the minimum width over all possible tree decompositions of G. A $k \times k$ grid is an undirected graph with vertex set $[k] \times [k]$ where (i,j) and (i',j') are neighbors if and only if $|ii'| + |jj'| = 1$. It is known that a $k \times k$ grid has treewidth $\Theta(k)$.

1.2 Coping with NP-Completeness

"For every polynomial algorithm you have, there is an exponential algorithm I would rather run" -Alan Perlis

In his seminal 1936 paper *"On Computable Numbers, With An Application To The Entscheidungsproblem"* [Tur36] the principle of the modern computer was first described by Alan Turing. The field of algorithms received a shot in the arm with the rapid development in power and easier availability of digital computers. Many combinatorial optimization problems were recognized as having practical applications. However, the Combinatorics community at that time was not interested in algorithms. Since the graphs considered in applications were finite, they had trivial solutions which ran in finite time. Even the Operations Research community viewed a problem as "solved" if it could be formulated as an integer programming problem. Edmonds in his seminal paper *"Paths, Trees and Flowers"* [Edm65] gave an algorithm for maximum matching in general graphs whose running time was polynomial in the input size. He called such algorithms as "efficient" or "good". We quote the following paragraph from Edmonds' paper: *"There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether* or not *there exists an algorithm whose difficulty increases only algebraically with the size of the graph."* This criterion of Edmonds for calling an algorithm as efficient (or good) when its running time is polynomially bounded is now the accepted convention in theoretical computer science. We refer the interested reader to a related survey by Pulleyblank [Pul12].

The complexity class P is the set of all decision problems that can be *solved* by a deterministic Turing machine using polynomial time. The complexity class NP (nondeterministic polynomial time) is the set of all decision problems for which the instances where the answer is "YES" have proofs that can be verified polynomial time by a deterministic Turing machine[1]. In other words, P is the class of problems which have efficient solutions and NP is the class of problems which have efficient "characterizations". In his seminal paper, Cook [Coo71] showed that there is a problem called SATISFIABILITY (SAT) in the class NP having the property that every other problem in the class NP can be reduced to it in polynomial time. That is, if SAT can be solved in polynomial time then every problem in NP can also be solved in polynomial time; conversely if any problem in NP is "intractable" then the SAT problem must also be intractable. Intuitively, SAT is the *hardest* problem in the class NP. Cook also suggested that other problems like CLIQUE may also share the property of being the hardest member of the class NP. Karp [Kar72] gave a list of 21 such *hardest* problems, which came to be known as NP-complete problems. See [GJ79] for more historical information about NP-completeness, including a list of over 300 NP-complete problems. As of today, to show NP-hardness of a given problem at hand, one has a choice of several thousands of known NP-complete problems to choose to reduce from.

Over the last few decades, a lot of effort has gone into trying to resolve the central question in theoretical computer science: is $P = NP$? A large fraction of the theoreti-

[1]Alternatively, we can think of NP as the set of all decision problems where the "YES" instances can be accepted in polynomial time by a non-deterministic Turing machine: in the first step we non-deterministically guess a solution, and in the second step check deterministically whether the guessed solution is indeed valid or not.

cal computer science community seems to believe that P$\neq$NP. Under this belief there is no polynomial time algorithm that solves all the instances of an NP-complete problem exactly. However as noted by Garey and Johnson [GJ79], hardness results such as NP-hardness should merely constitute the beginning of research. The traditional way to combat this intractability is to design approximation algorithms or randomized algorithms which run in polynomial time. These methods have their own shortcomings: we either get an approximate solution or lose the guarantee that the output is always correct. In order to not lose the guarantee on the optimality of the solution, one is forced to consider algorithms which require time beyond polynomial time. Two such paradigms are exact-exponential algorithms and fixed-parameter algorithms.

We now briefly describe the aforementioned four algorithmic approaches to dealing with NP-completeness.

1.2.1 Approximation Algorithms

The area of approximation algorithms is one of the most active areas of research in theoretical computer science. The output is allowed to be a (provable) factor away from the optimal solution but we are forced stay within polynomial time. Let $\alpha \geq 1$. For minimization problems, we say that an algorithm is an α-approximation if it outputs a solution that is at most α times larger than the optimum. Similarly, for maximization problems we say that an algorithm is an α-approximation if it outputs a solution that is at least an $(1/\alpha)$-fraction of the optimum. We now look at various types of known approximation ratios:

Polynomial: Outputting a single vertex gives an n-approximation for CLIQUE. This is optimal as Håstad [Hås96] and Zuckerman [Zuc06] showed that CLIQUE cannot be approximated to within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless P=NP.

Polylog: The well-known greedy algorithm for SET COVER gives an $\ln n$-approximation, where n is the size of the universe. This is optimal as Feige [Fei98] showed that SET COVER cannot be approximated to within a factor better than $\ln n$ unless we have that NP$\subseteq$ DTIME$(n^{\log\log n})$.

Constant Factor: The simple algorithm which takes one vertex from each edge of a maximal matching gives an 2-approximation for VERTEX COVER. Dinur and Safra [DS05] showed that VERTEX COVER cannot be approximated to within a factor better than 1.3606 unless P=NP. Khot and Regev [KR08] showed that it is not possible to give an $(2-\varepsilon)$-approximation for any $\varepsilon > 0$ under the Unique Games Conjecture of Khot [Kho02].

Polynomial Time Approximation Scheme (PTAS): A PTAS is an algorithm which given a parameter $\varepsilon > 0$ outputs a solution within a factor $(1+\varepsilon)$ of the optimal and has running time $n^{f(\frac{1}{\varepsilon})}$ for some function f, i.e., the running time is polynomial in n for every fixed ε but can be different as ε varies. There is a tradeoff between the approximation ratio and the running time. The most famous example of a PTAS is for the EUCLIDEAN TRAVELING SALESMAN problem due to Arora [Aro98].

We refer the reader to the books by Vazirani [Vaz02], Hochbaum [Hoc97] and Shmoys and Williamson [WS11] for more information on approximation algorithms.

Approximation Beyond Polynomial Time: So far we only looked at approximation algorithms which ran in polynomial time. Alternatively, one can also consider exponential time approximation algorithms [CKW09, BH06, BEP09, DGHK01, VWW06] or FPT approximation algorithms [Mar08, CHK13, BEKP13, DFMR08, Mar13, GG07]

1.2.2 Randomized Algorithms

Randomized algorithms are algorithms that make random choices during their execution. Randomized algorithms are usually simpler, and are often the first step towards obtaining a deterministic algorithm. There are two types of randomized algorithms: Monte Carlo and Las Vegas. A Monte Carlo algorithm is a randomized algorithm whose running time is deterministic, but whose output might be wrong with an (ideally small) probability that one can bound. A Las Vegas algorithm is an algorithm which always produces the correct output, but the randomness lies in the running time of the algorithm. We refer the reader to the books by Motwani and Raghavan [MR95] and Mitzemacher and Upfal [MU05] for more information on randomized algorithms.

In polynomial time randomized algorithms, one usually tries to do a randomized selection so that the probability of success is (provably) at most polynomially small. On the other hand, randomized FPT algorithms are designed by doing a randomized selection in order to satisfy a bounded number of constraints. In this case, the probability of success is (provably) at most exponentially small in the parameter. See the survey by Marx [Mar12b] on randomized techniques for FPT algorithms.

1.2.3 Exact Exponential Algorithms

All problems in NP have trivial exponential time algorithms which simply search and verify all the witnesses [Wig07]. Any algorithm which beats the brute-force algorithm can be thought of as making a clever search in the big space of all candidate solutions. For most graph problems, the trivial algorithm which enumerates all subsets and checks (in polytime) whether each subset is a solution or not runs in time $O^*(2^n)$ where n is the number of vertices in the graph. The goal in exact exponential algorithms is to reduce the base of the exponent, i.e., to design algorithms which run in $O^*((2-\varepsilon)^n)$ for some constant $\varepsilon > 0$. Breaking this barrier turned out to be highly non-trivial for several problems (see [Bjö14, BFPV13, CFL$^+$13, FGPR08, Raz07] among others) and techniques such as Inclusion-Exclusion Principle, Measure and Conquer, Subset Convolution, etc. were used. In some cases, even obtaining an algorithm which runs in $O^*(2^n)$ is not trivial. Consider the TRAVELLING SALESMAN problem (TSP): given n distinct cities $\{c_1, c_2, \ldots, c_n\}$ and pairwise distances $d(c_i, c_j)$ for each $1 \leq i \neq j \leq n$, find a permutation of minimum distance which visits all the cities and returns to its starting point. Note that the trivial algorithm runs in $O^*(n!)$ time. Using dynamic programming Bellman [Bel62] and Held and Karp [HK62] gave an algorithm which runs in $O^*(2^n)$ time. Despite many efforts over the last 50 years, no one has been able to design a faster algorithm.

See the book by Fomin and Kratsch [FK10] for more information on this exciting and relatively new area of algorithms.

1.2.4 Fixed-Parameter Algorithms

In this thesis we consider the paradigm of Parameterized Complexity which is essentially a two-dimensional analogue of P vs NP. Unlike classical complexity, we define a parameter which is a non-negative integer often denoted by k. We perform a refined multivariate analysis of the running time by expressing it as a function of both the input size n and the parameter k, instead of expressing it as solely as a function of n. The goal is to design algorithms that work efficiently if the parameters of the input instance are small, even if the size of the input is large. More precisely, we say that an NP-hard problem is fixed-parameter tractable (FPT) with respect to parameter k if the problem can be solved in time $f(k) \cdot n^{O(1)}$ where f is a computable function and n is the input size. A straightforward generalization is to define more than one parameter.

Parameterized complexity allows us to completely determine the effect of a parameter on the complexity of a certain problem. In particular, if we can design an FPT algorithm for a problem with respect to a parameter k then this means that instances of the problem where k is small are easier to handle than others. This theory of multivariate analysis has found applications to problems in varied areas such as social networks, game theory, coding theory, machine learning, etc. We refer the reader to the books by Downey and Fellows [DF99, DF13], Flum and Grohe [FG06] and Niedermeier [Nie06] for more information on parameterized complexity.

1.3 Parameterized Complexity

"The future of algorithms is multivariate"

-Robert G. Downey and Michael R. Fellows

In this section, we give a formal introduction to Parameterized Complexity. We follow the notation of Downey and Fellows [DF13].

Let Σ be a finite alphabet and $\mathbb{N}$ be the set of natural numbers. A parameterized problem (or language) Q is a subset of $\Sigma^* \times \mathbb{N}$. An instance of Q is given by $\langle x,k\rangle$ where k is called the parameter. For a fixed $\ell \in \mathbb{N}$, we call $Q_\ell = \{\langle x,\ell\rangle \ : \ \langle x,\ell\rangle \in Q\}$ as the ℓ^{th} slice of Q.

The class XP: A problem $Q \in$ XP if $Q_k \in$ P for each $k \in \mathbb{N}$ (perhaps with a different algorithm for each k). That is, there exists an algorithm that for a given instance $\langle x,k\rangle$ decides whether $\langle x,k\rangle \in Q$ in time $f(k)\cdot|x|^{O(g(k))}$ where f,g are arbitrary computable functions of k.

The class FPT: A problem $Q \in$ FPT if there exists an algorithm that for a given instance $\langle x,k\rangle$ decides whether $\langle x,k\rangle \in Q$ in time $f(k)\cdot|x|^{O(1)}$ where f is an arbitrary computable function of k.

Parameterized Reductions: A parameterized reduction from a parameterized problem Q to a parameterized problem Q' is an algorithm that given an instance $\langle x,k\rangle$ of Q outputs in time $f(k)\cdot|x|^{O(1)}$ an equivalent instance $\langle x',k'\rangle$ of Q' such that $k' \leq g(k)$, where f and

g are arbitrary computable functions.

The W-hierarchy[2]: A *Boolean circuit* consists of input gates, negation gates, AND gates, OR gates and a single output gate. In the WEIGHTED CIRCUIT SATISFIABILITY problem we are given a Boolean circuit C and an integer k, and the objective is to decide if there is an assignment which sets k inputs to true such that the output is true. The *depth* of a circuit is the maximum length of a path from an input to the output. A gate is *large* if it has more than 2 inputs. The *weft* of a circuit is the maximum number of large gates on a path from an input to the output. Let $C[t,d]$ be the set of all circuits having weft at most t and depth at most d.

A problem $Q \in$ W[t] if there exists a constant d such that there is a parameterized reduction from Q to WEIGHTED CIRCUIT SATISFIABILITY of $C[t,d]$. In the WEIGHTED n-SATISFIABILITY problem we are given an n-CNF Boolean formula and a parameter k, and the question is to find whether there is a satisfying assignment which sets k variables to true. The following theorem which is an analog of Cook's theorem was proved by Downey and Fellows [DF95]:

Theorem 1.1. *The* WEIGHTED n-SATISFIABILITY *problem is W[1]-complete.*

W[P] is the class of problems that admit a parameterized reduction from the WEIGHTED CIRCUIT SATISFIABILITY problem. The following set of classes is known as the W-hierarchy:

$$\text{FPT} = \text{W}[0] \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \ldots \subseteq \text{W}[\text{P}] \subseteq \text{XP}$$

[2]Refer to the slides by Marx [Marb]

It is conjectured that each of the containment relations above is proper, but the only known result (via a diagonalization argument) is FPT $\subset$ XP. It is widely believed that even FPT $\neq W[1]$, and hence a W[t]-hardness result (for any $t \in \mathbb{N}$) is a strong indication that the problem does not admit a FPT algorithm. For the purposes of reductions, the most famous problem W[1]-complete problem is CLIQUE and the most famous W[2]-complete problem is DOMINATING SET.

1.3.1 Classical Complexity vs Parameterized Complexity

Through the example of the VERTEX COVER problem and the CLIQUE problem, we illustrate how parameterized complexity allows for a more refined analysis of the running time than classical complexity, thereby revealing additional structure about the problem at hand. Consider an undiretced graph $G = (V, E)$. It is a simple fact to check that if $S \subseteq V$ is a vertex cover of G if and only if then $V \setminus S$ is an clique in the complement graph $\overline{G}$. Since classical complexity measures the running time only with respect to the input size, any classical exact algorithm for VERTEX COVER also works for the CLIQUE problem. The fastest such exact algorithm for the VERTEX COVER problem and the CLIQUE is due to Robson [Rob86] and runs in $O(1.2108^n)$ time, where n is the number of vertices in the graph.

In contrast, parameterized complexity shows a remarkable difference between the running times of the aforementioned two problems when parameterized by the size k of the solution. First, note that both these problems have a simple brute-force XP algorithm parameterized by the size k of the solution: enumerate all vertex subsets of size k and check (in polynomial time) whether each subset forms a solution or not. This brute-force algorithm runs in time $\binom{n}{k} \cdot n^{O(1)} = n^{O(k)}$. A simple branching algorithm[3] shows that parameterized VERTEX COVER can be solved in time $2^k \cdot n^{O(1)}$, and the fastest algorithm is due to Chen et al. [CKX10] and runs in time $1.2738^k \cdot n^{O(1)}$. However for the parameterized CLIQUE problem no one was able to design an FPT algorithm, or even improve upon the brute force XP algorithm. This in fact led Downey and Fellows to develop the theory of fixed-parameter intractability. Finally, Chen et al. [CHKX06] showed that under the Exponential Time Hypothesis (ETH) [IP01, IPZ01] there is no algorithm for the parameterized CLIQUE problem which runs in time $f(k) \cdot n^{o(k)}$ for any computable function f. This shows that for the parameterized CLIQUE problem the brute-force algorithm enumerating all the $\binom{n}{k}$ subsets of size k of the vertex set is asymptotically optimal.

1.3.2 Algorithmic Techniques For Designing FPT Algorithms

1.3.2.1 Kernelization

Kernelization is an efficient preprocessing algorithm which produces a smaller, equivalent output called the "kernel". Formally, a kernelization algorithm for a parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm which takes as an instance $\langle x, k \rangle$ and outputs in

[3]For every edge, branch on choosing either vertex into the solution. At each step we branch into two directions, and the depth of the search tree is at most k.

time polynomial in $(|x|+k)$ an equivalent[4] instance $\langle x',k'\rangle$ such that $\max\{|x'|,k'\} \leq f(k)$ for some computable function f. The output instance $\langle x',k'\rangle$ is called the kernel, while the function f is called the size of the kernel.

A folklore observation is that a parameterized problem Q has an FPT algorithm if and only if it has a kernel. However, the size of the kernel might be exponentially large. In practice, one would hope to find kernels of polynomial size and indeed many problems such as FEEDBACK VERTEX SET and VERTEX COVER admit such kernels. On the other hand, it turned out to be difficult to find polynomial kernels for some problems. This led to development of methods [BDFH09, DvM10, Dru12, FS11] to show that certain problems do not admit polynomial kernels unless NP $\subseteq$ coNP/poly.

Kernelization is an active subarea of parameterized complexity. After proving that a new problem is FPT, the next natural step is to try for a polynomial kernel or rule out such a possibility. We refer the reader to the surveys by Lokshtanov et al. [LMS12] and Misra et al. [MRS11] for more information on kernelization.

1.3.2.2 Iterative Compression

The technique of *Iterative Compression* was introduced by Reed et al. [RSV04] who used it to show that the ODD CYCLE TRANSVERSAL problem is FPT parameterized by the size of the deletion set. Since then it has been used to obtain faster FPT algorithms [CFL+08, CLL+08, DFL+07, GGH+06, HKMN08, MR11, RO09], exact algo-

[4]By equivalent we mean that $\langle x,k\rangle \in Q \Leftrightarrow \langle x',k'\rangle \in Q$

rithms [FGK$^+$10], kernels [DFRS04] and FPT approximation algorithms [MR09]. The technique of iterative compression can be used for hereditary minimization problems, i.e., problems where the conditions H is a subgraph of G and S is a solution for G imply $G[S \cap H]$ is a solution for H. We now illustrate the use of iterative compression through the example of the ODD CYCLE TRANSVERSAL (OCT) problem: given an undirected graph $G = (V,E)$ and an integer k, the question is whether there exists a set $S \subseteq V$ of size at most k such that $G \setminus S$ is odd-cycle free, i.e., bipartite.

The first idea is that by paying a factor of $O(n)$ in the running time, it is enough to solve a modified *compression* version of the problem where we are given a solution S' of size $k+1$. Suppose we have a blackbox which can solve the *compression* version of the OCT problem in $f(k) \cdot n^c$ for some function f and constant c. Let $V = \{v_1, v_2, \ldots, v_n\}$ and for each $1 \leq i \leq n$ let $V_i = \{v_1, v_2, \ldots, v_i\}$. Observe that the OCT problem is hereditary: if G has a solution S of size k then $S \cap V_i$ is a solution for $G[V_i]$ for each $1 \leq i \leq n$. We now give a $f(k) \cdot n^{c+1}$ time algorithm for OCT. Note that V_{k+1} is a (trivial) solution of size $k+1$ for the graph $G[V_{k+1}]$. Run the blackbox algorithm for the compression problem. If it says NO then $G[V_{k+1}]$ does not have a solution of size k, and hence neither does G. Otherwise it returns a set, say S_{k+1}, of size k which is a solution of size k for $G[V_{k+1}]$. Now, $S_{k+1} \cup \{v_{k+2}\}$ is a solution of size $k+1$ for the graph $G[V_{k+2}]$. Again we run the blackbox algorithm for the compression version and so on. If the blackbox algorithm ever returns NO, then we can safely answer that G does not have a solution of size k. Otherwise after $(n-k)$ calls to the blackbox algorithm we obtain a solution of size k for

G.

The second idea is that by paying a factor of $O(2^k)$ in the running time, it is enough to solve a *disjoint* version of the *compression* version of the problem where we are given a solution S' of size $k+1$ and we need to find a solution S of size k such that $S \cap S = \emptyset$. This is easy to see: we guess all the 2^k choices for the intersection $S'' = S \cap S'$, and then solve the *disjoint compression* version of the problem on the graph $G \setminus S''$ with the parameter $k - |S''|$.

Finally, Reed et al. [RSV04] gave an FPT algorithm for the *disjoint compression* version of the OCT problem which implied an FPT algorithm for the general OCT problem with an additional factor of $O(2^k \cdot n)$ in the running time.

1.3.2.3 Bidimensionality

WIN-WIN approaches have a long and storied history in the development of FPT algorithms: see [AKK+11, CCH+12, CMPP13, iKT11] for some recent examples. We perform a polynomial test on the input, and depending on which condition is satisfied we decide how to proceed with the next step. The most famous representative of the WIN-WIN approach is via the so called *Excluded Grid Minor* theorem which is one of the cornerstones of the Graph Minors project of Robertson and Seymour [RS04].

Theorem 1.2. [RS86] *There is a function $f(k)$ such that every graph with treewidth more than $f(k)$ contains a $k \times k$ grid as a minor.*

Theorem 1.2 gives a WIN-WIN approach as follows: either we have a small treewidth (and we can do dynamic programming), or otherwise the graph contains a large grid which gives allows for some structural insights. The original work of Robertson and Seymour [RS86] gave a bound on f which was huge. Robertson, Seymour and Thomas [RST94] conjectured that the function f should be a polynomial and gave a lower bound of $f(k) = \Omega(k^2 \log k)$. In a breakthrough result, Chekuri and Chuzhoy [CC14] proved this conjecture by showing that $f(k) = k^{O(1)}$. Demaine and Hajiaghayi [DH05b] showed a *linear* parameter-treewidth bound in H-minor free graphs.

Theorem 1.3. *Let H be a fixed graph. There is a computable function f such that any graph G which excludes H as a minor and has treewidth $f(|H|) \cdot k$ contains a $k \times k$ grid as a minor.*

The theory of bidimensionality [DFHT05b] exploits the idea that many problems can be solved efficiently via dynamic programming on graphs of bounded treewidth. A problem is said to be *bidimensional* if the solution on a $k \times k$ grid is $\Omega(k^2)$, and the solution value does not increase under the operation of taking minors. Theorem 1.3 gives a WIN-WIN approach for subexponential algorithms for bidimensional problems in H-minor free graphs as follows: either we have a small treewidth (say at most $O(\sqrt{k})$) and can then use the dynamic programming algorithms for bounded treewidth graphs, or the treewidth is large (say at least $f(H) \cdot \sqrt{k}$) which implies that the graph contains $\sqrt{k} \times \sqrt{k}$ grid as a minor. This implies that the solution size is at least k, since the parameter is bidimensional.

Bidimensionality provides a general framework for designing FPT algorithms and approximation algorithms for NP-hard graph problems in broad classes of graphs. It was developed in a series of papers [DHT05, DHN+04, DFHT05a, DH04a, DFHT05b, DH04b, DFHT04, DHT06, DH05b, DH05a]. See the surveys [DH08, DFT08] for more information on bidimensionality. In the last few years there has been some new progress [FLS12, FLRS11, FLST10, FGT09] on using the theory of bidimensionality and some extensions.

1.3.2.4 Important Separators in Undirected Graphs

In a seminal paper Marx [Mar06] introduced the concept of *important separators* to deal with the UNDIRECTED MULTIWAY CUT problem. Since then it has been used (implicitly or explicitly) quite frequently [CLL09, CLL+08, CHM12, CCHM12, CFG13, KPPW12a, LM13, LR12, MR11, RO09] in the design of fixed-parameter algorithms. We now give a brief introduction on important separators in undirected graphs, and see an application to the UNDIRECTED MULTIWAY CUT problem.

Definition 1.1. (separator) *Let $G = (V,E)$ be an undirected graph and $X,Y \subseteq V$ be two disjoint non-empty sets. A set $W \subseteq V \setminus (X \cup Y)$ is called as an $X-Y$* separator *if there is no path from X to Y in $G \setminus W$. A set W is a* minimal $X-Y$ separator *if no proper subset of W is an $X-Y$ separator.*

Definition 1.2. (important separator) *Let G be an undirected graph and let $X,Y \subseteq V$ be two disjoint non-empty sets. A minimal $X-Y$ separator W is called an* important $X-Y$ separator *if there is no $X-Y$ separator W' with $|W'| \leq |W|$ and $R_{G\setminus W}(X) \subset R_{G\setminus W'}(X)$, where $R_A(X)$ is the set of vertices reachable from X in the graph A.*

Roughly speaking, an important separator is a separator of small size that is *maximal* with respect to the set of vertices on one side. The first nice property of important separators is that the number of important $X-Y$ separators of size at most k can be bounded by a function of (only) k for any two sets X,Y:

Lemma 1.1. [CLL09, Mar06]**(number of important separators)** *Let $X,Y \subseteq V(G)$ be disjoint sets in an undirected graph G. Then for every $k \geq 0$, there are at most 4^k important $X-Y$ separators of size at most k. Furthermore, we can enumerate all these separators in time $O(4^k \cdot k(|V(G)+|E(G)|))$.*

The 4^k upper bound is almost tight: Marx [Mara] showed that there are instances with $\geq \frac{4^k}{\mathrm{poly}(k)}$ important separators of size k. The second nice property of important separators is the so called "pushing lemma". First we define the UNDIRECTED MULTIWAY CUT problem:

UNDIRECTED MULTIWAY CUT
Input : An undirected graph $G=(V,E)$, an integer k and a set of terminals $T = \{t_1,t_2,\dots,t_\ell\}$.
Output : A set $S \subseteq V(G)$ of size at most k such that $G \setminus S$ has no $t_i - t_j$ path for any $1 \leq i \neq j \leq \ell$, or "NO" if such a set does not exist.

Any solution of the UNDIRECTED MULTIWAY CUT problem contains a $t_1 - T \setminus t_1$ separator. The pushing lemma says that we can always branch on choosing one of the 4^k important separators.

Lemma 1.2. [Mar06]**(pushing lemma)** *Let S be a solution for the* UNDIRECTED MULTIWAY CUT *problem. Then there is another solution S' such that $|S'| \leq |S|$ and S' contains an $t_1 - T \setminus t_1$ important separator.*

The previous two lemmas give a $O^*(4^{k^2})$ branching algorithm for the UNDIRECTED MULTIWAY CUT problem: at each step we branch on 4^k important separators, and this is repeated at most $k-1$ times.

1.4 Why are Directed Graphs Harder?

In the study of graph algorithms, we usually consider the undirected version before the directed version. However, in some cases we need directions to accurately model the given network and hence directed graphs are unavoidable. For example, the Internet is best viewed as a directed graph: introduce a node for every web page, and a directed edge between nodes u and v if there is a hyperlink from u to v. Unfortunately, in most cases[5] the directed versions of the problems turn out to be *much harder* than their corresponding undirected variants. Below we present some evidence of this phenomenon as seen in the areas of approximation algorithms and parameterized complexity.

1.4.1 Evidence from Approximation Algorithms

Multicut: In the (directed) MULTICUT problem, we are given a (directed) graph $G = (V,E)$ and a set of terminal pairs $T = \{(s_1,t_1,),(s_2,t_2),\ldots,(s_k,t_k)\}$ and the objective is to find a set $S \subseteq V$ of minimum size such that $G \setminus S$ has no $s_i - t_i$ (resp. $s_i \rightsquigarrow t_i$) path. For $k = 2$, the undirected version is polynomial time solvable [YKCP83] but the directed version is NP-hard [GVY04]. For the undirected version the best known approximation

[5]See [Sta] for some exceptions

ratio is $O(\log k)$ due to Garg et al. [GVY96], and Chawla et al. [CKK$^+$06] showed that under the Unique Games Conjecture of Khot [Kho02] no constant factor approximation is possible. For the directed version the best known approximation ratio is $\tilde{O}(n^{11/23})$ due to Agarwal et al. [AAC07], and Chuzhoy and Khanna [CK09] showed that it is hard to approximate within a factor better than $2^{\Omega(\log^{1-\varepsilon} n)}$ for any constant $\varepsilon > 0$ unless NP=ZPP.

Feedback Vertex Set: In the (directed) FEEDBACK VERTEX SET problem, we are given a (directed) graph $G = (V,E)$ and the objective is to find a set $S \subseteq V$ of minimum size such that $G \setminus S$ has no (directed) cycles. For the undirected version the best known approximation ratio is 2 due to Bafna et al. [BBF99]. However, for the directed version the best known approximation ratio is $\log n \log\log n$ due to Even et al. [ENSS95] and Seymour [Sey95].

Metric TSP: In the symmetric TRAVELING SALESMAN problem (TSP), we are given a complete undirected graph $G = (V,E)$ with costs on the edges such that the cost function satisfies the triangle inequality. The objective is to find a Hamiltonian cycle of minimum cost. In the asymmetric TRAVELING SALESMAN problem (TSP) the cost of the edge (u,v) need not be the same as the cost of the edge (v,u). The best known ratio for the symmetric version is $3/2$ due to Christofides [Chr76], and the best approximation ratio for the asymmetric version is $O(\log n/\log\log n)$ due to Asadpour et al. [AGM$^+$10].

1.4.2 Evidence from Parameterized Complexity

Lack of a Good Directed Width Measure? There have been several attempts to introduce directed notions of treewidth, viz. directed treewidth [JRST01a], *D*-width [Saf05], Kelly width [HK08], DAG-width [BDHK06, Obd06] and entanglement [BG04]. However, all of these notions failed to have both the nice (algorithmic) properties that treewidth possesses for undirected graphs: the width measure is *small* on many simple interesting graphs, and many hard problems become *easy* on instances of where the width measure is small. Ganian et al. [GHK+10] asked the following question: have all the attempts at obtaining a directed width measure that is both algorithmically useful and does not increase under taking minors[6] failed because there can be no such measure? The answer is YES: assuming P $\neq$ NP, for some technical definitions of the terms *sensible, algorithmically useful* and *nice structural properties* they showed that any width measure for directed graphs which satisfies the aforementioned three properties must be bounded by a function of the treewidth of the underlying undirected graph. This suggests that there are no good directed width measures to work with from the viewpoint of algorithms.

Lack of Algorithmic Graph Minor Theory for Directed Graphs? In last paragraph, we have seen that there is no good notion of minor or treewidth in directed graphs. However, one can ask the following question: even after fixing some definitions for mi-

[6]There is no agreed concept of minors in directed graphs. One of the commonly used concept is the so called *butterfly minors* [JRST01a]

nor and treewidth can one develop a good algorithmic theory (say similar to bidimensionality)? The answer seems be NO: for the notion of butterfly minors and directed treewidth [JRST01a], an Excluded Grid Minor Theorem was claimed for directed planar graphs by Johnson et al [JRST01b]. Recently, Kawarayabayashi and Kreutzer [iKK14] extended the result to directed graphs excluding a fixed minor. However, this result seems to be of structural value only as so far it has not been possible to develop any good algorithmic consequences out of it.

Gap Between Resolution of Parameterized Complexity of Undirected and Directed versions: In 2004 Marx [Mar06] showed that the undirected MULTIWAY CUT problem is FPT parameterized by the size of solution. The directed version was open until Chitnis et al. [CHM12] showed it to be FPT in 2012. The undirected MULTICUT was known to be FPT parameterized by size of solution plus number of terminal pairs since 2004 [Mar06]. Marx and Razgon [MR14a] improved this result in 2011 by showing that the problem is FPT parameterized by the size of solution alone. They also showed that the directed MULTICUT problem is W[1]-hard parameterized by solution size. The status of directed MULTICUT parameterized by the size of solution plus number of terminal pairs is still open, although it is known to be FPT on DAGs [KPPW12a].

The first FPT algorithm for the undirected FEEDBACK VERTEX SET problem was given by Mehlhorn [Meh84] almost 30 years ago. Since then, there have been a series of papers [BBYG00, Bod91, CCL10, CFL+08, DFL+07, DF99, GGH+06, KPS04, RSS06, CNP+11] giving faster algorithms. For directed graphs, the fixed-parameter tractability status of FEEDBACK VERTEX SET was a long-standing open problem (almost 16 years) until Chen et al. [CLL+08] resolved it by giving an FPT algorithm.

By introducing the technique of iterative compression, Reed et al. [RSV04] showed in 2004 that the undirected ODD CYCLE TRANSVERSAL problem is FPT parameterized by the size of the deletion set. The fixed-parameter tractability status of directed ODD CYCLE TRANSVERSAL parameterized by the size of the deletion set is still open.

1.5 Outline of Thesis

In the first part of this thesis, we develop the framework of shadowless solutions. This allows us to design the first FPT algorithms for the DIRECTED MULTIWAY CUT problem (Section 2.2) and the SUBSET DIRECTED FEEDBACK VERTEX SET problem (Section 2.3). These results have appeared in [CHM12] and [CCHM12].

In the second part of this thesis, we present *tight bounds* on the parameterized complexity of well-studied directed connectivity problems such as STRONGLY CONNECTED STEINER SUBGRAPH and DIRECTED STEINER FOREST (even if the underlying undirected graph is planar) when parameterized by number of terminals/terminal pairs. Upper bounds are presented in Chapter 3.1 and Lower bounds are presented in Chapter 3.2. These results have appeared in [CHM14]

CHAPTER 2

The Framework of Shadowless Solutions

In Section 1.4.2 we saw some evidence as to why parameterized complexity of problems on directed graphs are harder than the corresponding undirected versions. Very few general techniques are known for the directed problems unlike for the undirected ones. We try to bridge this gap by developing a *general framework* of “shadowless solutions” to obtain FPT algorithms for directed graph problems. This was introduced by Marx and Razgon [MR14a, MR11] for undirected graphs. We adapt and extend their framework in a highly non-trivial way to the following general family of problems on directed graphs:

Finding an $\mathcal{F}$-transversal for some T-connected $\mathcal{F}$
Input : A directed graph $G = (V,E)$, a positive integer k, a set $T \subseteq V$ and a set $\mathcal{F} = \{F_1, F_2, \dots, F_q\}$ of subgraphs such that $\mathcal{F}$ is T-connected, i.e., $\forall\ i \in [q]$ each vertex of F_i can reach some vertex of T by a walk completely contained in F_i and is reachable from some vertex of T by a walk completely contained in F_i.
Parameter : k
Question : Is there an $\mathcal{F}$-transversal $W \subseteq V$ with $|W| \leq k$, i.e., a set W such that $F_i \cap W \neq \emptyset$ for every $i \in [q]$?

Remark 2.1. *We emphasize here that the collection $\mathcal{F}$ is implicitly defined in a problem specific-way and we* do not *assume that it is given explicitly in the input, in fact, it is possible that $\mathcal{F}$ is exponentially large.*

We define the "shadow" of a solution X as those vertices that are disconnected from T (in either direction) after the removal of X. A common idea in [MR11, CHM12] is to ensure first that there is a solution whose shadow is empty, as finding such a shadowless solution can be a significantly easier task. Our generic framework shows that for the $\mathcal{F}$-transversal problems defined above, we can invoke the *random sampling of important separators* technique and obtain a set which is disjoint from a minimum solution and covers its shadow. What we do with this set, however, is problem specific. Typically, given such a set, we can use (some problem-specific variant of) the "torso operation" to find an equivalent instance that has a shadowless solution. Therefore, we can focus on the simpler task of finding a shadowless solution; or more precisely, finding any solution under the guarantee that a shadowless solution exists. We believe our framework will provide a useful opening step in the design of FPT algorithms for other transversal and cut problems on directed graphs.

In the case of undirected MULTICUT [MR11], the problem of finding a shadowless solution could be reduced to an FPT problem called ALMOST 2SAT [LNR+12, RO09]. In the case of DIRECTED MULTIWAY CUT [CHM12], the problem of finding a shadowless solution can be reduced to the undirected version, which is known to be FPT [CLL09,CPPW13a,Mar06]. For SUBSET-DFVS [CCHM12], the situation turns out to be a bit more complicated. We first use the technique of *iterative compression* (see Section 1.3.2.2) to reduce the problem to an instance where we are given a solution T and we want to find a disjoint solution of size at most k. Also the "shadows" are defined with respect to the solution T that we want to compress, whereas for DIRECTED MULTIWAY CUT the shadows are defined with respect to the terminal set T. The "torso" operation for the SUBSET-DFVS problem, as it takes into account the set S and modifies it accordingly. Furthermore, even after ensuring that there is a solution T' whose shadow is empty, we are not done unlike in the case of DIRECTED MULTIWAY CUT. We then analyze the structure of the graph $G \setminus T'$ and focus on the last strongly connected component in the topological ordering of this graph, i.e., the strongly connected component which can only have incoming edges from other strongly connected components. We would like to find the subset of T' that separates this component from the rest of the graph. In most cases, a pushing argument can be used to argue that this subset of T' is an important separator, and hence we can branch on removing an important separator from the graph. However, due to the way the set S interacts with the solution T', there is a small number of vertices that behave in special way. We need surprisingly complex arguments to handle these special vertices.

2.1 General $\mathcal{F}$-transversal Problems: Covering the Shadow of a Solution

The purpose of this section is to present the "random sampling of important separators" technique for the general family of problems given by "$\mathcal{F}$-transversal for T-connected $\mathcal{F}$". The technique consists of two steps:

1. First find a set Z *small* enough to be disjoint from a solution X (of size $\leq k$) but *large* enough to cover the "shadow" of X.
2. Then define a "torso" operation that uses the set Z to reduce the problem instance in such a way that X becomes a shadowless solution of the reduced instance.

We now show that Step 1 can be efficiently performed for the general family of problems defined earlier in this chapter. The `torso` operation is problem-specific, and hence we define it separately later on for the two problems of SUBSET DIRECTED FEEDBACK VERTEX SET and DIRECTED MULTIWAY CUT. First we start by defining separators and shadows:

Definition 2.1. (separator) *Let G be a directed graph and $V^{\infty}(G)$ be the set of distinguished ("undeletable") vertices. Given two disjoint non-empty sets $X, Y \subseteq V$, we call a set $W \subseteq V \setminus (X \cup Y \cup V^{\infty})$ an $X - Y$* separator *if there is no path from X to Y in $G \setminus W$. A set W is a* minimal $X - Y$ separator *if no proper subset of W is an $X - Y$ separator.*

Note that here we explicitly define the $X - Y$ separator W to be disjoint from X and Y.

Definition 2.2. **(shadows)** *Let G be graph and T be a set of terminals. Let $W \subseteq V(G) \setminus V^{\infty}(G)$ be a subset of vertices.*

1. *The* forward shadow $f_{G,T}(W)$ *of W (with respect to T) is the set of vertices v such that W is a $T - \{v\}$ separator in G.*
2. *The* reverse shadow $r_{G,T}(W)$ *of W (with respect to T) is the set of vertices v such that W is a $\{v\} - T$ separator in G.*

The shadow *of W (with respect to T) is the union of $f_{G,T}(W)$ and $r_{G,T}(W)$.*

That is, we can imagine T as a light source with light spreading on the directed edges. The forward shadow is the set of vertices that remain dark if the set W blocks the light, hiding v from T's sight. In the reverse shadow, we imagine that light is spreading on the edges backwards. We abuse the notation slightly and write $v - T$ separator instead of $\{v\} - T$ separator. We also drop G and T from the subscript if they are clear from the context. Note that W itself is not in the shadow of W (as, by definition, a $T - v$ or $v - T$ separator needs to be disjoint from T and v), that is, W and $f_{G,T}(W) \cup r_{G,T}(W)$ are disjoint. See Figure 2.1 for an illustration.

Let G be a directed graph and $T \subseteq V(G)$. Let $\mathcal{F} = \{F_1, F_2, \ldots, F_q\}$ be a set of subgraphs of G. We define the following property:

Definition 2.3. ***(T-connected)*** *Let $\mathcal{F} = \{F_1, F_2, \ldots, F_q\}$ be a set of subgraphs of G. For a set $T \subseteq V$, we say that $\mathcal{F}$ is* T-connected *if for every $i \in [q]$, each vertex of the subgraph F_i can reach some vertex of T by a walk completely contained in F_i and is reachable from some vertex of T by a walk completely contained in F_i.*

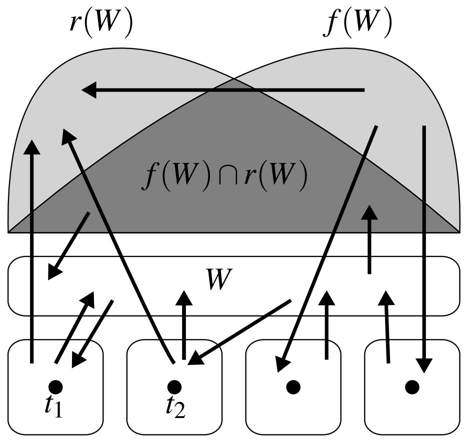


Figure 2.1: For every vertex $v \in f(W)$, the set W is a $T - v$ separator. For every vertex $w \in r(W)$, the set W is a $w - T$ separator. For every vertex $y \in f(W) \cap r(W)$, the set W is both a $T - y$ and $y - T$ separator. Finally for every $z \in V(G) \setminus [W \cup r(W) \cup f(W) \cup T]$, there are both $z - T$ and $T - z$ paths in the graph $G \setminus W$.

For a set $\mathcal{F}$ of subgraphs of G, an *$\mathcal{F}$-transversal* is a set of vertices that intersects each subgraph in $\mathcal{F}$.

Definition 2.4. *(**$\mathcal{F}$-transversal**) Let $\mathcal{F} = \{F_1, F_2, \ldots, F_q\}$ be a set of subgraphs of G. Then $W \subseteq V(G)$ is said to be an $\mathcal{F}$-*transversal *if $\forall\ i \in [q]$ we have $F_i \cap W \neq \emptyset$.*

The main result of this section is a randomized algorithm for producing a set that covers the shadow of some $\mathcal{F}$-transversal:

Theorem 2.2. (randomized covering of the shadow) *Let $T \subseteq V(G)$. There is an algorithm* RandomSet(G, T, k) *that runs in $O^*(4^k)$ time and returns a set $Z \subseteq V(G)$ such that for any set $\mathcal{F}$ of T-connected subgraphs, if there exists an $\mathcal{F}$-transversal of size $\leq k$, then the following holds with probability $2^{-O(k^2)}$: there is an $\mathcal{F}$-transversal X of size $\leq k$ such that*

1. $X \cap Z = \emptyset$ and

2. *Z covers the shadow of X.*

Note that $\mathcal{F}$ is *not* an input of the algorithm described by Theorem 2.2: the set Z constructed in the above theorem works for *every* T-connected set $\mathcal{F}$ of subgraphs. Therefore, issues related to the representation of $\mathcal{F}$ do not arise. Using the theory of splitters, we also prove the following derandomized version of Theorem 2.2:

Theorem 2.3. (deterministic covering of the shadow) *Let $T \subseteq V(G)$. We can construct a set $\{Z_1, Z_2, \ldots, Z_t\}$ with $t = 2^{O(k^2)} \log^2 n$ in time $O^*(2^{O(k^2)})$ such that for any set $\mathcal{F}$ of T-connected subgraphs, if there exists an $\mathcal{F}$-transversal of size $\leq k$, then there is an $\mathcal{F}$-transversal X of size $\leq k$ such that for at least one $i \in [t]$ we have*

1. *$X \cap Z_i = \emptyset$ and*
2. *Z_i covers the shadow of X.*

Sections 2.1.1–2.1.3 are devoted to the proofs of Theorems 2.2–2.3.

2.1.1 Important Separators and Random Sampling

This subsection reviews the notion of important separators and the random sampling technique introduced in [MR11] for undirected graphs, and adapts it to directed graphs.

Important separators: In Section 1.3.2.4 we have seen the use of important separators to design FPT algorithms in undirected graphs. Now we define and use this concept in the setting of directed graphs. Roughly speaking, an important separator is a separator of small size that is *maximal* with respect to the set of vertices on one side.

Definition 2.5. (important separator) *Let G be a directed graph and let $X, Y \subseteq V$ be two disjoint non-empty sets. A minimal $X - Y$ separator W is called an* important $X - Y$ separator *if there is no $X - Y$ separator W' with $|W'| \leq |W|$ and $R^+_{G \setminus W}(X) \subset R^+_{G \setminus W'}(X)$, where $R^+_A(X)$ is the set of vertices reachable from X in the graph A.*

Let X, Y be disjoint sets of vertices of an *undirected graph*. Then for every $k \geq 0$, it is known [CLL09, Mar06] that there are at most 4^k important $X - Y$ separators of size at most k for any sets X, Y. The next lemma shows that the same bound holds for important separators even in directed graphs. For the sake of continuity, we defer the proof to Section 2.1.4

Lemma 2.1. (number of important separators) *Let $X, Y \subseteq V(G)$ be disjoint sets in a directed graph G. Then for every $k \geq 0$ there are at most 4^k important $X - Y$ separators of size at most k. Furthermore, we can enumerate all these separators in time $O(4^k \cdot k(|V(G) + |E(G)|))$.*

For ease of notation, we now define the following collection of important separators:

Definition 2.6. *Given a graph G, a set $T \subseteq V(G)$, and an integer k, the set $\mathcal{I}_k$ contains the set $W \subseteq V(G)$ if W is an important $v - T$ separator of size at most k in G for some vertex v in $V(G) \setminus T$.*

Remark 2.4. It follows from Lemma 2.1 that $|\mathcal{I}_k| \leq 4^k \cdot |V(G)|$ and we can enumerate the sets in $\mathcal{I}_k$ in time $O^*(4^k)$.

We now define a special type of shadows which we use later for the random sampling:

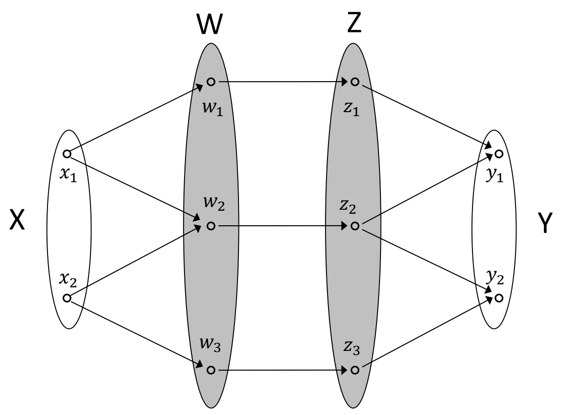


Figure 2.2: W is a minimal $X-Y$ separator, but it is not an important $X-Y$ separator as Z satisfies $|Z|=|W|$ and $R^{+}_{G\setminus W}(X)=X\subset X\cup W=R^{+}_{G\setminus Z}(X)$. In fact it is easy to check that the only important $X-Y$ separator of size 3 is Z. If $k\geq 2$ then the set $\{z_1,z_2\}$ is in $\mathcal{I}_k$, since it is an important x_1-Y separator of size 2. Finally, x_1 belongs to the "exact reverse shadow" of each of the sets $\{w_1,w_2\},\{w_1,z_2\},\{w_2,z_1\}$ and $\{z_1,z_2\}$, since they are all minimal x_1-Y separators. However x_1 does not belong to the exact reverse shadow of the set W as it is not a minimal x_1-Y separator.

Definition 2.7. **(exact shadows)** *Let G be a directed graph and $T\subseteq V(G)$ a set of terminals. Let $W\subseteq V(G)\setminus V^{\infty}(G)$ be a set of vertices. Then for $v\in V(G)$ we say that*

1. *v is in the* "exact reverse shadow" *of W (with respect to T) if W is a minimal $v-T$ separator in G, and*
2. *v is in the* "exact forward shadow" *of W (with respect to T) if W is a minimal $T-v$ separator in G.*

We refer the reader to Figure 2.2 for examples of Definitions 2.5, 2.6 and 2.7. The exact reverse shadow of W is a subset of the reverse shadow of W: it contains a vertex v only if every vertex $w\in S$ is "useful" in separating v, i.e., vertex w can be reached from v and T can be reached from w. Similarly for the forward shadow. This slight difference between the shadow and the exact shadow will be crucial in the analysis of the algorithm

(Section 2.1.3).

The weaker version of the random sampling described in Section 2.1.1 (Theorem 2.6) randomly selects members of $\mathcal{I}_k$ and creates a subset by taking the union of the exact reverse shadows of these sets. The following lemma will be used to give an upper bound on the probability that a vertex is covered by the union.

Lemma 2.2. *Let z be any vertex. Then there are at most 4^k members of $\mathcal{I}_k$ that contain z in their exact reverse shadows.*

For the proof of Lemma 2.2, we need to establish first the following:

Lemma 2.3. *If $W \in \mathcal{I}_k$ and v is in the exact reverse shadow of W, then W is an important $v-T$ separator.*

Proof. Let w be the witness that W is in $\mathcal{I}_k$, i.e., W is an important $w-T$ separator in G. Let v be any vertex in the exact reverse shadow of W, which means that W is a minimal $v-T$ separator in G. Suppose that W is not an important $v-T$ separator. Then there exists a $v-T$ separator W' such that $|W'| \leq |W|$ and $R^+_{G\setminus W}(v) \subset R^+_{G\setminus W'}(v)$. We will arrive to a contradiction by showing that $R^+_{G\setminus W}(w) \subset R^+_{G\setminus W'}(w)$, i.e., W is not an important $w-T$ separator.

First, we claim that W' is a $(W \setminus W') - T$ separator. Suppose that there is a path P from some $x \in W \setminus W'$ to T that is disjoint from W'. As W is a minimal $v-T$ separator, there is a path Q from v to x whose internal vertices are disjoint from W. Furthermore, $R^+_{G\setminus W}(v) \subset R^+_{G\setminus W'}(v)$ implies that the internal vertices of Q are disjoint from W' as well.

Therefore, concatenating Q and P gives a path from v to T that is disjoint from W', contradicting the fact that W' is a $v-T$ separator.

We show that W' is a $w-T$ separator and its existence contradicts the assumption that W is an important $w-T$ separator. First we show that W' is a $w-T$ separator. Suppose that there is a $w-T$ path P disjoint from W'. Path P has to go through a vertex $y \in W \setminus W'$ (as W is a $w-T$ separator). Thus by the previous claim, the subpath of P from y to T has to contain a vertex of W', a contradiction.

Finally, we show that $R^{+}_{G\setminus W}(w) \subseteq R^{+}_{G\setminus W'}(w)$. As $W \neq W'$ and $|W'| \leq |W|$, this will contradict the assumption that W is an important $w-T$ separator. Suppose that there is a vertex $z \in R^{+}_{G\setminus W}(w) \setminus R^{+}_{G\setminus W'}(w)$ and consider a $w-z$ path that is fully contained in $R^{+}_{G\setminus W}(w)$, i.e., disjoint from W. As $z \notin R^{+}_{G\setminus W'}(w)$, path Q contains a vertex $q \in W' \setminus W$. Since W' is a minimal $v-T$ separator, there is a $v-T$ path that intersects W' only in q. Let P be the subpath of this path from q to T. If P contains a vertex $r \in W$, then the subpath of P from r to T contains no vertex of W' (as $z \neq r$ is the only vertex of W' on P), contradicting our earlier claim that W' is a $(W \setminus W') - T$ separator. Thus P is disjoint from W, and hence the concatenation of the subpath of Q from w to q and the path P is a $w-T$ path disjoint from W, a contradiction. $\square$

Lemma 2.2 easily follows from Lemma 2.3. Let J be a member of $\mathcal{I}_k$ such that z is in the exact reverse shadow of J. By Lemma 2.3, J is an important $z-T$ separator. By Lemma 2.1, there are at most 4^k important $z-T$ separators of size at most k and hence z belongs to at most 4^k exact reverse shadows.

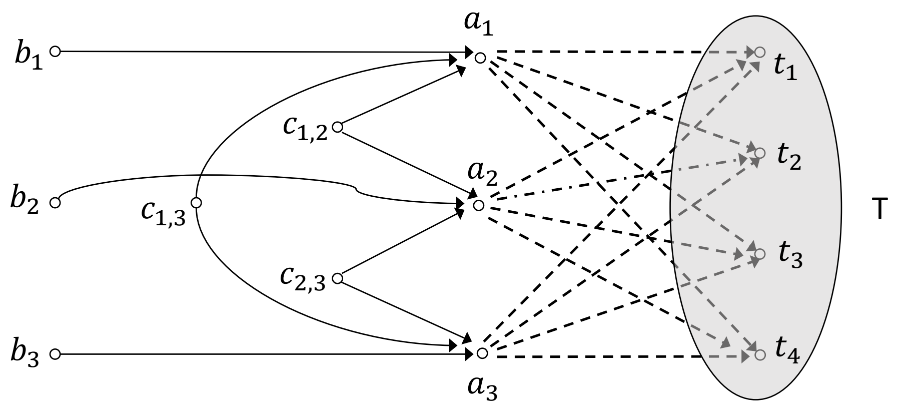


Figure 2.3: An illustration of Remark 2.5 in the special case when $k = 4$ and $r = 3$.

Remark 2.5. It is crucial to distinguish between "reverse shadow" and "exact reverse shadow": Lemma 2.3 (and hence Lemma 2.2) does not remain true if we remove the word "exact." Consider the following example (see Figure 2.3). Let $a_1, \ldots, a_r$ be vertices such that there is an edge going from every a_i to every vertex of $T = \{t_1, t_2, \ldots, t_k\}$. For every $1 \leq i \leq r$, let b_i be a vertex with an edge going from b_i to a_i. For every $1 \leq i < j \leq r$, let $c_{i,j}$ be a vertex with two edges going from $c_{i,j}$ to a_i and a_j. Then every set $\{a_i, a_j\}$ is in $\mathcal{I}_k$, since it is an important $c_{i,j} - T$ separator; and every set $\{a_i\}$ is in $\mathcal{I}_k$ as well, as it is an important $b_i - T$ separator. Every b_i is in the reverse shadow of $\{a_j, a_i\}$ for $1 \leq i \neq j \leq r$. However, b_i is in the *exact* reverse shadow of exactly one member of $\mathcal{I}_k$, the set $\{a_i\}$.

Random sampling: In this subsection, we describe the technique of random sampling of important separators, which is crucial to the proof of Theorem 2.2. This technique was introduced in [MR11] and was adapted to directed graphs in [CHM12]. In Section 2.3.2, in order to reduce the problem (via the "torso" operation) to a shadowless instance, we need a set Z that has the following property:

Property (*)
There is an $\mathcal{F}$-transversal T^* such that Z covers the shadow of T^*, but Z is disjoint from T^*.

Of course, when we are trying to construct this set Z, we do not know anything about the $\mathcal{F}$-transversals of the instance. In particular we have no way of checking if a given set Z satisfies this property. Nevertheless, we use a randomized procedure that creates a set Z and we give a lower bound on the probability that Z satisfies the requirements. For the construction of this set Z, one can use a very specific probability distribution that was introduced in [MR11]. This probability distribution is based on randomly selecting "important separators" and taking the union of their shadows. We modify the selection of important separators in a way that improves the success probability. The precise description of the randomized procedure and the properties of the distribution it creates is described in Theorems 2.6 and 2.7. Using the theory of splitters we can derandomize the randomized selection into a deterministic algorithm that returns a bounded number of sets such that at least one of them satisfies the required property (see Section 2.1.2).

Roughly speaking, we want to select a random set Z such that for every (W,Y) where Y is in the reverse shadow of W, the probability that Z is disjoint from W but contains Y can be bounded from below. We can guarantee such a lower bound only if (W,Y) satisfies two conditions. First, it is not enough that Y is in the shadow of W (or in other words, W is an $Y-T$ separator), but W should contain important separators separating the vertices of Y from T (see Theorems 2.6 and 2.7 for the exact statement). Second, a vertex of W cannot be in the reverse shadow of other vertices of W, this is expressed by the following technical definition:

Definition 2.8. (thin) *Let G be a directed graph. We say that a set $W \subseteq V(G)$ is* thin *in G if there is no $v \in W$ such that v belongs to the reverse shadow of $W \setminus v$ with respect to T.*

We first give an easy version of the random sampling, which only gives a double exponentially small lower bound on the probability of constructing a set Z with the required properties.

Theorem 2.6. (random sampling) *There is an algorithm* RandomSet(G,T,k) *that produces a random set $Z \subseteq V(G) \setminus T$ in time $O^*(4^k)$ such that the following holds. Let W be a* thin *set with $|W| \leq k$, and let Y be a set such that for every $v \in Y$ there is an important $v-T$ separator $W' \subseteq W$. For every such pair (W,Y), the probability that the following two events both occur is $2^{-2^{O(k)}}$:*

1. *$W \cap Z = \emptyset$, and*
2. *$Y \subseteq Z$.*

Proof. The algorithm RandomSet(G,T,k) first enumerates the collection $\mathcal{I}_k$; let $\mathcal{X}$ be the set of all exact reverse shadows of these sets. By Lemma 2.1, the size of $\mathcal{X}$ is $O^*(4^k)$ and can be constructed in time $O^*(4^k)$. Let $\mathcal{X}'$ be the subset of $\mathcal{X}$ where each element from $\mathcal{X}$ occurs with probability $\frac{1}{2}$ independently at random. Let Z be the union of the exact reverse shadows in $\mathcal{X}'$. We claim that the set Z satisfies the requirement of the theorem.

Let us fix a pair (W,Y) as in the statement of the theorem. Let $X_1, X_2, \ldots, X_d \in \mathcal{X}$ be the exact reverse shadows of every member of $\mathcal{I}_k$ that is a subset of W. As $|W| \leq k$, we have $d \leq 2^k$. By the assumption that W is *thin*, we have $X_j \cap W = \emptyset$ for every $j \in [d]$. Now consider the following events:

(E1) $Z \cap W = \emptyset$

(E2) $X_j \subseteq Z$ for every $j \in [d]$

First we show that (E2) implies that $Y \subseteq Z$: $v \in Y$ implies there is an important separator $W' \subseteq W$, i.e., there is some $\ell \in [d]$ such that X_ℓ is the exact reverse shadow of W. Also note that $v \in X_\ell$ since W' is a minimal (in fact important) $v - T$ separator. Since $X_j \subseteq Z$ for every $j \in [d]$, we have that $v \in Z$. This shows that $Y \subseteq Z$.

Our goal is to show that both events (E1) and (E2) occur with probability $2^{-2^{O(k)}}$. Let $A = \{X_1, X_2, \ldots, X_d\}$ and $B = \{X \in \mathcal{X} \mid X \cap W \neq \emptyset\}$. By Lemma 2.2, each vertex of W is contained in the exact reverse shadows of at most 4^k members of $\mathcal{I}_k$. Thus $|B| \leq |W| \cdot 4^k \leq k \cdot 4^k$. If no exact reverse shadow from B is selected, then event (E1) holds. If every exact reverse shadow from A is selected, then event (E2) holds. Thus the probability that both (E1) and (E2) occur is bounded from below by the probability of the event that every element from A is selected and no element from B is selected. Note that A and B are disjoint: A contains only sets disjoint from W, while B contains only sets intersecting W. Therefore, the two events are independent and the probability that both events occur is at least

$$\left(\frac{1}{2}\right)^{2^k} \left(1 - \frac{1}{2}\right)^{k \cdot 4^k} = 2^{-2^{O(k)}}$$

□

We now give an improved version of the random sampling that gives a stronger lower bound on the success probability than the one guaranteed by Theorem 2.6. Recall that in Theorem 2.6, we randomly selected members of $\mathcal{I}_k$ and took Z as the union of the exact reverse shadows of the selected sets. However, we only had single-exponential upper bounds on both types of exact reverse shadows: number of shadows intersecting W was at most $k \cdot 4^k$ and the number of exact reverse shadows of every subset of W is at most 2^k. In Theorem 2.7, we take a different view: randomly select a subset of vertices $\mathcal{P}$ and take Z as the union of exact reverse shadows of every subset of $\mathcal{P}$. This will give us a stronger (single exponentially small) lower bound on the probability that the constructed set Z satisfies the required properties.

Theorem 2.7. (improved random sampling) *There is an algorithm* RandomSet(G,T,k) *that produces a random set $Z \subseteq V(G) \setminus T$ in time $O^*(4^k)$ such that the following holds. Let W be a* thin *set with $|W| \leq k$, and let Y be a set such that for every $v \in Y$ there is an important $v-T$ separator $W' \subseteq W$. For every such pair (W,Y), the probability that the following two events both occur is $2^{-O(k^2)}$:*

1. *$W \cap Z = \emptyset$, and*
2. *$Y \subseteq Z$.*

Proof. For each $w \in W$, we define

$$\mathcal{L}_w = \{S \mid S \text{ is an important } w-T \text{ separator of size } \leq k\},$$

$$I_w = \bigcup_{S \in \mathcal{L}_w} S,$$

$$I = \bigcup_{w \in W} I_w.$$

Since $|W| \leq k$ and for each $w \in W$ there are at most 4^k important $w-T$ separators of size at most k, we have $|I_w| \leq k \cdot 4^k$. Since $|W| \leq k$, we have $|I| \leq k^2 \cdot 4^k$. The algorithm

RandomSet(G,T,k) picks a subset P of $V(G)$ where each element occurs with probability 4^{-k} uniformly at random. For every $S \in \mathcal{I}_k$ with $S \subseteq P$, let us add the exact reverse shadow of S to $\mathcal{X}'$. Let Z be the union of the exact reverse shadows in $\mathcal{X}'$. We claim that the set Z satisfies the requirement of the theorem.

Fix a pair (W,Y) as in the statement of the theorem. Let $\mathcal{X}$ be the set of exact reverse shadows of every set $S \in \mathcal{I}_k$. Let $X_1, X_2, \ldots, X_d \in \mathcal{X}$ be the exact reverse shadows of every $S \in \mathcal{I}_k$ with $S \subseteq W$. Let $A = \{X_1, X_2, \ldots, X_d\}$ and $B = \{X \in \mathcal{X} \mid X \cap W \neq \emptyset\}$. Now consider the following events:

(E1) $Z \cap W = \emptyset$

(E2) $X_j \subseteq Z$ for every $j \in [d]$

First we show that (E2) implies that $Y \subseteq Z$: $v \in Y$ implies there is an important separator $W' \subseteq W$, i.e., there is some $\ell \in [d]$ such that X_ℓ is the exact reverse shadow of W. Also note that $v \in X_\ell$ since W' is a minimal (in fact important) $v - T$ separator. Since $X_j \subseteq Z$ for every $j \in [d]$, we have that $v \in Z$. This shows that $Y \subseteq Z$.

Our goal is to show that both events (E1) and (E2) occur with probability $2^{-O(k^2)}$. If every vertex from W is selected in P, then every reverse shadow from A is selected in $\mathcal{X}'$ and event (E2) holds. If no vertex from $I \setminus W$ is selected in P, then we claim that no exact reverse shadow from B is selected in $\mathcal{X}'$ and hence event $(E1)$ will also hold. Suppose to the contrary that an exact reverse shadow $X \in B$ was selected in $\mathcal{X}'$. Let $J \in \mathcal{I}_k$ be the set whose exact reverse shadow is X. We now claim that $J \nsubseteq W$, which contradicts the fact that no vertex of $I \setminus W$ was selected in P. Suppose $J \subseteq W$. Since X is the exact reverse shadow of J, we know that J is a minimal $X-T$ separator. But $J \subseteq W$ implies that $W \setminus X$ is also a $X-T$ separator, i.e., $W \cap X$ lies in the reverse shadow of $W \setminus (W \cap X)$. This contradicts the fact that W is a thin set (see Definition 2.8).

Thus the probability that both the events (E1) and (E2) occur is bounded from below by the probability of the event that every vertex from W is selected in P and no vertex from $I \setminus W$ is selected in P. Note that the sets W and $I \setminus W$ are clearly disjoint. Therefore, the two events are independent and the probability that both events occur is at least

$$(4^{-k})^k(1-4^{-k})^{k^2 \cdot 4^k} \geq 4^{-k^2} \cdot e^{-2k^2} = 2^{-O(k^2)}$$

where we used the inequalities that $1+x \geq e^{\frac{x}{1+x}}$ for every $x > -1$ and $1-4^{-k} \geq \frac{1}{2}$ for every $k \geq 1$. □

2.1.2 Derandomization

We now derandomize the process of choosing exact reverse shadows in Theorem 2.7 using the technique of *splitters*. An (n,r,r^2)-splitter is a family of functions from $[n] \rightarrow [r^2]$ such that for every $M \subseteq [n]$ with $|M| = r$, at least one of the functions in the family is injective on M. Naor et al. [NSS95] give an explicit construction of an (n,r,r^2)-splitter of size $O(r^6 \log r \log n)$ in time poly(n,r).

Theorem 2.8. (deterministic sampling) *There is an algorithm* DeterministicSets(G,T,k) *that produces $t = 2^{O(k^2)} \log |V(G)|$ subsets of $Z_1, \ldots, Z_t$ of $V(G) \setminus T$ in time $O^*(2^{O(k^2)})$ such that the following holds. Let W be a* thin *set with $|W| \leq k$, and let Y be a set such* *that for every $v \in Y$ there is an important $v - T$ separator $W' \subseteq W$. For every such pair (W,Y), there is at least one $1 \leq i \leq t$ with*

1. *$W \cap Z = \emptyset$, and*
2. *$Y \subseteq Z$.*

Proof. In the proof of Theorem 2.7, a random subset P of a universe $V(G)$ of size n is selected. We argued that if every vertex from W is selected in P and no element from $I \setminus W$ is selected, then both the events (E1) and (E2) occur. Instead of selecting a random subset P, we will construct several subsets such that at least one of them will contain only vertices from W. Let $n = |V(G)|$, $a = |W| \leq k$, and $b = |I \setminus W| \leq k^2 \cdot 4^k$. Each subset is defined by a pair (h,H), where h is a function in an $(n, a+b, (a+b)^2)$-splitter family and H is a subset of $[(a+b)^2]$ of size a (there are $\binom{(a+b)^2}{a} = \binom{(k+k^2 \cdot 4^k)^2}{k} = 2^{O(k^2)}$ such sets H). For a particular choice of h and H, we select those vertices $v \in V(G)$ into P for which $h(v) \in H$. The size of the splitter family is $O\Big((a+b)^6 \log(a+b)\log(n)\Big) = 2^{O(k)} \log n$ and the number of possibilities for H is $2^{O(k^2)}$. Therefore, we construct $2^{O(k^2)} \log n$ subsets of $V(G)$. The total time taken for constructing these subsets is $\text{poly}(n, a+b) = \text{poly}(n, 4^k)$.

By the definition of the splitter, there is a function h that is injective on W, and there is a subset H such that $h(v) \in H$ for every set $v \in W$ and $h(y) \notin H$ for every $y \in I \setminus W$. For such an h and H, the selection will ensure that (E1) and (E2) hold. Thus at least one of the constructed subsets has the required properties, which is what we had to show. $\square$

Algorithm 1: COVERING (randomized version)

```
Input: A directed graph $G_1$, integer $k$.
Output: A set $Z$.
1: Let $Z_1 = \text{RandomSet}(G_1, T, k)$.
2: Let $G_2$ be obtained from $G_1$ by reversing the orientation of every edge and
   setting the weight of every vertex of $Z_1$ to infinity.
3: Let $Z_2 = \text{RandomSet}(G_2, T, k)$.
4: Let $Z = Z_1 \cup Z_2$.
```

2.1.3 Proof of Theorem 2.2: The COVERING Algorithm

To prove Theorem 2.2, we show that Algorithm 1 gives a set Z satisfying the properties of Theorem 2.2. Due to the delicate way separators behave in directed graphs, we construct the set Z in two phases, calling the function RandomSet of Section 2.1.1 twice. For consistency of notation, we denote the input graph by G_1. Our aim is to show that there is a transversal T^* such that we can give a lower bound on the probability that Z_1 covers $r_{G_1,T}(T^*)$ and Z_2 covers $f_{G_1,T}(T^*)$. Note that the graph G_2 obtained in Step 2 depends on the set Z_1 returned in Step 1 (as we made the weight of every vertex in Z_1 infinite), thus the distribution of the second random sampling depends on the result Z_1 of the first random sampling. This means that we cannot make the two calls in parallel.

To prove the existence of the required transversal T^*, we need the following definition:

Definition 2.9. (shadow-maximal transversal) *An $\mathcal{F}$-transversal W is* minimum *if there is no $\mathcal{F}$-transversal of size less than $|W|$. A minimum $\mathcal{F}$-transversal W is called* shadow-maximal *if $r_{G_1,T}(W) \cup f_{G_1,T}(W) \cup W$ is inclusion-wise maximal among all minimum $\mathcal{F}$-*

transversals.

For the rest of the proof, let us fix T^* to be a shadow-maximal $\mathcal{F}$-transversal such that $|r_{G_1,T}(T^*)|$ is maximum possible among all shadow-maximal $\mathcal{F}$-transversals. We bound the probability that $Z \cap T^* = \emptyset$ and $r_{G_1,T}(T^*) \cup f_{G_1,T}(T^*) \subseteq Z$. More precisely, we bound the probability that all of the following four events occur:

1. $Z_1 \cap T^* = \emptyset$,
2. $r_{G_1,T}(T^*) \subseteq Z_1$,
3. $Z_2 \cap T^* = \emptyset$, and
4. $f_{G_1,T}(T^*) \subseteq Z_2$.

That is, the first random selection takes care of the reverse shadow, the second takes care of the forward shadow, and none of Z_1 or Z_2 hits T^*. Note that it is somewhat counterintuitive that we choose an T^* for which the shadow is large: intuitively, it seems that the larger the shadow is, the less likely that it is fully covered by Z. However, we need this maximality property in order to bound the probability that $Z \cap T^* = \emptyset$.

We want to invoke Theorem 2.7 to bound the probability that Z_1 covers $Y = r_{G_1,T}(T^*)$ and $Z_1 \cap T^* = \emptyset$. First, we need to ensure that T^* is a *thin* set, but this follows easily from the fact that T^* is a minimum $\mathcal{F}$-transversal:

Lemma 2.4. *If W is a minimum $\mathcal{F}$-transversal for some T-connected $\mathcal{F}$, then no $v \in W$ is in the reverse shadow of some $W' \subseteq W \setminus \{v\}$.*

Proof. Suppose to the contrary that there is a vertex $v \in W$ such that $v \in r(W')$ for some

$W' \subseteq W \setminus \{v\}$. Then we claim that $W \setminus \{v\}$ is also an $\mathcal{F}$-transversal, contradicting the minimality of W. Let $\mathcal{F} = \{F_1, F_2, \ldots, F_q\}$ and suppose that there is a $i \in [q]$ such that $F_i \cap W = \{v\}$. As $\mathcal{F}$ is T-connected, there is a $v \to T$ walk P in F_i. But $P \cap W = \{v\}$ implies that there is a $v \to T$ walk in $G \setminus (W \setminus \{v\})$, i.e., v cannot belong to the reverse shadow of any $W' \subseteq W \setminus \{v\}$. □

More importantly, if we want to use Theorem 2.7 with $Y = r_{G_1,T}(T^*)$, then we have to make sure that for every vertex v of $r_{G_1,T}(T^*)$, there is an important $v - T$ separator that is a subset of T^*. The "pushing argument" of Lemma 2.5 shows that if this is not true for some v, then we can modify the $\mathcal{F}$-transversal in a way that increases the size of the reverse shadow. The extremal choice of T^* ensures that no such modification is possible, thus T^* contains an important $v - T$ separator for every v.

Lemma 2.5. **(pushing)** *Let W be an $\mathcal{F}$-transversal for some T-connected $\mathcal{F}$. For every $v \in r(W)$, either there is an $W_1 \subseteq W$ that is an important $v - T$ separator, or there is an $\mathcal{F}$-transversal W' such that*

1. $|W'| \le |W|$,
2. $r(W) \subset r(W')$,
3. $(r(W) \cup f(W) \cup W) \subseteq (r(W') \cup f(W') \cup W')$.

Proof. Let W_0 be the subset of W reachable from v without going through any other vertices of W. Then W_0 is clearly a $v - T$ separator. Let W_1 be the minimal $v - T$ separator contained in W_0. If W_1 is an important $v - T$ separator, then we are done as W itself contains W_1. Otherwise, there exists an important $v-T$ separator W_1', i.e., $|W_1'| \leq |W_1|$ and $R^+_{G\setminus W_1}(v) \subset R^+_{G\setminus W_1'}(v)$. Now we show that $W' = (W \setminus W_1) \cup W_1'$ is also an $\mathcal{F}$-transversal. Note that $W_1' \subseteq W'$ and $|W'| \leq |W|$.

First we claim that $r(W) \cup (W \setminus W') \subseteq r(W')$. Suppose that there is a walk P from β to T in $G \setminus W'$ for some $\beta \in r(W) \cup (W \setminus W')$. If $\beta \in r(W)$, then walk P has to go through a vertex $\beta' \in W$. As β' is not in W', it has to be in $W \setminus W'$. Therefore, by replacing β with β', we can assume in the following that $\beta \in W \setminus W' \subseteq W_1 \setminus W_1'$. By the minimality of W_1, every vertex of $W_1 \subseteq W_0$ has an incoming edge from some vertex in $R^+_{G\setminus W}(v)$. This means that there is a vertex $\alpha \in R^+_{G\setminus W}(v)$ such that $(\alpha, \beta) \in E(G)$. Since $R^+_{G\setminus W}(v) \subset R^+_{G\setminus W'}(v)$, we have $\alpha \in R^+_{G\setminus W'}(v)$, implying that there is a $v \to \alpha$ walk in $G \setminus W'$. The edge $\alpha \to \beta$ also survives in $G \setminus W'$ as $\alpha \in R^+_{G\setminus W'}(v)$ and $\beta \in W_1 \setminus W_1'$. By assumption, we have a walk in $G \setminus W'$ from β to some $t \in T$. Concatenating the three walks we obtain a $v \to t$ walk in $G \setminus W'$, which contradicts the fact that W' contains an (important) $v-T$ separator W_1'. This proves the claim. Since $W \neq W'$ and $|W| = |W'|$, the set $W_1 \setminus W_1'$ is non-empty. Thus $r(W) \subset r(W')$ follows from the claim $r(W) \cup (W \setminus W') \subseteq r(W')$.

Suppose now that W' is not an $\mathcal{F}$-transversal. Then there is some $i \in [q]$ such that $F_i \cap W' = \emptyset$. As W is an $\mathcal{F}$-transversal, there is some $w \in W \setminus W'$ with $w \in F_i$. As $\mathcal{F}$ is T-connected, there is a $w \to T$ walk in F_i, which gives a $w \to T$ walk in $G \setminus W'$ as $W' \cap F_i = \emptyset$. However, we have $W \setminus W' \subseteq r(W')$ (by the claim in the previous paragraph), a contradiction. Thus W' is also an $\mathcal{F}$-transversal.

Finally, we show that $r(W) \cup f(W) \cup W \subseteq r(W') \cup f(W') \cup W'$. We know that

$r(W) \cup (W \setminus W') \subseteq r(W')$. Thus it is sufficient to consider a vertex $v \in f(W) \setminus r(W)$. Suppose that $v \notin f(W')$ and $v \notin r(W')$: there are walks P_1 and P_2 in $G \setminus W'$, going from T to v and from v to T, respectively. As $v \in f(W)$, walk P_1 intersects W, i.e., it goes through a vertex of $\beta \in W \setminus W' \subseteq r(W')$. However, concatenating the subwalk of P_1 from β to v and the walk P_2 gives a walk from $\beta \in r(W')$ to T in $G \setminus W'$, a contradiction. □

Note that if W is a shadow-maximal $\mathcal{F}$-transversal, then the $\mathcal{F}$-transversal W' in Lemma 2.5 is also a minimum $\mathcal{F}$-transversal and shadow-maximal. Therefore, by the extremal choice of T^*, applying Lemma 2.5 on T^* cannot produce a shadow-maximal $\mathcal{F}$-transversal T' with $r_{G_1,T}(T^*) \subset r_{G_1,T}(T')$, and hence T^* contains an important $v - T$ separator for every $v \in r_{G_1,T}(T^*)$. Thus by Theorem 2.7 for $Y = r_{G_1,T}(T^*)$, we get:

Lemma 2.6. *With probability at least $2^{-O(k^2)}$, both $r_{G_1,T}(T^*) \subseteq Z_1$ and $Z_1 \cap T^* = \emptyset$ occur.*

In the following, we assume that the events in Lemma 2.6 occur. Our next goal is to bound the probability that Z_2 covers $f_{G_1,T}(T^*)$. Let us define a collection $\mathcal{F}'$ of subgraphs of G_2 as follows: for every subgraph $F \in \mathcal{F}$ of G_1, let us add to $\mathcal{F}'$ the corresponding subgraph F' of G_2, i.e., F' is the same as F with every edge reversed. Note that $\mathcal{F}'$ is T-connected in G_2: the definition of T-connected is symmetric with respect to the orientation of the edges. Moreover, T^* is an $\mathcal{F}'$-transversal in G_2: the vertices in T^* remained finite (as $Z_1 \cap T^* = \emptyset$ by Lemma 2.6), and reversing the orientation of the edges does not change the fact that T^* is a transversal. Set T^* is also shadow-maximal as an $\mathcal{F}'$-transversal in G_2: Definition 2.9 is insensitive to reversing the orientation of the edges and making some of the weights infinite can only decrease the set of potential transversals.

Furthermore, the forward shadow of T^* in G_2 is same as the reverse shadow of T^* in G_1, that is, $f_{G_2,T}(T^*) = r_{G_1,T}(T^*)$. Therefore, assuming that the events in Lemma 2.6 occur, every vertex of $f_{G_2,T}(T^*)$ has infinite weight in G_2. We show that now it holds that T^* contains an important $v-T$ separator in G_2 for every $v \in r_{G_2,T}(T^*) = f_{G_1,T}(T^*)$:

Lemma 2.7. *If W is a shadow-maximal $\mathcal{F}$-transversal for some T-connected $\mathcal{F}$ and every vertex of $f(W)$ has infinite weight, then W contains an important $v-T$ separator for every $v \in r(W)$.*

Proof. Suppose to the contrary that there exists $v \in r(W)$ such that W does not contain an important $v-T$ separator. Then by Lemma 2.5, there is a another shadow-maximal $\mathcal{F}$-transversal W'. As W is shadow-maximal, it follows that $r(W) \cup f(W) \cup W = r(W') \cup f(W') \cup W'$. Therefore, the nonempty set $W' \setminus W$ is fully contained in $r(W) \cup f(W) \cup W$. However, it cannot contain any vertex of $f(W)$ (as they are infinite by assumption) and cannot contain any vertex of $r(W)$ (as $r(W) \subset r(W')$), a contradiction. $\square$

Recall that T^* is a shadow-maximal $\mathcal{F}'$-transversal in G_2. In particular, T^* is a minimal $\mathcal{F}'$-transversal in G_2, hence Lemma 2.4 implies that T^* is thin in G_2 also. Thus Theorem 2.7 can be used again (this time with $Y = r_{G_2,T}(T^*)$) to bound the probability that $r_{G_2,T}(T^*) \subseteq Z_2$ and $Z_2 \cap T^* = \emptyset$. As the reverse shadow $r_{G_2,T}(T^*)$ in G_2 is the same as the forward shadow $f_{G_1,T}(T^*)$ in G, we can state the following:

Lemma 2.8. *Assuming the events in Lemma 2.6 occur, with probability at least $2^{-O(k^2)}$ both $f_{G_1,T}(T^*) \subseteq Z_2$ and $Z_2 \cap T^* = \emptyset$ occur.*

Therefore, with probability $(2^{-O(k^2)})^2$, the set $Z_1 \cup Z_2$ covers $f_{G_1,T}(T^*) \cup r_{G_1,T}(T^*)$ and it is disjoint from T^*. This completes the proof of Theorem 2.2.

Finally, to prove Theorem 2.3, the derandomized version of Theorem 2.3, we use the deterministic variant DeterministicSets(G,T,k) of the function RandomSet(G,T,k) that, instead of returning a random set Z, returns a deterministic set $Z_1, \ldots, Z_t$ of $t = 2^{O(k^2)} \log n$ sets in poly$(n, 4^k)$ time (Theorem 2.8). Therefore, in Steps 1 and 3 of Algorithm 1, we can replace RandomSet with this deterministic variant DeterministicSets, and branch on the choice of one Z_i from the returned sets. By the properties of the deterministic algorithm, if I is a yes-instance, then Z has Property (*) in at least one of the $2^{O(k^2)} \log^2 n$ branches. The branching increases the running time only by a factor of $(O^*(2^{O(k^2)}))^2$ and therefore the total running time is $O^*(2^{O(k^2)})$. This completes the proof of Theorem 2.3.

2.1.4 Upper Bound on the Number of Important Separators

For the proof of Lemma 2.1, we need to establish first some simple properties of important separators, which will allow us to use recursion.

Lemma 2.9. *Let G be a directed graph and S be an important $X-Y$ separator. Then*

1. *For every $v \in S$, the set $S \setminus v$ is an important $X-Y$ separator in the graph $G \setminus v$.*
2. *If S is an $X'-Y$ separator for some $X' \supset X$, then S is also an important $X'-Y$ separator.*

Proof.

1. Suppose $S \setminus v$ is not a minimal $X-Y$ separator in $G \setminus v$. Let $S_0 \subset S \setminus v$ be an $X-Y$ separator in $G \setminus v$. Then $S_0 \cup v$ is an $X-Y$ separator in G, but $S_0 \cup v \subset S$ holds, which contradicts the fact that S is a minimal $X-Y$ separator in G. Now suppose that there exists an $S' \subseteq V(G) \setminus v$ such that $|S'| \leq |S \setminus v| = |S| - 1$ and $R^+_{(G\setminus v)\setminus(S\setminus v)}(X) \subset R^+_{(G\setminus v)\setminus S'}(X)$. Noting that $(G \setminus v) \setminus (S \setminus v) = G \setminus S$ and $(G \setminus v) \setminus S' = G \setminus (S' \cup v)$, we get $R^+_{G\setminus S}(X) \subset R^+_{G\setminus(S'\cup v)}(X)$. As $|S' \cup v| = |S'| + 1 \leq |S|$, this contradicts the fact that S is an important $X-Y$ separator.

2. As S is an inclusionwise minimal $X-Y$ separator, it is an inclusionwise minimal $X'-Y$ separator as well. Let S' be a witness that S is not an important $X'-Y$ separator in G, i.e., S' is an $X'-Y$ separator such that $|S'| \leq |S|$ and $R^+_{G\setminus S}(X') \subset R^+_{G\setminus S'}(X')$. We claim first that $R^+_{G\setminus S}(X) \subseteq R^+_{G\setminus S'}(X)$. Indeed, if P is any path from X and fully contained in $R^+_{G\setminus S}(X)$, then P is disjoint from S', otherwise vertices of $P \cap S'$ are in $R^+_{G\setminus S}(X')$, but not in $R^+_{G\setminus S'}(X')$, a contradiction. Next we show that the inclusion $R^+_{G\setminus S}(X) \subset R^+_{G\setminus S'}(X)$ is proper, contradicting that S is an important $X-Y$ separator. As $|S'| \leq |S|$, there is a vertex $v \in S \setminus S'$. Since S is a minimal $X-Y$ separator, it has an in-neighbor $u \in R^+_{G\setminus S}(X) \subseteq R^+_{G\setminus S'}(X)$. Now $v \in S$ and $v \notin S'$ imply that $v \in R^+_{G\setminus S'}(X) \setminus R^+_{G\setminus S}(X)$, a contradiction.

□

Next we show that the size of the out-neighborhood of a vertex set is a submodular function. Recall that a function $f : 2^U \to \mathbb{N} \cup \{0\}$ is *submodular* if for all $A, B \subseteq U$ we have $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.

Lemma 2.10. **(submodularity)** *The function* $\gamma(A) = |N^+(A)|$ *is submodular.*

Proof. Let $L = \gamma(A) + \gamma(B)$ and $R = \gamma(A \cup B) + \gamma(A \cap B)$. To prove $L \geq R$ we show that for each vertex $x \in V$ its contribution to L is at least as much as its contribution to R. Suppose that the weight of x is w (in our setting, w is either 1 or ∞, but submodularity holds even if the weights are arbitrary). The contribution of x to L or R is either 0, w, or $2w$. We have the following four cases:

1. $x \notin N^+(A)$ and $x \notin N^+(B)$.

   In this case, x contributes 0 to L. It contributes 0 to R as well: every vertex in $N^+(A \cap B)$ or in $N^+(A \cup B)$ is either in $N^+(A)$ or in $N^+(B)$.

2. $x \in N^+(A)$ and $x \notin N^+(B)$.

   In this case, x contributes w to L. To see that x does not contribute $2w$ to R, suppose that $x \in N^+(A \cup B)$ holds. This implies $x \notin A \cup B$ and therefore $x \in N^+(A \cap B)$ can be true only if $x \in N^+(A)$ and $x \in N^+(B)$, which is a contradiction. Therefore, x contributes only w to R.

3. $x \notin N^+(A)$ and $x \in N^+(B)$.

   Symmetric to the previous case.

4. $x \in N^+(A)$ and $x \in N^+(B)$

   In this case, x contributes $2w$ to L, and can anyways contribute at most $2w$ to R.

In all four cases the contribution of x to L is always greater than or equal to its contribution to R and hence $L \geq R$, i.e., γ is submodular. $\square$

Recall that $R^+_{G\setminus S}(X)$ is the set of vertices reachable from X in $G\setminus S$. The following claim will be useful for the use of submodularity:

Lemma 2.11. *Let G be a directed graph. If S_1, S_2 are $X-Y$ separators, then both the sets $N^+(R^+_{G\setminus S_1}(X)\bigcup R^+_{G\setminus S_2}(X))$ and $N^+(R^+_{G\setminus S_1}(X)\bigcap R^+_{G\setminus S_2}(X))$ are also $X-Y$ separators.*

Proof. 1. Let $R_\cap = R^+_{G\setminus S_1}(X)\bigcap R^+_{G\setminus S_2}(X)$ and $S_\cap = N^+(R_\cap)$. As S_1 and S_2 are disjoint from X and Y by definition, we have $X\subseteq R_\cap$ and Y is disjoint from $R_\cap$. Therefore, every path P from X to Y has a vertex $u\in R_\cap$ followed by a vertex $v\notin R_\cap$, and therefore $v\in S_\cap$. As this holds for every path P, the set $S_\cap$ is an $X-Y$ separator.

2. The argument is the same with the sets $R_\cup = R^+_{G\setminus S_1}(X)\bigcup R^+_{G\setminus S_2}(X)$ and $S_\cup = N^+(R_\cup)$. □

Now we prove the well-known fact that there is a unique minimum size separator whose "reach" is inclusion-wise maximal.

Lemma 2.12. *There is a unique $X-Y$ separator S^* of minimum size such that $R^+_{G\setminus S^*}(X)$ is inclusion-wise maximal.*

Proof. Let λ be the size of a smallest $X-Y$ separator. Suppose to the contrary that there are two separators S_1 and S_2 of size λ such that $R^+_{G\setminus S_1}(X)$ and $R^+_{G\setminus S_2}(X)$ are incomparable and inclusion-wise maximal. Let $R_1 = R^+_{G\setminus S_1}(X)$, $R_2 = R^+_{G\setminus S_2}(X)$, $R_\cap = R_1\cap R_2$, and $R_\cup = R_1\cup R_2$. By Lemma 2.10, γ is submodular and hence

$$\gamma(R_1)+\gamma(R_2)\geq \gamma(R_\cup)+\gamma(R_\cap). \tag{2.1}$$

As $N^+(R_1) \subseteq S_1$ and $N^+(R_2) \subseteq S_2$, the left hand side is at most 2λ (in fact, as S_1 and S_2 are minimal $X-Y$ separators, it can be seen that the left hand side is exactly 2λ). By Lemma 2.11, both the sets $N^+(R_\cap)$ and $N^+(R_\cup)$ are $X-Y$ separators. Therefore, the right hand side is at least 2λ. This implies that equality holds in Equation 2.1 and in particular $|N^+(R_\cup)| = \lambda$, i.e., $N^+(R_\cup)$ is also a minimum $X-Y$ separator. As $R_1, R_2 \subseteq R_\cup$, every vertex of R_1 and every vertex of R_2 is reachable from X in $G \setminus N^+(R_\cup)$. This contradicts the inclusion-wise maximality of the reach of S_1 and S_2. □

Let S^* be the unique $X-Y$ separator of minimum size given by Lemma 2.12. The following lemma shows that every important $X-Y$ separator S is "behind" this separator S^*:

Lemma 2.13. *Let S^* be the unique $X-Y$ separator of minimum size given by Lemma 2.12. For every important $X-Y$ separator S, we have $R^+_{G\setminus S^*}(X) \subseteq R^+_{G\setminus S}(X)$.*

Proof. Note that the condition trivially holds for $S = S^*$. Lemma 2.12 implies that the only important $X-Y$ separator of minimum size is S^*.

Suppose there is an important $X-Y$ separator $S \neq S^*$ such that $R^+_{G\setminus S^*}(X) \not\subseteq R^+_{G\setminus S}(X)$. Let $R = R^+_{G\setminus S}(X)$, $R^* = R^+_{G\setminus S^*}(X)$, $R_\cap = R \cap R^*$, and $R_\cup = R \cup R^*$. By Lemma 2.10, γ is submodular and hence

$$\gamma(R^*) + \gamma(R) \geq \gamma(R_\cup) + \gamma(R_\cap). \tag{2.2}$$

As $N^+(R^*) \subseteq S^*$, we have that the first term on the left hand side is at most $|S^*| = \lambda$. By Lemma 2.11, the set $N^+(R_\cap)$ is an $X-Y$ separator, hence the second term on the right hand side is at least λ. It follows that $|N^+(R_\cup)| \leq |N^+(R)(X))| \leq |S|$. Since $R^* \not\subseteq R$

by assumption, we have $R \subset R_{\cup}$. By Lemma 2.11, $N^{+}(R_{\cup})$ is also an $X - Y$ separator and we have seen that it has size at most $|S|$. Furthermore, $R \subset R_{\cup}$ implies that any vertex reachable from X in $G \setminus S$ is reachable in $G \setminus N^{+}(R_{\cup})$ as well, contradicting the assumption that S is an important separator. $\square$

Now we finally have all the required tools to prove Lemma 2.1, which we restate below.

Lemma 2.1 . *Let $X, Y \subseteq V(G)$ be disjoint sets in a directed graph G. Then for every $k \geq 0$ there are at most 4^k important $X - Y$ separators of size at most k. Furthermore, we can enumerate all these separators in time $O(4^k \cdot k(|V(G) + |E(G)|))$.*

Proof. Let λ be the size of a smallest $X - Y$ separator. We show by induction on $2p - \lambda$ that the number of important $X - Y$ separators of size at most p is upper bounded by $2^{2p-\lambda}$. Note that if $2p - \lambda < 0$, then $\lambda > 2p \geq p$ and so there is no (important) $X - Y$ separator of size at most p. If $2p - \lambda = 0$, then $\lambda = 2p$. Now if $p = 0$ then $\lambda = p = 0$ and the empty set is the unique important $X - Y$ separator of size at most p. If $p > 0$, then $\lambda = 2p > p$ and hence there is no important $X - Y$ separator of size at most p. Thus we have checked the base case for induction. From now on, the induction hypothesis states that if $X', Y' \subseteq V(G)$ are disjoint sets such that λ' is the size of a smallest $X' - Y'$ separator and p' is an integer such that $(2p' - \lambda') < (2p - \lambda)$, then the number of important $X' - Y'$ separators of size at most p' is upper bounded by $2^{2p'-\lambda'}$.

Let S^* be the unique $X - Y$ separator of minimum size given by Lemma 2.12. Consider an arbitrary vertex $v \in S^*$. Note that $\lambda > 0$ and so S^* is not empty. Any important

$X-Y$ separator S of size at most p either contains v or not. If S contains v, then by Lemma 2.9(1), the set $S\setminus\{v\}$ is an important $X-Y$ separator in $G\setminus v$ of size at most $p':=p-1$. As $v\notin X\cup Y\cup V^{\infty}$, the size λ' of the minimum $X-Y$ separator in $G\setminus v$ is at least $\lambda-1$. Therefore, $2p'-\lambda'=2(p-1)-\lambda'=2p-(\lambda'+2)<2p-\lambda$. The induction hypothesis implies that there are at most $2^{2p'-\lambda'}\leq 2^{2p-\lambda-1}$ important $X-Y$ separators of size p' in $G\setminus v$. Hence there are at most $2^{2p-\lambda-1}$ important $X-Y$ separators of size at most p in G that contain v.

Now we give an upper bound on the number of important $X-Y$ separators *not* containing v. By minimality of S^*, vertex v has an in-neighbor in $R^+_{G\setminus S^*}(X)$. For every important $X-Y$ separator S, Lemma 2.13 implies $R^+_{G\setminus S^*}(X)\subseteq R^+_{G\setminus S}(X)$. As $v\notin S$ and v has an in-neighbor in $R^+_{G\setminus S^*}(X)$, even $R^+_{G\setminus S^*}(X)\bigcup\{v\}\subseteq R^+_{G\setminus S}(X)$ holds. Therefore, setting $X'=R^+_{G\setminus S^*}(X)\bigcup\{v\}$, the set S is also an $X'-Y$ separator. Now Lemma 2.9(2) implies that S is in fact an important $X'-Y$ separator. Since S is an $X-Y$ separator, we have $|S|\geq\lambda$. We claim that in fact $|S|>\lambda$: otherwise $|S|=|S^*|=\lambda$ and $R^+_{G\setminus S^*}(X)\bigcup\{v\}\subseteq R^+_{G\setminus S}(X)$, contradicting the fact that S^* is an important $X-Y$ separator. So the minimum size λ' of an $X'-Y$ separator in G is at least $\lambda+1$. By the induction hypothesis, the number of important $X'-Y$ separators of size at most p in G is at most $2^{2p-\lambda'}\leq 2^{2p-\lambda-1}$. Hence there are at most $2^{2p-\lambda-1}$ important $X-Y$ separators of size at most p in G that do not contain v.

Adding the bounds in the two cases, we get the required upper bound of $2^{2p-\lambda}$. An algorithm for enumerating all the at most 4^p important separators follows from the above proof. First, we can find a maximum $X-Y$ flow in time $O(p(|V(G)|+|E(G)|))$ using at most p rounds of the Ford-Fulkerson algorithm, where n and m are the number of vertices and edges of G. It is well-known that the separator S^* of Lemma 2.12 can be deduced from the maximum flow in linear time by finding those vertices from which Y cannot be reached in the residual graph [FF62]. Pick any arbitrary vertex $v \in S^*$. Then we branch on whether vertex $v \in S^*$ is in the important separator or not, and recursively find all possible important separators for both cases. The formal description is given in Algorithm 2.

Algorithm 2: $\textsc{ImpSep}(G, X, Y, p)$

Input: A directed graph G, disjoint sets $X, Y \subseteq V$ and an integer p.
Output: A collection of $X-Y$ separators that is a superset of all important $X-Y$ separators of size at most p in G.

```
1:  Find the minimum $X-Y$ separator $S^*$ of Lemma 2.12
2:  Let $\lambda = |S'|$
3:  **if** $p < \lambda$ **then**
4:      **return** $\emptyset$
5:  **else**
6:      Pick any arbitrary vertex $v \in S^*$
7:      Let $\mathcal{S}_1 = \textsc{ImpSep}(G \setminus \{v\}, X, Y, p-1)$
8:      Let $\mathcal{S}_1' = \{v \cup S \mid S \in \mathcal{S}_1\}$
9:      Let $X' = R^+_{G \setminus S^*}(X) \cup \{v\}$
10:     Let $\mathcal{S}_2 = \textsc{ImpSep}(G, X', Y, p)$
11:     **return** $\mathcal{S}_1' \cup \mathcal{S}_2$
```

Note that this algorithm enumerates a superset of all important separators: by our analysis above, every important separator appears in either S_1' or S_2, but there is no guarantee that all the separators in these sets are important. Therefore, the algorithm has to be followed by a filtering phase where we check for each returned separator whether it is important. Observe that S is an important $X-Y$ separator if and only if S is the unique minimum $R^+_{G\setminus S}(X) - Y$ separator. As the size of S is at most p, this can be checked in time $O(p(|V(G)|+|E(G)|))$ by finding a maximum flow and constructing the residual graph. The search tree has at most 4^p leaves and the work to be done in each node is $O(p(|V(G)|+|E(G)|))$. Therefore, the total running time of the branching algorithms is $O(4^p \cdot p(|V(G)|+|E(G)|))$ and returns at most 4^p separators. This is followed by the filtering phase, which takes time $O(4^p \cdot p(|V(G)|+|E(G)|))$.

□

2.2 Application I: FPT Algorithm for DIRECTED MULTIWAY CUT

Ford and Fulkerson [FF56] gave the classical result on finding a minimum cut that separates two terminals s and t. A natural and well-studied generalization of the minimum $s-t$ cut problem is MULTIWAY CUT, in which given a graph G and a set of terminals $\{s_1, s_2, \ldots, s_p\}$, the task is to find a minimum subset of vertices or edges whose deletion disconnects all the terminals from one another. Dahlhaus et al. [DJP+94] showed the edge version in undirected graphs is APX-complete for $p \geq 3$.

The problem behaves very differently on directed graphs. Interestingly, for directed graphs, the edge and vertex versions turn out to be equivalent. Garg et al. [GVY04] showed that computing a minimum multiway cut in directed graphs is NP-hard and MAX SNP-hard already for $p = 2$. Dahlhaus et al. [DJP+94] showed that undirected MULTIWAY CUT can be solved in time $n^{O(p)}$ on planar graphs, which can be an efficient solution if the number of terminals is small. On the other hand, on general graphs the problem becomes NP-hard already for $p = 3$. In both the directed and the undirected version, brute force can be used to check in time $n^{O(k)}$ if a solution of size at most k exists: one can go through all sets of size at most k. Thus the problem can be solved in polynomial time if the optimum is assumed to be small. In the undirected case, significantly better running time can be obtained: the current fastest algorithms run in $O^*(2^k)$ time for both the vertex version [CPPW13a] and the edge version [Xia10] (the O^* notation hides all factors which are polynomial in size of input).

Our main result in this chapter is that the directed version of MULTIWAY CUT is also fixed-parameter tractable parameterized by the solution size:

Theorem 2.9. DIRECTED VERTEX MULTIWAY CUT *and* DIRECTED EDGE MULTIWAY CUT *can be solved in* $O^*(2^{O(k^2)})$ *time.*

Note that the hardness result of Garg et al. [GVY04] shows that in the directed case the problem is nontrivial (in fact, NP-hard) even for $p = 2$ terminals; our result holds without any bound on the number of terminals. The question was first asked explicitly in [Mar06] and was also stated as an open problem in [MR11]. Our result shows in particular that DIRECTED MULTIWAY CUT is solvable in polynomial time if the size of the optimum solution is $O(\sqrt{\log n})$, where n is the number of vertices in the digraph.

We now formally define the DIRECTED MULTIWAY CUT problem. A multiway cut is a set of edges/vertices that separate the terminal vertices from each other:

Definition 2.10. (multiway cut) *Let G be a directed graph and let $T = \{t_1, t_2, \dots, t_k\} \subseteq V(G)$ be a set of terminals.*

1. *$S \subseteq V(G)$ is a* vertex multiway cut *of (G,T) if $G \setminus S$ does not have a path from t_i to t_j for any $i \neq j$.*
2. *$S \subseteq E(G)$ is a* edge multiway cut *of (G,T) if $G \setminus S$ does not have a path from t_i to t_j for any $i \neq j$.*

In the edge case, it is straightforward to define the problem we are trying to solve:

DIRECTED EDGE MULTIWAY CUT
Input : A directed graph G, an integer p and a set of terminals T.
Output : A multiway cut $S \subseteq E(G)$ of (G,T) of size at most p or "NO" if such a multiway cut does not exist.

In the vertex case, there is a slight technical issue in the definition of the problem: are the terminal vertices allowed to be deleted? We focus here on the version of the problem where the vertex multiway cut we are looking for has to be disjoint from the set of terminals. More generally, we define the problem in such a way that the graph has some *distinguished* vertices which cannot be included as part of any separator (and we assume that every terminal is a distinguished vertex). This can be modeled by considering weights on the vertices of the graph: weight of ∞ on each distinguished vertex and 1 on every non-distinguished vertex. We only look for solutions of finite weight. From here on, for a graph G we will denote by $V^{\infty}(G)$ the set of distinguished vertices of G with the meaning that these distinguished vertices cannot be part of any separator, i.e., all separators we consider are of finite weight. In fact, for any separator we can talk interchangeably about size or weight as these notions are the same since each vertex of separator has weight 1.

Our main focus is the following vertex version, where we require $T \subseteq V^{\infty}(G)$, i.e., terminals cannot be deleted:

DIRECTED VERTEX MULTIWAY CUT
Input : A directed graph G, an integer p, a set of terminals T and a set $V^{\infty} \supseteq T$ of distinguished vertices.
Output : A multiway cut $S \subseteq V(G) \setminus V^{\infty}(G)$ of (G,T) of size at most p or "NO" if such a multiway cut does not exist.

We note that if we want to allow the deletion of the terminal vertices, then it is not difficult to reduce the problem to the version defined above. For each terminal t we introduce a new vertex t' and we add the directed edges (t,t') and (t',t). Let the new graph be G' and let $T' = \{t' \mid t \in T\}$. Then there is a clear bijection between vertex multiway cuts which can include terminals in the instance (G,T,p) and vertex multiway cuts which cannot include terminals in the instance (G',T',p).

The two versions DIRECTED VERTEX MULTIWAY CUT and DIRECTED EDGE MULTIWAY CUT defined above are known to be equivalent. For sake of completeness, we prove the equivalence below. Now we concentrate on finding an FPT algorithm for DIRECTED VERTEX MULTIWAY CUT, which will be henceforth called as DIRECTED MULTIWAY CUT for brevity.

Equivalence of Vertex and Edge versions of DIRECTED MULTIWAY CUT: We first show how to solve the vertex version using the edge version. Let (G,T,p) be a given instance of DIRECTED VERTEX MULTIWAY CUT and let $V^{\infty}(G)$ be the set of distinguished vertices. We construct an equivalent instance (G',T',p) of DIRECTED EDGE MULTIWAY CUT as follows. Let the set V' contain two vertices v^{in}, v^{out} for every $v \in V(G) \setminus V^{\infty}(G)$ and a single vertex $u^{\text{in}} = u^{\text{out}}$ for every $u \in V^{\infty}(G)$. The idea is that all incoming/outgoing edges of v in G will now be incoming/outgoing edges of v^{in} and v^{out}, respectively. For every vertex $v \in V(G) \setminus V^{\infty}(G)$, add an edge $(v^{\text{in}}, v^{\text{out}})$ to G'. Let us call these as Type I edges. For every edge $(x,y) \in E(G)$, add $(p+1)$ parallel $(x^{\text{out}}, y^{\text{in}})$ edges. Let us call these as Type II edges. Define $T' = \{v^{\text{in}} \mid v \in T\}$. Note that the number of terminals is preserved. We have the following lemma:

Lemma 2.14. *(G,T,p) is a yes-instance of* DIRECTED VERTEX MULTIWAY CUT *if and only if (G',T',p) is a yes-instance of* DIRECTED EDGE MULTIWAY CUT.

Proof. Suppose G has a vertex multiway cut, say S, of size at most p. Then the set $S' = \{(v^{\text{in}}, v^{\text{out}}) \mid v \in S\}$ is clearly a edge multiway cut for G' and $|S'| = |S| \leq p$.

Suppose G' has an edge multiway cut say S' of size at most p. Note that it does not help to pick in S any edges of Type II as each edge has $(p+1)$ parallel copies and our budget is p. So let $S = \{v \mid (v^{\text{in}}, v^{\text{out}}) \in S'\}$. Then S is a vertex multiway cut for G and $|S| \leq |S'| \leq p$. □

We now show how to solve the edge version using the vertex version. Let (G,T,p) be a given instance of DIRECTED EDGE MULTIWAY CUT. We construct an equivalent instance (G',T',p) of DIRECTED VERTEX MULTIWAY CUT as follows. For each vertex $u \in V(G) \setminus T$, create a set C_u which contains u along with p other copies of u. For $t \in T$ we let $C_t = \{t\}$. For each edge $(u,v) \in E(G)$ create a vertex β_{uv}. Add edges (x, β_{uv}) for all $x \in C_u$ and (β_{uv}, y) for all $y \in C_v$. Define $T' = \bigcup_{t \in T} C_t = T$. Let $V^{\infty}(G') = T'$

Lemma 2.15. *(G,T,p) is a yes-instance of* DIRECTED EDGE MULTIWAY CUT *if and only if (G',T',p) is a yes-instance of* DIRECTED VERTEX MULTIWAY CUT.

Proof. Suppose G has an edge multiway cut, say S, of size at most p. Then the set $S' = \{\beta_{uv} \mid (u,v) \in S\}$ is clearly a vertex multiway cut for G' and $|S'| = |S| \leq p$.

Suppose G' has a vertex multiway cut say S' of size at most p. Note that it does not help to pick in S any vertices from the C_z of any vertex $z \in V(G) \setminus T$ as each vertex has $(p+1)$ equivalent copies and our budget is p. So let $S = \{(u,v) \mid \beta_{uv} \in S'\}$. Then S is a edge multiway cut for G and $|S| \leq |S'| \leq p$. □

2.2.1 Torsos and Shadowless Solutions

First we show that Directed Multiway Cut belongs to the general family of problems described by "$\mathcal{F}$-transversal for T-connected $\mathcal{F}$". Take T as the set of terminals and $\mathcal{F}$ as the set of all walks between different terminals; note that $\mathcal{F}$ is clearly T-connected. It is now easy to see that the problem exactly becomes the Directed Multiway Cut cut problem, and hence we can use the framework developed in Chapter 2. Applying Theorem 2.3, we obtain the following theorem

Theorem 2.10. (deterministic covering of the shadow) *For the* Directed Multiway Cut *problem, we can construct a set $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_t\}$ with $t = 2^{O(k^2)} \log^2 n$ in time $O^*(2^{O(k^2)})$ such that if there exists a solution of size $\leq k$, then there is a solution X of size $\leq k$ such that for at least one $i \in [t]$ we have*

1. *$X \cap Z_i = \emptyset$ and*
2. *Z_i covers the shadow of X.*

Shadowless Solutions: Recall that a solution S of Directed Multiway Cut is *shadowless* (with respect to T) if $f(S) = r(S) = \emptyset$. The following lemma shows the importance of *shadowless solutions* for Directed Multiway Cut. Clearly, any solution of the underlying undirected instance (where we disregard the orientation of the edges) is a solution for Directed Multiway Cut cut. The converse is not true in general: a solution of the directed problem is a not always solution of the undirected problem. However, the converse statement is true for shadowless solutions of the directed instance:

Lemma 2.16. *Let G^* be the underlying undirected graph of G. If S is a* shadowless *solution for an instance (G, T, p) of* Directed Multiway Cut*, then S is also a solution for the instance (G^*, T, p) of* Undirected Multiway Cut.

Proof. If S is a shadowless solution, then for each vertex v in $G \setminus S$, there is a $t_1 \rightarrow v$ path and a $v \rightarrow t_2$ path for some $t_1, t_2 \in T$. As S is a solution, it is not possible that $t_1 \neq t_2$: this would give a $t_1 \rightarrow t_2$ path in $G \setminus S$. Therefore, if S is a shadowless solution, then each vertex in the graph $G \setminus S$ belongs to the strongly connected component of exactly one terminal. A directed edge between the strongly connected components of t_i and t_j would imply the existence of either a $t_i \rightarrow t_j$ or a $t_j \rightarrow t_i$ path, which contradicts the fact that S is a solution of the Directed Multiway Cut instance. Hence the strongly connected components of $G \setminus S$ are exactly the same as the weakly connected components of $G \setminus S$, i.e., S is also a solution for the underlying instance of Undirected Multiway Cut. □

An illustration of Lemma 2.16 is given in Figure 2.4. Lemma 2.16 shows that if we can transform the instance in a way that ensures the existence of a shadowless solution,

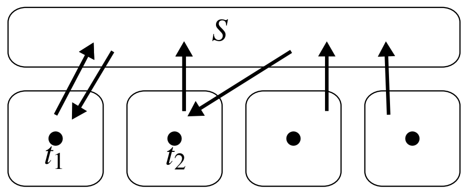


Figure 2.4: A shadowless solution S for a DIRECTED MULTIWAY CUT instance. Every vertex of $G \setminus S$ is in the strongly connected component of some terminal t_i. There are no edges between the strongly connected components of the terminals t_i, thus S is also a solution of the underlying UNDIRECTED MULTIWAY CUT instance.

then we can reduce the problem to undirected MULTIWAY CUT and use the $O^*(4^k)$ algorithm for that problem due to Guillemot [Gui11] which can handle the case when there are some distinguished vertices similar to what we consider.

Torsos: Suppose there is a set Z of vertices that we want to get rid of. However we must be careful: when getting rid of the set Z we should ensure that the information relevant to Z is captured in the reduced instance. This is exactly accomplished by the `torso` operation which removes a set of vertices without making the problem any easier. We formally define this operation as follows:

Definition 2.11. (torso) *Let G be a directed graph and let $C \subseteq V(G)$. The graph* $\texttt{torso}(G,C)$ *has vertex set C and there is a (directed) edge (a,b) in* $\texttt{torso}(G,C)$ *if there is an $a \to b$ path in G whose internal vertices are not in C.*

See Figure 2.6 for an example of the `torso` operation. Note that if $a,b \in C$ and (a,b) is a directed edge of G, then $\texttt{torso}(G,C)$ contains (a,b) as well. Thus $G[C]$, which is the graph induced by C in G, is a subgraph of $\texttt{torso}(G,C)$. The following lemma shows that the `torso` operation preserves separation inside C.

Lemma 2.17. (torso preserves separation) *Let G be a directed graph and $C \subseteq V(G)$. Let $G' = \mathtt{torso}(G,C)$ and $S \subseteq C$. For $a, b \in C \setminus S$, the graph $G \setminus S$ has an $a \to b$ path if and only if $G' \setminus S$ has an $a \to b$ path.*

Proof. Let P be a path from a to b in G. Suppose P is disjoint from S. Then P contains vertices from C and $V(G) \setminus C$. Let u, v be two vertices of C such that every vertex of P between u and v is from $V(G) \setminus C$. Then by definition there is an edge (u, v) in $\mathtt{torso}(G,C)$. Using such edges we can modify P to obtain an $a \to b$ path that lies completely in $\mathtt{torso}(G,C)$ but avoids S.

Conversely suppose P' is an $a \to b$ path in $\mathtt{torso}(G,C)$ and it avoids $S \subseteq C$. If P' uses an edge $(u, v) \notin E(G)$, then this means that there is a $u \to v$ path P'' whose internal vertices are not in C. Using such paths we modify P to get an $a \to b$ path P_0 that only uses edges from G. Since $S \subseteq C$ we have that the new vertices on the path are not in S and so P_0 avoids S. □

If we want to remove a set Z of vertices, then we create a new instance by taking the `torso` on the *complement* of Z:

Definition 2.12. *Let $I = (G, T, p)$ be an instance of* DIRECTED MULTIWAY CUT *and $Z \subseteq V(G) \setminus T$. The reduced instance $I/Z = (G', T', p)$ is defined as*

- $G' = \mathtt{torso}(G, V(G) \setminus Z)$
- $T' = T$

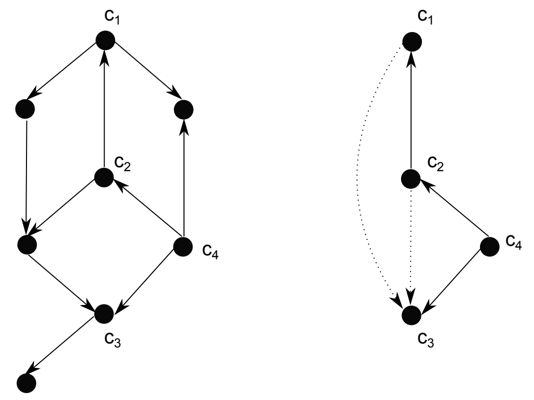


Figure 2.5: Let $C = \{c_1, c_2, c_3, c_4\}$. In the graph $\texttt{torso}(G,C)$ the edges (c_4, c_3) and (c_4, c_2) carry over from G. The new edges (shown by dotted arrows) that get added because of the `torso` operation are (c_1, c_3) and (c_2, c_3).

The following lemma states that the operation of taking the `torso` does not make the DIRECTED MULTIWAY CUT problem easier for any $Z \subseteq V(G) \setminus T$ in the sense that any solution of the reduced instance I/Z is a solution of the original instance I. Moreover, if we perform the `torso` operation for a set Z_i that satisfies the conditions of Theorem 2.10, i.e., Z_i is large enough to contain the shadow of some solution X but at the same time small enough to be disjoint from X, then X remains a solution for the reduced instance I/Z_i and in fact it is a shadowless solution for I/Z_i. Then by Lemma 2.16, we can apply the algorithm for undirected MULTIWAY CUT.

Lemma 2.18. (creating a shadowless instance) *Let $I = (G, T, p)$ be an instance of* DIRECTED MULTIWAY CUT *and $Z \subseteq V(G) \setminus T$.*

1. *If S is a solution for I/Z, then S is also a solution for I.*
2. *If S is a solution for I such that $f_{G,T}(S) \cup r_{G,T}(S) \subseteq Z$ and $S \cap Z = \emptyset$, then S is a shadowless solution for I/Z.*

Proof. Let G' be the graph $\texttt{torso}(G, V(G) \setminus Z)$. To prove the first part, suppose that $S \subseteq V(G')$ is a solution for I/Z and S is not a solution for I. Then there are terminals $t_1, t_2 \in T$ such that there is an $t_1 \to t_2$ path P in $G \setminus S$. As $t_1, t_2 \in T$ and $Z \subseteq V(G) \setminus T$, we have that $t_1, t_2 \in V(G) \setminus Z$. In fact, we have $t_1, t_2 \in (V(G) \setminus Z) \setminus S$. Lemma 2.17 implies that there is an $t_1 \to t_2$ path in $G' \setminus S$, which is a contradiction as S is a solution for I/Z.

For the second part of the lemma, let S be a solution for I such that $S \cap Z = \emptyset$ and $f_{G,T}(S) \cup r_{G,T}(S) \subseteq Z$. We want to show that S is a shadowless solution for I/Z. First we show that S is a solution for I/Z. Suppose to the contrary that there are terminals $x', y' \in T'(=T)$ such that $G' \setminus S$ has an $x' \to y'$ path. As $x', y' \in V(G) \setminus Z$, Lemma 2.17 implies $G \setminus S$ also has an $x' \to y'$ path, which is a contradiction as S is a solution of I.

Finally, we show that S is shadowless in I/Z, i.e., $r_{G',T}(S) = \emptyset = f_{G',T}(S)$. We only prove that $r_{G',T}(S) = \emptyset$: the argument for $f_{G',T}(S) = \emptyset$ is analogous. Assume to the contrary that there exists $w \in r_{G',T}(S)$ (note that we have $w \in V(G')$, i.e., $w \notin Z$). So S is a $w - T$ separator in G', i.e., there is no $w - T$ path in $G' \setminus S$. Lemma 2.17 gives that there is no $w - T$ path in $G \setminus S$, i.e., $w \in r_{G,T}(S)$. But $r_{G,T}(S) \subseteq Z$ and so we have $w \in Z$ which is a contradiction. Thus $r_{G,T}(S) \subseteq Z$ in G implies that $r_{G',T}(S) = \emptyset$. □

2.2.2 Proof of Theorem 2.9

The description of our algorithm is given in Algorithm 3. Recall that we are trying to solve a version of DIRECTED MULTIWAY CUT where we are given a set V^{∞} of distinguished vertices which are undeletable, i.e., have infinite weight. To solve the undirected

Algorithm 3: FPT ALGORITHM FOR DIRECTED MULTIWAY CUT

```
Input: An instance $I = (G, T, k)$ of DIRECTED MULTIWAY CUT.
1: Let $\mathcal{Z} = \{Z-1, Z_2, \ldots, Z_t\}$ be the set given by Theorem 2.10.
2: for Each $1 \leq i \leq t$ do
3:     Let $G_i = \texttt{torso}(G, V(G) \setminus Z_i)$.                {Get rid of $Z_i$}
4:     Solve the underlying undirected instance $(G_i^*, T, k)$ of MULTIWAY CUT.
5:     if $(G_i^*, T, k)$ has a solution $S$ then
6:         return $S$
7:     else
8:         return "NO"
```

MULTIWAY CUT instance (obtained by disregarding the orientation of the edges), we can use the algorithm of Guillemot [Gui11] that solves the undirected problem in time $O^*(4^k)$. Note that the algorithm for undirected MULTIWAY CUT in [Gui11] explicitly considers the variant where we have a set of distinguished vertices which cannot be deleted. By Theorem 2.10, the set $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_t\}$ can be constructed in time $O^*(2^{O(k^2)})$. For each Z_i, we need $O^*(4^k)$ time to solve the underlying *undirected* instance (G_i^*, T, k). Hence the total running time of Algorithm 3 is $O^*(2^{O(k^2)}) \times O^*(4^k) = O^*(2^{O(k^2)})$

The following two lemmas show the correctness of Algorithm 3. One direction is easy to see: the algorithm has no false positives.

Lemma 2.19. *Let $I = (G, T, k)$ be an instance of* DIRECTED MULTIWAY CUT. *If Algorithm 3 returns a set S, then S is a solution for I.*

Proof. Any solution S of the undirected instance (G_i^*, T, k) returned by Algorithm 3 is clearly a solution of the directed instance (G_i, T, k) as well. By Lemma 2.24(1) the `torso` operation does not make the problem easier by creating new solutions. Hence S is also a solution for $I = (G, T, k)$ □

The following lemma shows that if the instance has a solution, then the algorithm finds one with certain probability.

Lemma 2.20. *Let $I = (G, T,)$ be an instance of* DIRECTED MULTIWAY CUT. *If I is a yes-instance of* DIRECTED MULTIWAY CUT, *then Algorithm 3 returns a set X which is a solution for I.*

Proof. By Theorem 2.10, there exists $i \in [t]$ and a solution X of the DIRECTED MULTIWAY CUT instance such that $X \cap Z_i = \emptyset$ and Z_i covers the shadow of X. By Lemma 2.24, X is a shadowless solution of the instance I/Z_i. Lemma 2.16 implies that X is a solution of the underlying undirected instance (G_i^*, T, k). Hence Algorithm 3 will return the set X.

□

2.2.3 FPT Algorithm for DIRECTED MULTICUT with two terminals

A more general problem than MULTIWAY CUT is MULTICUT: the input contains a set $\{(s_1,t_1),\ldots,(s_p,t_p)\}$ of p pairs, and the task is to break every path from s_i to its corresponding t_i by the removal of at most k vertices. Very recently, it was shown that undirected MULTICUT is FPT parameterized by k [BDT11, MR11], but the directed version is unlikely to be FPT as it is W[1]-hard [MR11] with this parameterization. However, in the special case of $p = 2$ terminal pairs, there is a simple reduction from DIRECTED MULTICUT to DIRECTED MULTIWAY CUT, thus our result shows that the latter problem is FPT parameterized by k for $p = 2$. Let us briefly sketch the reduction. (Note that the reduction we sketch works only for the variant of DIRECTED MULTICUT which allows the deletion of terminals. Marx and Razgon [MR11] asked about the FPT status of this variant which is in fact equivalent to the one which does not allow deletion of the terminals.)

Corollary 2.1. DIRECTED MULTICUT *with* $p = 2$ *can be solved in time* $O^*(2^{O(k^2)})$.

Proof. Consider a given instance (G, T, k) of DIRECTED MULTICUT, and let $T = \{(s_1, t_1), (s_2, t_2)\}$. We construct an equivalent instance of DIRECTED MULTIWAY CUT as follows: Graph G' is obtained by adding two new vertices s, t to the graph and adding the four edges $s \to s_1$, $t_1 \to t$, $t \to s_2$, and $t_2 \to s$. It is easy to see that the DIRECTED MULTIWAY CUT instance $(G', \{s, t\}, k)$ is equivalent to the original DIRECTED MULTICUT instance. G has a $s_i \to t_i$ path for some i if and only if G' has a $s \to t$ or $t \to s$ path. This is because G has a $s_1 \to t_1$ path if and only if G' has a $s \to t$ path and G has a $s_2 \to t_2$ path if and only if G' has a $t \to s$ path. This property of paths also holds after removing some vertices/edges and thus the two instances are equivalent. □

The complexity of DIRECTED MULTICUT for the case with $p = 3$ terminals remains an interesting open problem.

2.3 Application II: FPT Algorithm for SUBSET-DFVS

The FEEDBACK VERTEX SET (FVS) problem has been one of the most extensively studied problems in the parameterized complexity community. Given a graph G and an integer k, it asks if there is a set $T \subseteq V(G)$ of size at most k which hits all cycles in G. The FVS problem in both undirected and directed graphs was shown to be NP-hard by

Karp [Kar72]. A generalization of the FVS problem is SUBSET FEEDBACK VERTEX SET (SFVS): given a subset $S \subseteq V(G)$ (resp., $S \subseteq E(G)$), find a set $T \subseteq V(G)$ of size at most k such that T hits all cycles passing through a vertex of S (resp., an edge of S). It is easy to see that $S = V(G)$ (resp., $S = E(G)$) gives the FVS problem.

As compared to undirected graphs, FVS behaves quite differently on directed graphs. In particular the trick of replacing each edge of an undirected graph G by arcs in both directions does not work: every feedback vertex set of the resulting digraph is a vertex cover of G and vice versa. Any other simple transformation does not seem possible either and thus the directed and undirected versions are very different problems.

In the undirected case, the first FPT algorithm for FVS in undirected graphs was given by Mehlhorn [Meh84] almost 30 years ago. Since then there have been a number of papers [BBYG00,Bod91,CCL10,CFL+08,DFL+07,DF99,GGH+06,KPS04,RSS06] giving faster algorithms and the current fastest (randomized) algorithm runs in time $O^*(3^k)$ [CNP+11]. For directed graphs, the fixed-parameter tractability status of FVS was a long-standing open problem (almost 16 years) until Chen et al. [CLL+08] resolved it by giving an $O^*(4^k k!)$ algorithm. This was recently generalized by Bonsma and Lokshtanov [BL11] who gave a $O^*(47.5^k k!)$ algorithm for FVS in mixed graphs, i.e., graphs having both directed and undirected edges.

In the more general SUBSET FEEDBACK VERTEX SET problem, an additional subset S of vertices is given and we want to find a set $T \subseteq V(G)$ of size at most k that hits all cycles passing through a vertex of S. In the edge version, we are given a subset $S \subseteq E(G)$

and we want to hit all cycles passing through an edge of S. The vertex and edge versions are indeed known to be equivalent in the parameterized sense in both undirected and directed graphs. Recently, Cygan et al. [CPPW13b] and independently Kakimura et al. [KKK12] have shown that SUBSET FEEDBACK VERTEX SET in undirected graphs is FPT parameterized by the size of the solution. Our main result in this chapter is that SUBSET FEEDBACK VERTEX SET in directed graphs is also fixed-parameter tractable parameterized by the size of the solution. This completes the picture for the parameterized complexity of feedback vertex set problems and their subset versions in undirected and directed graphs.

Theorem 2.11. SUBSET FEEDBACK VERTEX SET *(*SUBSET-DFVS*) in directed graphs can be solved in time* $O^*(2^{O(k^3)})$.

Observe that a directed graph contains no cycles if and only if it contains no closed walks; moreover, there is a cycle going through S if and only there is a closed walk going through S. For this reason, throughout we use the term closed walks instead of cycles, since it is sometimes easier to show the existence of a closed walk and avoid discussion whether it is a simple cycle or not. A *feedback vertex set* is a set of vertices that hits all the closed walks of the graph.

Definition 2.13. (feedback vertex set) *Let G be a directed graph. A set $T \subseteq V(G)$ is a* feedback vertex set *of G if $G \setminus T$ does not contain any closed walks.*

This gives rise to the DIRECTED FEEDBACK VERTEX SET (DFVS) problem where we are given a directed graph G and we want to find if G has a feedback vertex set of size at most k. The DFVS problem was shown to be FPT by Chen et al. [CLL+08], answering a long-standing open problem in the parameterized complexity community. We consider a generalization of the DFVS problem where given a set $S \subseteq V(G)$, we ask if there exists a vertex set of size $\leq k$ that hits all closed walks passing through S.

SUBSET DIRECTED FEEDBACK VERTEX SET (SUBSET-DFVS)
Input: A directed graph G, a set $S \subseteq V(G)$, and a positive integer k.
Parameter: k
Question: Does there exist a set $T \subseteq V(G)$ with $|T| \leq k$ such that $G \setminus T$ has no closed walk containing a vertex of S?

It is easy to see that DFVS is a special case of SUBSET-DFVS obtained by setting $S = V(G)$. We also define a variant of SUBSET-DFVS where the set S is a subset of edges. In this variant, we have destroy the following type of closed walks:

Definition 2.14. *(**S-closed-walk**) Let G be a directed graph and $S \subseteq E(G)$. A closed walk (starting and ending at same vertex) C in G is said to be a S*-closed-walk *if it contains an edge from S.*

EDGE SUBSET DIRECTED FEEDBACK VERTEX SET (EDGE-SUBSET-DFVS)
Input : A directed graph G, a set $S \subseteq E(G)$, and a positive integer k.
Parameter : k
Question : Does there exist a set $T \subseteq V(G)$ with $|T| \leq k$ such that $G \setminus T$ has no S-closed-walks?

The above two problems can be shown to be equivalent as follows. If (G, S, k) is an instance of SUBSET-DFVS we create an instance (G, S', k) of EDGE-SUBSET-DFVS by taking S' as the set of edges incident to any vertex of S. Then any closed walk passing through a vertex of S must pass through an edge of S', and conversely any closed walk passing through an edge of S' must contain a vertex from S.

On the other hand, given an instance (G,S',k) of EDGE-SUBSET-DFVS we create an instance (G',S,k) of SUBSET-DFVS where G' is obtained from G by the following modification: For every edge $(u,v) \in E(G)$ we add a new vertex x_{uv} and path $u \to x_{uv} \to v$ of length 2. We set $S = \{x_e \ : \ e \in S'\}$. Then any closed walk in G passing through an edge of S' corresponds to a closed-walk in G' which must pass through a vertex of S, and conversely any closed walk in G' passing through a vertex of S can be easily converted to a closed walk in G passing through an edge of S'. Both the reductions work in polynomial time and do not change the parameter. Henceforth we concentrate on solving the EDGE SUBSET DIRECTED FEEDBACK VERTEX SET problem and we shall refer to both the above problems as SUBSET-DFVS.

2.3.1 Applying Iterative Compression

The first step of our algorithm is to use the technique of *iterative compression* introduced by Reed et al. [RSV04]. who used it to show that the ODD CYCLE TRANSVERSAL[1] problem is FPT parameterized by the size of the deletion set. Since then it has been used to obtain faster FPT algorithms [CFL+08, CLL+08, DFL+07, GGH+06, HKMN08, MR11, RO09] and also in exact algorithms [FGK+10]. We transform the SUBSET-DFVS problem into the following problem:

[1]In this problem, we are given an undirected graph $G = (V,E)$ and the question is whether there exists a set $S \subseteq V$ such that $G \setminus S$ is odd-cycle free, i.e., bipartite

SUBSET-DFVS COMPRESSION
Input: A directed graph G, a set $S \subseteq E(G)$, a positive integer k, and a set $T \subseteq V$ such that $G \setminus T$ has no S-closed-walks.
Parameter: $k + |T|$
Question: Does there exist a set $T' \subseteq V(G)$ with $|T'| \leq k$ such that $G \setminus T'$ has no S-closed-walks?

Lemma 2.21. ***(power of iterative compression)*** SUBSET-DFVS *can be solved by $O(n)$ calls to an algorithm for the* SUBSET-DFVS COMPRESSION *problem with $|T| \leq k+1$.*

Proof. Let $V(G) = \{v_1, \dots, v_n\}$ and for $i \in [n]$ let $V_i = \{v_1, \dots v_i\}$. We construct a sequence of subsets $X_i \subseteq V_i$, such that X_i is a solution for $G[V_i]$. Clearly, $X_1 = \emptyset$ is a solution for $G[V_1]$. Observe that if X_i is a solution for $G[V_i]$, then $X_i \cup \{v_{i+1}\}$ is a solution for $G[V_{i+1}]$. Therefore, for each $i \in [n-1]$, we set $T = X_i \cup \{v_{i+1}\}$ and use, as a blackbox, an algorithm for SUBSET-DFVS COMPRESSION, to construct a set X_{i+1} that is a solution of size at most k for $G[V_{i+1}]$. Note that if there is no solution for $G[V_i]$ for some $i \in [n]$, then there is no solution for the whole graph G and moreover, since $V_n = V(G)$, if all the calls to the reduction problem are successful, then X_n is a solution for the graph G. $\square$

Now we transform the SUBSET-DFVS COMPRESSION problem into the following problem whose only difference is that the subset feedback vertex set in the output must be disjoint from the one in the input:

DISJOINT SUBSET-DFVS COMPRESSION
Input: A directed graph G, a set $S \subseteq E(G)$, a positive integer k, and a set $T \subseteq V$ such that $G \setminus T$ has no S-closed-walks.
Parameter: $k + |T|$
Question: Does there exist a set $T' \subseteq V(G)$ with $|T'| \leq k$ such that $T \cap T' = \emptyset$ and $G \setminus T'$ has no S-closed-walks?

Lemma 2.22. ***(adding disjointness)*** SUBSET-DFVS COMPRESSION *can be solved by*

$O(2^{|T|})$ *calls to an algorithm for the* DISJOINT SUBSET-DFVS COMPRESSION *problem.*

Proof. Given an instance $I = (G,S,T,k)$ of SUBSET-DFVS COMPRESSION we guess the intersection X of T and the subset feedback vertex set T' in the output. We have at most $2^{|T|}$ choices for X. Then for each guess for X, we solve the DISJOINT SUBSET-DFVS COMPRESSION problem for the instance $I_X = (G \setminus X, S, T \setminus X, k - |X|)$. It is easy to see that if T' is a solution for instance I of SUBSET-DFVS COMPRESSION, then $T' \setminus X$ is a solution of instance I_X of DISJOINT SUBSET-DFVS COMPRESSION for $X = T' \cap T$. Conversely, if T'' is a solution to some instance I_X, then $T'' \cup X$ is a solution for X. □

From Lemmas 2.21 and 2.22, an FPT algorithm for DISJOINT SUBSET-DFVS COMPRESSION translates into an FPT algorithm for SUBSET-DFVS with an additional blowup factor of $O(2^{|T|}n)$ in the running time.

2.3.2 Reduction to Shadowless Solutions

In DISJOINT SUBSET-DFVS COMPRESSION, the set T is the solution that we want to compress and $\mathcal{F}$ is the set of all S-closed-walks passing through some vertex of T. Again, $\mathcal{F}$ is T-connected: every S-closed-walk goes through T (as T is a solution), hence any vertex on an S-closed-walk is reachable from T, and some vertex of T is reachable from every vertex of the S-closed-walk. Hence, we can use the framework developed in Chapter 2 and apply Theorem 2.3 to construct a set Z of vertices that we want to get rid of. The second ingredient of our algorithm is an operation that removes a set of vertices without making the problem any easier. This transformation can be conveniently described using the operation of taking the *torso* of a graph. For DISJOINT SUBSET-DFVS COMPRESSION, we define it as follows:

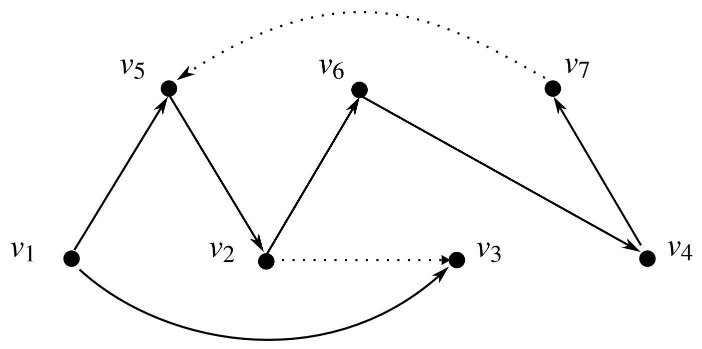
Definition 2.15. (torso) *Let (G,S,T,k) be an instance of* DISJOINT SUBSET-DFVS COMPRESSION *and $C \subseteq V(G)$. Then $\texttt{torso}(G,C,S)$ is a pair (G',S') defined as follows:*

- *G' has vertex set C and there is (directed) edge (a,b) in G' if there is an $a \to b$ walk in G whose internal vertices are not in C,*
- *S' contains those edges of S whose both endpoints are in C; furthermore, we add the edge (a,b) to S' if there is an $a \to b$ walk in G that contains an edge from S and whose internal vertices are not in C.*

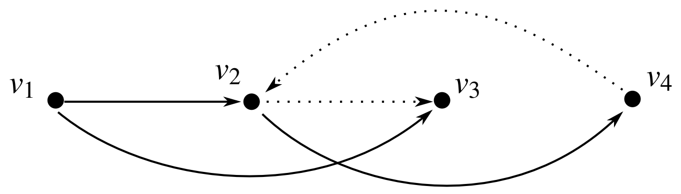
In particular, if $a,b \in C$ and (a,b) is a directed edge of G and $\texttt{torso}(G,C,S) = (G',S')$, then G' contains (a,b) as well. Thus G' is a supergraph of the subgraph of G induced by C. Figure 2.6 illustrates the definition of $\texttt{torso}$ with an example. The following lemma shows that the $\texttt{torso}$ operation preserves S-closed-walks inside C.

Lemma 2.23. (torso preserves S-closed-walks) *Let G be a directed graph with $C \subseteq V(G)$ and $S \subseteq E(G)$. Let $(G',S') = \texttt{torso}(G,C,S)$, $v \in C$, and $W \subseteq C$. Then $G \setminus W$ has an S-closed-walk passing through v if and only if $G' \setminus W$ has an S'-closed-walk passing through v.*

Proof. Let P be an S-closed-walk in $G \setminus W$ passing through v. If P is not contained in G, then it contains vertices from C and $V(G) \setminus C$. Let u,w be two vertices of C such that



The graph G



The graph $\texttt{torso}(G,C,S)$

Figure 2.6: In the top graph G we have $C = \{v_1, v_2, v_3, v_4\}$. The edges in S are given by the dotted lines. In the bottom graph we show the graph $\texttt{torso}(G,C,S)$. All edges from $G[C]$ appear in this graph. In addition, we also add the edges $(v_1,v_2),(v_2,v_4)$ and (v_4,v_2). The edge $(v_2,v_3) \in G[C] \cap S$ appears in S'. In addition, we also add the edge (v_4,v_2) to S' since the $v_4 \to v_7 \to v_5 \to v_2$ path in G has an edge $(v_7,v_5) \in S$.

every vertex of P between u and w is from $V(G) \setminus C$. Then, by definition of torso, there is an edge (u,w) in G'. Using such edges, we can modify P to obtain another closed walk say P' passing through v that lies completely in G' but avoids W. Note that since P is a S-closed-walk, at least one of the edges on some $u \to w$ walk that we short-circuited above must have been from S and by Definition 2.15 we would have put the edge (u,w) edge into S', which makes P' an S'-closed-walk in G'.

Conversely, suppose that P' is an S'-closed-walk passing through a vertex v in G' and it avoids $W \subseteq C$. If P' uses an edge $(u,w) \notin E(G)$, then this means that there is a

$u \to w$ walk P_{uw} whose internal vertices are not in C. Using such walks, we modify P' to get a closed walk P passing through v that only uses edges from G, i.e., P is a closed walk in $G \setminus W$. It remains to show that P is an S-closed-walk: since P' is an S'-closed-walk, either some edge of P' was originally in S or there exist some $a, b \in P'$ such that there is a $a \to b$ walk does not contain any vertex from C and some edge on this walk was originally in S. $\square$

If we want to remove a set Z of vertices, then we create a new instance by taking the torso on the *complement* of Z:

Definition 2.16. (reduced instance) *Let $I = (G, S, T, k)$ be an instance of* Disjoint Subset-DFVS Compression *and $Z \subseteq V(G) \setminus T$. The reduced instance $I/Z = (G', S', T, k)$ is obtained by setting $(G', S') = \texttt{torso}(G, V(G) \setminus Z, S)$.*

The following lemma states that the operation of taking the torso does not make the Disjoint Subset-DFVS Compression problem easier for any $Z \subseteq V(G) \setminus T$ in the sense that any solution of the reduced instance I/Z is a solution of the original instance I. Moreover, if we perform the torso operation for a Z that is large enough to cover the shadow of some solution T^* and also small enough to be disjoint from T^*, then T^* becomes a shadowless solution for the reduced instance I/Z.

Lemma 2.24. (creating a shadowless instance) *Let $I = (G, S, T, k)$ be an instance of* Disjoint Subset-DFVS Compression *and $Z \subseteq V(G) \setminus T$.*

1. *If I is a no-instance, then the reduced instance I/Z is also a no-instance.*

2. *If I has solution T' with $f_{G,T}(T') \cup r_{G,T}(T') \subseteq Z$ and $T' \cap Z = \emptyset$, then T' is a shadowless solution of I/Z.*

Proof. Let $C = V(G) \setminus Z$ and $(G', S') = \texttt{torso}(G, C, S)$. To prove the first statement, suppose that $T' \subseteq V(G')$ is a solution for I/Z. We show that T' is also a solution for I. Suppose to the contrary that there exists a vertex $v \in T$ such that $G \setminus T'$ has an S-closed-walk passing through v (since $G \setminus T$ has no S-closed-walks). Note that $v \in T$ and $Z \subseteq V(G) \setminus T$ implies $v \in C$. Then by Lemma 2.23, $G' \setminus T'$ also has an S'-closed-walk passing through v contradicting the fact that T' is a solution for I/Z.

For the second statement, let T' be a solution of I with $T' \cap Z = \emptyset$ and $f_{G,T}(T') \cup r_{G,T}(T') \subseteq Z$. We claim T' is a solution of I/Z as well. Suppose to the contrary that $G' \setminus T'$ has an S'-closed-walk passing through some vertex $v \in C$. As $v \in C$, Lemma 2.23 implies $G \setminus T'$ also has an S-closed-walk passing through v, which is a contradiction as T' is a solution of I.

Finally, we show that T' is a shadowless solution, i.e, $r_{G',T}(T') = f_{G',T}(T') = \emptyset$. We only prove $r_{G',T}(T') = \emptyset$: the argument for $f_{G',T}(T') = \emptyset$ is analogous. Assume to the contrary that there exists $w \in r_{G',T}(T')$ (note that we have $w \in V(G')$, i.e., $w \notin Z$). This means that T' is a $w - T$ separator in G', i.e., there is no $w - T$ walk in $G' \setminus T'$. By Lemma 2.23, it follows that there is no $w - T$ walk in $G \setminus T'$ either, i.e., $w \in r_{G,T}(T')$. But $r_{G,T}(T') \subseteq Z$ and therefore we have $w \in Z$, which is a contradiction. □

For every Z_i in the output of Theorem 2.3, we use the torso operation to remove the vertices in Z_i. We prove that this procedure is safe in the following sense:

Lemma 2.25. *Let $I = (G, S, T, k)$ be an instance of* Disjoint Subset-DFVS Compression. *Let the sets in the output of Theorem 2.3 be $Z_1, Z_2, \dots, Z_t$. For every $i \in [t]$, let G_i be the reduced instance G/Z_i.*

1. *If I is a no-instance, then G_i is also a no-instance for every $i \in [t]$.*
2. *If I is a yes-instance, then there exists a solution T^* of I which is a shadowless solution of some G_j for some $j \in [t]$.*

Proof. The first claim is easy to see: any solution T' of the reduced instance (G_i, S, T, k) is also a solution of (G, S, T, k) (by Lemma 2.24(1), the torso operation does not make the problem easier by creating new solutions).

By the derandomization of Covering algorithm, there is a $j \in [t]$ such that Z has the Property $(*)$, i.e., there is a solution T^* of I such that $Z \cap T^* = \emptyset$ and Z covers shadow of T^*. Then Lemma 2.24(2) implies that T^* is a shadowless solution for the instance $G_j = I/Z_j$. □

2.3.3 Finding a Shadowless Solution

Consider an instance (G, S, T, k) of Disjoint Subset-DFVS Compression. First, let us assume that we can reach a start point of some edge of S from each vertex of T, since otherwise we can clearly remove such a vertex from the graph (and from the set T) without changing the problem. Next, we branch on all $2^{O(k^2)} \log^2 n$ choices for Z taken from $\{Z_1, Z_2, \dots, Z_t\}$ (given by Theorem 2.3) and build a reduced instance I/Z for each choice of Z. By Lemma 2.24, if I is a no-instance, then I/Z_j is a no-instance for each

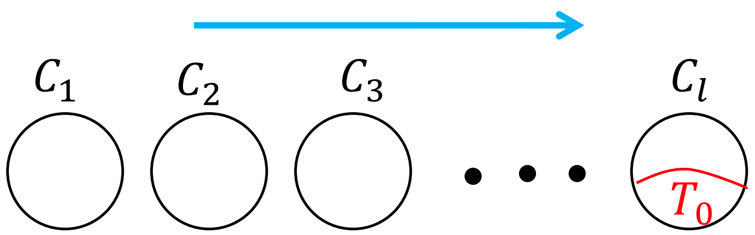


Figure 2.7: We arrange the strongly connected components of $G \setminus T'$ in a topological order so that the only possible direction of edges between the strongly connected components is as shown by the blue arrow. We will show later that the last component C_ℓ must contain a non-empty subset T_0 of T and further that no edge of S can be present within C_ℓ. This allows us to make some progress as we shall see in Theorem 2.13.

$j \in [t]$. If I is a yes-instance, then by the results of Section 2.1.2, there is at least one $i \in [t]$ such that I has a shadowless solution for the reduced instance I/Z_i.

Let us consider the branch where $Z = Z_i$ and let $T' \subseteq V \setminus T$ be a hypothetical shadowless solution for I/Z. We know that each vertex in $G \setminus T'$ can reach some vertex of T and can be reached from a vertex of T. Since T' is a solution for the instance (G, S, T, k) of Disjoint Subset-DFVS Compression, we know that $G \setminus T'$ does not have any S-closed-walks. Consider a topological ordering $C_1, C_2, \ldots, C_\ell$ of the strongly connected components of $G \setminus T'$, i.e., there can be an edge from C_i to C_j only if $i \leq j$. We illustrate this in Figure 2.7.

Definition 2.17. (starting/ending points of S) *Let S^- and S^+ be the sets of starting and ending points of edges in S respectively, i.e., $S^- = \{u \mid (u,v) \in S\}$ and $S^+ = \{v \mid (u,v) \in S\}$.*

Lemma 2.26. (properties of C_ℓ) *For a shadowless solution T' for an instance of* Disjoint Subset-DFVS Compression, *let C_ℓ be the last strongly connected component* *in the topological ordering of $G \setminus T'$ (refer to Figure 2.7). Then*

1. *C_ℓ contains a non-empty subset T_0 of T.*
2. *No edge of S is present within C_ℓ.*
3. *For each edge $(u,v) \in S$ with $u \in C_\ell$, we have $v \in T'$.*
4. *If $T' \cap S^+ = \emptyset$, then $C_\ell \cap S^- = \emptyset$.*

Proof.

1. If C_ℓ does not contain any vertex from T, then the vertices of C_ℓ cannot reach any vertex of T in $G \setminus T'$. This means that C_ℓ is in the shadow of T', which is a contradiction to the fact that T' is shadowless.
2. If C_ℓ contains an edge of S, then we will have an S-closed-walk in the strongly connected component C_ℓ, which is a contradiction, as T' is a solution for the instance (G,S,T,k) of DISJOINT SUBSET-DFVS COMPRESSION.
3. Consider an edge $(u,v) \in S$ such that $u \in C_\ell$ and $v \notin T'$. All outgoing edges from u must lie within C_ℓ, since C_ℓ is the last strongly connected component. In particular $v \in C_\ell$, which contradicts the second claim of the lemma.
4. Assume that $(u,v) \in S$ and $u \in C_\ell$ (which means $u \in C_\ell \cap S^-$). Since T' contains no vertex of S^+ we have $v \notin T'$ and by the third property we have $u \notin C_\ell$, a contradiction.

$\square$

Given a set X of removed vertices, we say that edge $(u,v) \in S$ is *traversable* from T_0 in $G \setminus X$ if $u,v \notin X$ and vertex u (and hence v) is reachable from T_0 in $G \setminus X$. If T' is a shadowless solution, then Lemma 2.26(2) implies that no edge of S is traversable from T_0 in $G \setminus T'$. There are two ways of making sure that an edge $(u,v) \in S$ is not traversable: (i) by making u unreachable, or (ii) by including v in T'. The situation is significantly simpler if every edge of S is handled the first way, that is, S^- is unreachable from T_0 in $G \setminus T'$. Then T' contains a $T_0 - S^-$ separator, and (as we shall see later) we may assume that T' contains an important $T_0 - S^-$ separator. Therefore, we can proceed by branching on choosing an important $T_0 - S^-$ separator of size at most k and including it into the solution.

The situation is much more complicated if some edges of S are handled the second way. Given a set X of vertices, we say that an edge $(u,v) \in S$ is *critical* (with respect to X) if $v \in X$ and u is reachable from T_0 in $G \setminus X$. Our main observation is that only a bounded number of vertices can be the head of a critical edge in a solution. Moreover, we can enumerate these vertices (more precisely, a bounded-size superset of these vertices) and therefore we can branch on including one of these vertices in the solution. We describe next how to enumerate these vertices.

Let us formalize the property of the vertices we are looking for:

Definition 2.18. (critical vertex) *For a fixed non-empty set $T_0 \subseteq V$, a vertex $v \in (V \setminus T_0) \cap S^+$ is called an ℓ-*critical vertex*, with respect to T_0, if there exist an edge $(u,v) \in S$ and a set $W \subseteq V \setminus T_0$ such that:*

- *$|W| \leq \ell$,*
- *edge (u,v) is critical with respect to W (that is, u is reachable from T_0 in $G \setminus W$ and $v \in W$),*
- *no edge of S is traversable from T_0 in $G \setminus W$.*

We say that v is witnessed by u, T_0 and W.

We need an upper bound on the number of critical vertices, furthermore our proof needs to be algorithmic, as we want to find the set of critical vertices, or at least its superset. Roughly speaking, to test if v is a critical vertex, we need to check if there is a set T' that separates every edge of S from T_0 in a way that some vertex u with $(u,v) \in S$ is still reachable from T_0. One could argue that it is sufficient to look at important separators: if there is such a separator where u is reachable from T_0, then certainly there is an important separator where u is reachable from T_0. However, describing the requirement as "separating every edge of S from T_0" is imprecise: what we need is that no edge of S is traversable from T_0, which cannot be simply described by the separation of two sets. We fix this problem by moving to an auxiliary graph G' by duplicating vertices; whether or not an edge of S is traversable from T_0 translates to a simple reachability question in G'. However, due to technical issues that arise from this transformation, it is not obvious how to enumerate precisely the k-critical vertices. Instead, we construct a set F of bounded size that contains each k-critical vertex, and potentially some additional vertices. Thus if the solution has a critical edge, then we can branch on including a vertex of F into the solution.

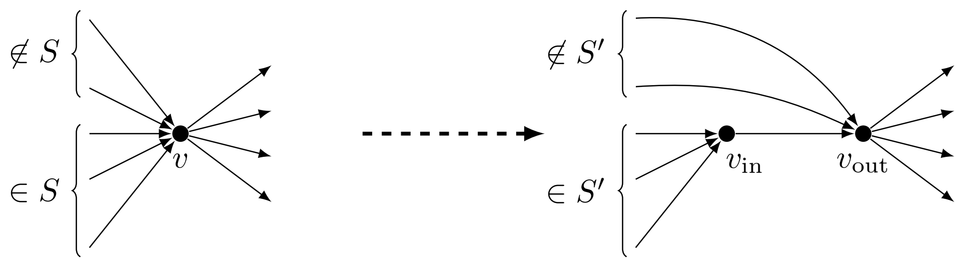


Figure 2.8: On the left there is a vertex v of G and on the right the corresponding vertices v_{in} and v_{out} of G'.

Theorem 2.12. (bounding critical vertices) *Given a directed graph G, a subset of its edges S, and a fixed non-empty subset $T_0 \subseteq V(G)$, we can find in time $O^*(2^{O(k)})$ a set F_{T_0} of $2^{O(k)}$ vertices that is a superset of all k-critical vertices with respect to T_0.*

Proof. We create an auxiliary graph G', where the vertex set of G' consists of two copies for each vertex of V and two extra vertices s and t, i.e., $V(G') = \{v_{\text{in}}, v_{\text{out}} : v \in V\} \cup \{s,t\}$. The edges of G' are defined as follows (see also Fig. 2.8):

- For each edge $e = (u,v) \in E(G)$, we add the following edges to $E(G')$: if $e \in S$, then add to $E(G')$ an edge $(u_{\text{out}}, v_{\text{in}})$, otherwise add to $E(G')$ an edge $(u_{\text{out}}, v_{\text{out}})$.
- For each vertex $v \in V$, we add to $E(G')$ an edge $(v_{\text{in}}, v_{\text{out}})$.
- For each vertex $v \in V$, we add an edge (v_{in}, t) to $E(G')$.
- For each vertex $v \in T_0$, we add an edge (s, v_{out}) to $E(G')$.

Let F'_{T_0} be the set of vertices of G' which belong to some important $s-t$ separator of size at most $2k$. By Lemma 2.1 the cardinality of F'_{T_0} is at most $2k \cdot 4^{2k}$. We define F_{T_0} as $\{v \in V : v_{\text{in}} \in F'_{T_0}\}$. Clearly, the claimed upper bound of $2^{O(k)}$ on $|F_{T_0}|$ follows, hence it remains to prove that each k-critical vertex belongs to F_{T_0}.

Let x be an arbitrary k-critical vertex witnessed by u, T_0 and W. Define $W' = \{v_{\text{in}}, v_{\text{out}} : v \in W\}$ and note that $|W'| \leq 2k$. The only out-neighbors of s are $\{v_{\text{out}} \mid v \in T_0\}$ while the only in-neighbors of t are $\{v_{\text{in}} \mid v \in V\}$. Hence the existence of an $s-t$ path in G' implies that there is in fact an edge $(a,b) \in S$ that is traversable from T_0 in $G \setminus W$. This is a contradiction to Definition 2.18. Therefore, no in-neighbor of t is reachable from s in $G' \setminus W'$, i.e., W' is an $s-t$ separator. Finally, a path from T_0 to u in $G \setminus W$ translates into a path from s to u_{out} in $G' \setminus W'$. Consider an important $s-t$ separator W'', i.e., $|W''| \leq |W'|$ and $R^+_{G' \setminus W'}(s) \subset R^+_{G' \setminus W''}(s)$. As u_{out} is reachable from s in $G' \setminus W'$ we infer that u_{out} is also reachable from s in $G' \setminus W''$. Consequently $x_{\text{in}} \in W''$, as otherwise there would be an $s-t$ path in $G' \setminus W''$. Hence x_{in} belongs to F'_{T_0}, which implies that x belongs to F_{T_0} and the theorem follows. $\square$

The following theorem characterizes a solution, so that we can find a vertex contained in it by inspecting a number of vertices in V bounded by a function of k. We apply Theorem 2.12 for each subset $T_0 \subseteq T$ and let $F = \bigcup_{T_0 \subseteq T} F_{T_0}$. Note that $|F| \leq 2^{|T|} \cdot 2^{O(k)} = 2^{O(|T|+k)}$, and we can generate F in time $2^{|T|} \cdot O^*(2^{O(k)}) = O^*(2^{O(|T|+k)})$

Theorem 2.13. (pushing) *Let $I = (G,S,T,k)$ be an instance of* Disjoint Subset-DFVS Compression *having a shadowless solution and let F be a set generated by the algorithm of Theorem 2.12. Let G^+ be obtained from G by introducing a new vertex t and adding an edge (u,t) for every $u \in S^-$. Then there exists a solution $T' \subseteq V \setminus T$ for I such that either*

- *T' contains a vertex of $F \setminus T$, or*

- *T' contains an important $T_0 - (\{t\} \cup (T \setminus T_0))$ separator of G^+ for some non-empty $T_0 \subseteq T$.*

Proof. Let T' be any shadowless solution for I and by Property 1 of Lemma 2.26 let T_0 be the non-empty subset of T belonging to the last strongly connected component of $G \setminus T'$.

We consider two cases: either there is a $T_0 - S^-$ path in $G \setminus T'$ or not. First assume that there is a path from T_0 to a vertex $u \in S^-$ in $G \setminus T'$. Clearly, $u \in C_\ell$, since all vertices of T_0 belong to C_ℓ and no edge from C_ℓ can go to previous strongly connected components. Consider any edge from S that has u as its starting point, say $(u,v) \in S$. By Property 3 of Lemma 2.26, we know that $v \in T'$. Observe that v is a k-critical vertex witnessed by u, T_0, and T', since $|T'| \leq k$, by definition of u, there is a path from T_0 to u in $G \setminus T'$; and by Property 3 of Lemma 2.26, no edge of S is traversable from T_0. Consequently, by the property of the set F, we know that $v \in T' \cap F \neq \emptyset$ and the theorem holds.

Now we assume that no vertex of S^- is reachable from T_0 in $G \setminus T'$. By the definition of T_0, the set T' is a $T_0 - (T \setminus T_0)$ separator in G, hence we infer that T' is a $T_0 - (\{t\} \cup (T \setminus T_0))$ separator in G^+. Let T^* be the subset of T' reachable from T_0 without going through any other vertices of T'. Then T^* is clearly a $T_0 - (\{t\} \cup (T \setminus T_0))$ separator in G^+. Let T^{**} be the minimal $T_0 - (\{t\} \cup (T \setminus T_0))$ separator contained in T^*. If T^{**} is an important $T_0 - (\{t\} \cup (T \setminus T_0))$ separator, then we are done, as T' itself contains T^{**}.

Otherwise, there exists an important $T_0 - (\{t\} \cup (T \setminus T_0))$ separator T^{***} that dominates T^{**}, i.e., $|T^{***}| \leq |T^{**}|$ and $R^+_{G^+ \setminus T^{**}}(T_0) \subset R^+_{G^+ \setminus T^{***}}(T_0)$. Now we claim that $T'' = (T' \setminus T^{**}) \cup T^{***}$ is a solution for the instance (G,S,T,k) of DISJOINT SUBSET-

DFVS COMPRESSION. If we show this, then we are done, as $|T''| \leq |T'|$ and T'' contains the important $T_0 - (\{t\} \cup (T \setminus T_0))$ separator T^{***}.

Suppose T'' is a not a solution for the instance (G, S, T, k) of DISJOINT SUBSET-DFVS COMPRESSION. We have $|T''| \leq |T'| \leq k$ (as , $|T^{***}| \leq |T^{**}|$) and $T'' \cap T = \emptyset$ (as T^{***} is an important $T_0 - (\{t\} \cup (T \setminus T_0))$ separator of G^+, hence disjoint from T). Therefore, the only possible problem is that there is an S-closed-walk in $G \setminus T''$ passing through some vertex $v \in T^{**} \setminus T^{***}$; in particular, this implies that there is a $v - S^-$ walk in $G \setminus T''$. Since T^{**} is a minimal $T_0 - (\{t\} \cup (T \setminus T_0))$ separator, we have $(T^{**} \setminus T^{***}) \subseteq R^+_{G^+ \setminus T''}(T_0)$, implying $v \in R^+_{G^+ \setminus T''}(T_0)$. This gives a $T_0 - S^-$ walk via v in $G \setminus T''$, a contradiction as T'' contains an (important) $T_0 - (\{t\} \cup (T \setminus T_0))$ separator by construction.

□

Theorem 2.13 tells us that there is always a minimum solution which either contains some critical vertex of F or an important $T_0 - (\{t\} \cup (T \setminus T_0))$ separator of G^+ where T_0 is a non-empty subset of T. In the former case, we branch into $|F|$ instances, in each of which we put one vertex of F to the solution, generating $2^{O(|T|+k)}$ instances with reduced budget. Next we can assume that the solution does not contain any vertex of F and we try all $2^{|T|} - 1$ choices for T_0. For each guess of T_0 we enumerate at most 4^k important $T_0 - (\{t\} \cup (T \setminus T_0))$ separators of size at most k in time $O^*(4^k)$ as given by Lemma 2.1. This gives the branching algorithm described in Algorithm 4.

Algorithm 4: BRANCH

```
Input: An instance $I = (G, S, T, k)$ of DISJOINT SUBSET-DFVS COMPRESSION.
Output: A new set of $2^{O(|T|+k)}$ instances of DISJOINT SUBSET-DFVS
COMPRESSION where the budget $k$ is reduced.
1: for every vertex $v \in F \setminus T$ found by Theorem 2.12 do
2:     Create a new instance $I_v = (G \setminus v, S, T, k-1)$ of DISJOINT SUBSET-DFVS
       COMPRESSION.
3: for every non-empty subset $T_0$ of $T$: do
4:     Use Lemma 2.1 to enumerate all the at most $4^k$ important $T_0 - (\{t\}^- \cup (T \setminus T_0))$
       separators of size at most $k$ in $G^+$.
5:     Let the important separators be $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$.
6:     for each $i \in [m]$ do
7:         Create a new instance $I_{T_0,i} = (G \setminus B_i, S, T, k - |B_i|)$ of DISJOINT
           SUBSET-DFVS COMPRESSION.
```

2.3.4 DISJOINT SUBSET-DFVS COMPRESSION: Summary of Algorithm

Lemma 2.25 and the BRANCH algorithm together combine to give a bounded search tree FPT algorithm for DISJOINT SUBSET-DFVS COMPRESSION described in Algorithm 5.

We then repeatedly perform Steps 1 and 2. Note that for every instance, one execution of steps 1 and 2 gives rise to $2^{O(k^2)} \log^2 n$ instances such that for each instance, either we know that the answer is NO or the budget k has decreased, because we have assumed that from each vertex of T one can reach the set S^-, and hence each important separator is non-empty. Therefore, considering a level as an execution of Step 1 followed by Step 2, the height of the search tree is at most k. Each time we branch into at most $2^{O(k^2)} \cdot \log^2 n$ directions (as $|T|$ is at most $k+1$). Hence the total number of nodes in the search tree is

Algorithm 5: FPT Algorithm for SUBSET-DFVS

Step 1: For a given instance $I = (G, S, T, k)$, use Theorem 2.3 to obtain a set of instances $\{Z_1, Z_2, \ldots, Z_t\}$ where $t = 2^{O(k^2)} \cdot \log^2 n$ and Lemma 2.25 implies

- If I is a no-instance, then all the reduced instances $G_j = G/Z_j$ are no-instances for all $j \in [t]$
- If I is a yes-instance, then there is at least one $i \in [t]$ such that there is a solution T^* for I which is a shadowless solution for the reduced instance $G_i = G/Z_i$.

At this step we branch into $2^{O(k^2)} \cdot \log^2 n$ directions.

Step 2 : For each of the instances obtained from the above step, we run the BRANCH algorithm to obtain a set of $2^{O(k+|T|)}$ instances where in each case either the answer is NO, or the budget k is reduced. We solve these instances recursively and return YES if at least one of them returns YES.

$\left(2^{O(k^2)} \cdot \log^2 n\right)^k$.

Lemma 2.27. *For every n and $k \le n$, we have* $(\log n)^k \le (2k \cdot \log k)^k + \frac{n}{2^k}$

Proof. If $\frac{\log n}{1+\log\log n} \ge k$, then $n \ge (2\log n)^k$. Otherwise we have $\frac{\log n}{1+\log\log n} < k$ and then $(4k\log k) \ge (2\log n)$ as follows: $2k\log k \ge \frac{2\log n \log k}{1+\log\log n}$. Now $\frac{2\log n \log k}{1+\log\log n} \ge \log n \Leftrightarrow 2\log k \ge 1 + \log\log n \Leftrightarrow k^2 \ge 2\log n$. But, $\frac{k^2}{2\log n} = \frac{\log n}{2(1+\log\log n)^2}$ which is greater than 1 for $n \ge 2^{2^7}$.

□

The total number of nodes in the search tree is $\left(2^{O(k^2)} \cdot \log^2 n\right)^k = \left(2^{O(k^2)}\right)^k (\log^2 n)^k = (2^{O(k^3)})(\log^2 n)^k \le (2^{O(k^3)})\left((2k \cdot \log k)^k + \frac{n}{2^k}\right)^2 \le 2^{O(k^3)} n^2$.

We then check the leaf nodes and see if there are any S-closed-walks left even after the budget k has become zero. If the graph in at least one of the leaf nodes is S-closed-walk free, then the given instance is a yes-instance. Otherwise it is a no-instance. This gives an $O^*(2^{O(k^3)})$ algorithm for DISJOINT SUBSET-DFVS COMPRESSION. By Lemma 2.21, this gives an $O^*(2^{O(k^3)})$ algorithm for the SUBSET-DFVS problem.

CHAPTER 3

Optimal Algorithms for Connectivity Problems

The STEINER TREE (ST) problem is one of the earliest and most fundamental problems in combinatorial optimization: given an undirected graph $G = (V, E)$ and a set $T \subseteq V$ of terminals, the objective is to find a tree of minimum size which connects all the terminals. The STEINER TREE problem is believed to have been first formally defined by Gauss in a letter in 1836. The first combinatorial formulation of the ST problem is attributed independently to Hakimi [Hak71] and Levin [Lev71] in 1971. The ST problem is known be to NP-complete, and was in fact was one of Karp's original list [Kar72] of 21 NP-complete problems. In the directed version of the ST problem, called DIRECTED STEINER TREE (DST), we are also given a root vertex r and the objective is to find a minimum size arborescence which connects the root r to each terminal from T.

DIRECTED STEINER TREE (DST)
Input : A directed graph $G = (V,E)$, a root vertex r and a set of terminals $T = \{t_1, t_2, \ldots, t_k\}$.
Question : Find the smallest $S \subseteq V(G)$ such that $G[S]$ has a $r \rightsquigarrow t_i$ path for each $i \in [k]$.

Steiner-type of problems arise in the design of networks. Since many networks are symmetric, the directed versions of Steiner type of problems were mostly of theoretical interest. However in recent years, it has been observed [Ram96, SRV97] that the connection cost in various networks such as satellite or radio networks are not symmetric. Therefore, directed graphs form the most suitable model for such networks. In addition, Ramanathan [Ram96] also used the DST problem to find low-cost multicast trees, which have applications in point-to-multipoint communication in high bandwidth networks. We refer the interested reader to Winter [Win87] for a survey on applications of Steiner problems in networks.

We consider two generalizations of the DST problem, namely the STRONGLY CONNECTED STEINER SUBGRAPH problem (by requiring two-way connectivity) and the DIRECTED STEINER FOREST problem (by requiring connectivity between terminal pairs). We define the problems formally:

STRONGLY CONNECTED STEINER SUBGRAPH (SCSS)
Input : A directed graph $G = (V,E)$ and a set of terminals $T = \{t_1, t_2, \ldots, t_k\}$.
Question : Find the smallest $S \subseteq V(G)$ such that $G[S]$ has a $t_i \rightsquigarrow t_j$ path for each $1 \leq i \neq j \leq k$.

DIRECTED STEINER FOREST (DSF)
Input : A directed graph $G = (V,E)$ and a pair of terminals $T = \{(s_1,t_1), (s_2,t_2), \ldots, (s_k,t_k)\}$.
Question : Find the smallest $S \subseteq V(G)$ such that $G[S]$ has a $s_i \rightsquigarrow t_i$ path for each $i \in [k]$.

The following reduction shows that SCSS is a special case of DSF: an instance of SCSS with k terminals can be viewed as an instance of DSF with $k(k-1)$ terminal pairs by listing all ordered two-tuples of the terminals. Now we describe the known results for both SCSS and DSF.

Previous Work on SCSS and DSF: Since both DSF and SCSS are NP-complete, one can try to design polynomial-time approximation algorithms for these problems. An α-approximation for DST implies a 2α-approximation for SCSS as follows: fix a terminal $t \in T$ and take the union of the solutions of the DST instances $(G, t, T \setminus t)$ and $(G_{\text{rev}}, t, T \setminus t)$, where G_{rev} is the graph obtained from G by reversing the orientations of all edges. The best known approximation ratio in polynomial time for SCSS is k^{ε} for any $\varepsilon > 0$ [CCC+99]. A result of Halperin and Krauthgamer [HK] implies SCSS has no $\Omega(\log^{2-\varepsilon} n)$-approximation for any $\varepsilon > 0$, unless NP has quasi-polynomial Las Vegas algorithms. For the more general DSF problem, the best known approximation ratio is $n^{2/3+\varepsilon}$ for any $\varepsilon > 0$. Berman et al. [BBM+13] showed that DSF has no $\Omega(2^{\log^{1-\varepsilon} n})$-approximation for any $0 < \varepsilon < 1$, unless NP has a quasi-polynomial time algorithm.

Rather than finding approximate solutions in polynomial time, one can look for exact solutions in time that is still better than the running time obtained by brute force solutions. For both SCSS and DSF problems, brute force can be used to check in time $n^{O(p)}$ if a solution of size at most p exists: one can go through all sets of size at most p. Recall that a problem is *fixed-parameter tractable* (FPT) with a particular parameter p if it can be solved in time $f(p) \cdot n^{O(1)}$, where f is an arbitrary function depending only on

p. One can also consider parameterization by the number k of terminals (terminal pairs); with this parameterization, it is not even clear if there is a polynomial-time algorithm for every fixed k, much less if the problem is FPT. It is known that STEINER TREE on undirected graphs is FPT: the classical algorithm of Dreyfus and Wagner [DW71] solves the problem in time $3^k \cdot n^{O(1)}$, where k is the number of terminals. The running time was recently improved to $2^k \cdot n^{O(1)}$ by Björklund et al. [BHKK07]. The same algorithms work for DIRECTED STEINER TREE as well.

For the SCSS and DSF problems, we cannot expect fixed-parameter tractability: Guo et al. [GNS11] showed that SCSS is W[1]-hard parameterized by the number of terminals k, and DSF is W[1]-hard parameterized by the number of terminal pairs k. In fact, it is not even clear how to solve these problems in polynomial time for small fixed values of the number k of terminals/pairs. The case of $k = 1$ in DSF is the well-known shortest path problem in directed graphs, which is known to be polynomial time solvable. For the case $k = 2$ in DSF, an $O(n^5)$ algorithm was given by Li et al. [LMSL92] which was later improved to $O(mn + n^2 \log n)$ by Natu and Fang [NF97]. The question regarding the existence of a polynomial algorithm for DSF when $k = 3$ was open. Feldman and Ruhl [FR06] solved this question by giving an $n^{O(k)}$ algorithm for DSF, where k is the number of terminal pairs. They first designed an $n^{O(k)}$ algorithm for SCSS, where k is the number of terminals, and used it as a subroutine in the algorithm for the more general DSF problem.

Steiner Problems on Planar Graphs: Given the amount of attention the planar version of Steiner-type problems received from the viewpoint of approximation (see, e.g., [BCE$^+$11, BHM11, BKM09, DHK09, EKM12]) and the availability of techniques for parameterized algorithms on planar graphs (see, e.g., [BFL$^+$09, DH08, FG01]), it is natural to explore SCSS and DSF restricted to planar graphs. In general, one can have the expectation that the problems restricted to planar graphs become easier, but sophisticated techniques might be needed to exploit planarity. For the SCSS problem, we show in Section 3.1.2 that one can obtain a faster algorithm in planar graphs as compared to general graphs. However, for the more general DSF problem it turns out that the best known algorithm is tight even for planar graphs (see Section 3.2.4)

Other Parameterizations: Instead of parameterizing by the number of terminals/terminal pairs, we can consider parameterization by the number of edges/vertices. Let us briefly and informally discuss this parameterization. Note that the number of terminals is a lower bound on the number of edges/vertices of the solution (up to a factor of 2 in the case of DSF parameterized by the number of edges), thus fixed-parameter tractability could be easier to obtain by parameterizing with the number of edges/vertices. However, our lower bound for SCSS on general graphs (as well as the W[1]-hardness of Guo et al. [GNS11]) actually proves hardness also with these parameterizations, making fixed-parameter tractability unlikely. On the other hand, it follows from standard techniques that both SCSS and DSF are FPT on planar graphs when parameterizing by the number k of edges/vertices in the solution. The main argument here is that the solution is fully contained in the k-neighborhood of the terminals, whose number is $O(k)$. It is known that the k-neighborhood of $O(k)$ vertices in a planar graph has treewidth $O(k)$, thus one can use standard techniques on bounded-treewidth graphs (dynamic programming or Courcelle's Theorem). Alternatively, at least in the unweighted case, one can formulate the problem as a first order formula of size depending only on k and then invoke the result of Frick and Grohe [FG01] stating that such problems are FPT. Therefore, as fixed-parameter tractability is easy to establish on planar graphs, the challenge here is to obtain optimal dependence on k. One would expect a subexponential dependence on k (e.g., $2^{O(\sqrt{k})}$ or $k^{O(\sqrt{k})}$) at least for SCSS, but this is not yet fully understood even for undirected STEINER TREE [PPSvL13]. A slightly different parameterization is to consider the number k of *nonterminal* vertices in the solution, which can be much smaller than the number of terminals. This leads to problems of somewhat different flavor, see, e.g., [JLR$^+$13].

3.1 Upper Bounds

3.1.1 The $n^{O(k)}$ Algorithms of Feldman-Ruhl for SCSS

In this section we give a self-contained description of Feldman-Ruhl algorithms for SCSS [FR06].

3.1.1.1 **Legal Token Moves**

Let the set of terminals for SCSS be $T = \{t_1, t_2, \dots, t_k\}$. For ease of notation, we set $q := k-1$ and $r = t_k$. Any solution H for SCSS contains paths from each of $t_1, t_2, \dots, t_{k-1}$ to r. These paths together can be chosen to form an in-tree T_{in} rooted at r. Similarly H must also contain paths from r to each of $t_1, t_2, \dots, t_{k-1}$: these paths together can be chosen to form an out-tree T_{out} rooted at r. Furthermore, any subgraph H which is the union of such an in-tree and an out-tree rooted at r is a solution for SCSS. However, a crucial feature of the problem is that these two trees can share edges/vertices, thus taking the union of an optimum in-tree and an optimum out-tree is not necessarily an optimum solution.

The algorithm can be visualized as follows: we have two types of tokens, namely F-tokens and B-tokens. Place a "F-token" and a "B-token" at each t_i for $i \in [q]$. The F-tokens move forward along edges of the in-tree T_{in} towards r. The B-tokens move backward along edges of the out-tree T_{out} towards r. The set of tokens left at any stage are called "alive" tokens. Since tokens of the same type trace out a tree, as soon as two tokens of the same type arrive at a common vertex we can merge them into one token. This can also be viewed as one token "eating up" the other token, which then becomes dead. Therefore it is enough to describe the pair of sets $\langle F, B \rangle$ which denote the set of nodes occupied by the F-tokens and B-tokens, respectively. Since there are at most q tokens of each type, the sets F, B have size at most q. Let $\binom{V}{\leq q}$ denote the set of subsets of $V(G)$ of size at most q. We now define the set of "legal" token moves and show that a minimum solution corresponds to a solution for SCSS.

Legal Token Moves for SCSS

1. Single Moves for F-tokens: For each edge $(u,v) \in E$ and all token sets $F,B \in \binom{V}{\leq q}$ such that $u \in F$, we have the move
$$\langle F,B\rangle \xrightarrow{c} \langle (F \setminus u) \cup \{v\}, B\rangle.$$
The cost c of this move is 1 if $v \notin F \cup B$ and 0 otherwise.
2. Single Moves for B-tokens: For each edge $(u,v) \in E$ and all token sets $F,B \in \binom{V}{\leq q}$ such that $v \in B$ we have the move
$$\langle F,B\rangle \xrightarrow{c} \langle F, (B \setminus v) \cup \{u\}\rangle.$$
The cost c of this move is 1 if $u \notin F \cup B$ and 0 otherwise.
3. Flipping: For every pair f,b and vertex sets $F,B,F' \subset F, B' \subset B$ such that
   - $f \in F$ and $F \in \binom{V}{\leq q}$,
   - $b \in B$ and $B \in \binom{V}{\leq q}$, and
   - there is an $f \rightsquigarrow b$ walk in G going through all vertices in $F' \cup B'$,

   we have the move
$$\langle F,B\rangle \xrightarrow{c} \langle (F \setminus (\{f\} \cup F') \cup \{b\}), (B \setminus (\{b\} \cup B') \cup \{f\})\rangle.$$
The cost c of this move is discussed below.

Cost of flips. There is a technical issue about the cost of a flipping move that is not explained in detail in [FR06]. Initially, [FR06] defines the cost c of the move as the size of the set M of vertices of a shortest walk from f to b in G going through all vertices in $F' \cup B'$, excluding f,b and vertices in $F' \cup B'$. The problem is that it is not clear how to find a walk minimizing this definition of cost. However, one can try all possible ordering of the tokens in $F' \cup B'$, and find a shortest walk that visits the tokens in this order. Then we can define the cost of the walk as its lengths plus one (i.e., the number of visited vertices, possibly with repetitions), minus the size of the set $\{f,b\} \cup F' \cup B'$. We will denote the

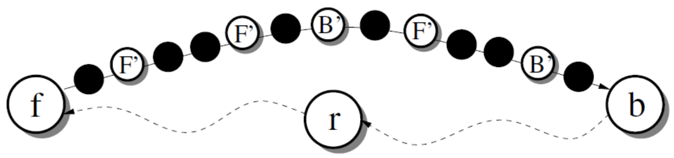


Figure 3.1: Flipping move between f and b: the black nodes form the set M and tokens $F' \cup B'$ need to be "picked up". Figure taken from [FR06]

cost as c_1-cost and c_2-cost if the cost of a flip is interpreted these two ways, respectively. Clearly, the c_1-cost is at most the c_2-cost. It turns out that these two costs are the same in optimum solutions (see Lemmas 3.1 and 3.2 below).

Intuition about the legal moves. A single move for an F-token corresponds to that F-token moving forward along an edge. Similarly, a single move for a B-token corresponds to that B-token moving backward along an edge. We charge only if the new vertex (where the token has now moved to) does not already have a token on it. The flipping move allows F-tokens and B-tokens to pass each other. The two outer tokens are an F-token f and a B-token b (see Figure 3.1). In between the outer tokens f and b, there are other F-tokens moving forward along the edges and trying to pass b, and B-tokens moving backward along edges and trying to pass f. These tokens, which occupy the vertex sets F' and B' respectively, are *picked up* during the flipping move.

Building the game graph $\tilde{G}$. Let $\tilde{V} = \binom{V}{\leq q} \times \binom{V}{\leq q}$. Build a game graph $\tilde{G} = (\tilde{V}, \tilde{E})$, where $\tilde{E}$ is the set of all legal token moves. We assign weights to the edges of $\tilde{G}$ according to the costs of the corresponding legal moves. Consider a single move for an F-token given by $\langle F,B\rangle \xrightarrow{c} \langle (F\setminus u)\cup\{v\},B\rangle$. Its cost can be computed easily: it is 1 if $v\notin F\cup B$, and 0 otherwise. Similarly, the cost of a single move for a B-token can be computed easily. On the other hand, to compute the cost of a flipping move between f and b, we need to find the size of the shortest $f\rightsquigarrow b$ walk in G that passes through all vertices in $F'\cup B'$. The main observation is the following: if we know the order in which the vertices from $F'\cup B'$ appear on the $f\rightsquigarrow b$ walk, then the shortest walk is just the concatenation of shortest paths between two consecutive nodes in the ordering. The number of tokens is at most $2q$ and hence $|F'\cup B'|\le 2q-2$. We try all the at most $(2q-2)!$ permutations of vertices from $F'\cup B'$, and select the one that gives the shortest walk. In this way, we can build the game graph $\tilde{G}$ and assign weights to its edges.

3.1.1.2 $n^{O(k)}$ Algorithm for SCSS

Recall that initially each of the vertices $t_1,t_2,\dots t_q$ has an F-token and a B-token as well. Finally we want all the F-tokens and all the B-tokens to reach the vertex r via legal moves. This suggests the following algorithm for SCSS.

Algorithm 6: Feldman-Ruhl Algorithm for SCSS

```
1: Construct the game graph $\tilde{G}=(\tilde{V},\tilde{E})$, where $\tilde{E}$ is the set of all legal token moves.
2: Find a minimum weight path $P$ in $\tilde{G}$ from $(\{t_1,t_2,\dots,t_q\},\{t_1,t_2,\dots,t_q\})$ to $(r,r)$.
3: Let $H$ be the union of $\{t_1,t_2,\dots,t_q,r\}$ and all nodes given by $P$ (including those in
   sets $M$ for flipping moves).
4: return $H$
```

To show the correctness, the main idea is that when we move the tokens, we only pay a cost when a new vertex is encountered. The following two lemmas demonstrate the correctness of Algorithm 6:

Lemma 3.1. (Lemma 3.1 from [FR06]) *If there is a move sequence from $(\{t_1,t_2,\dots,t_q\},\{t_1,t_2,\dots,t_q\})$ to (r,r) of c_1-cost c, then there is a solution H for k-SCSS of size $\leq c+q$. Moreover given the move sequence the subgraph H can be easily constructed.*

The proof of Lemma 3.1 follows easily from the definition of the legal moves for the game. The converse statement saying that there is a move sequence corresponding to an optimum solution is more surprising and its proof is more involved.

Lemma 3.2. (Lemma 3.2 from [FR06]) *For any minimal solution H^* to SCSS there is a move sequence from $(\{t_1,t_2,\dots,t_q\},\{t_1,t_2,\dots,t_q\})$ to (r,r) of c_2-cost at most $|H^*|-q$.*

Note that having c_1-cost in Lemma 3.1 (instead of c_2-cost) makes it stronger and having c_2-cost in Lemma 3.2 (instead of c_1-cost) makes it stronger. Lemmas 3.1 and 3.2 together imply that if a move sequence minimizes the c_2-cost, then its c_1-cost is the same as its c_2-cost. Henceforth, we define the cost as the c_2-cost and note that for minimum move sequences the two functions give the same value. It follows that all the flips of a minimum move sequence should have the same cost under both interpretations:

Proposition 3.1. *For every move sequence from $(\{t_1,t_2,\dots,t_q\},\{t_1,t_2,\dots,t_q\})$ to (r,r) having minimum cost, every flip in the move sequence has the following proprety: the walk of minimum length visiting $F'\cup B'$ is a simple path.*

The crucial point in the proof of Lemma 3.2 is that when moving the tokens, we "pay" each time we encounter a new vertex. However, it can happen that we pay twice for a vertex if a token enters the vertex, then leaves it, then later some token visits the vertex again. Feldman and Ruhl are able to avoid this situation by enforcing the following rule:

$\boxed{\text{Once a token moves off a vertex, no other token will ever move to that vertex again (*)}}$

Feldman-Ruhl say that a vertex becomes "dead" once a token moves from it, so that tokens are allowed to only move to vertices in H^* that are "alive." We need to clarify what we mean by "moving off" in a flipping move. We imagine that tokens f and b change places by following the walk, hence we consider all vertices of M becoming dead. However, Feldman and Ruhl do not state explicitly whether or not the original locations of f and b become dead in a flipping move. Observation of [FR06, Claim 3.4] shows that we may make f and b dead (the proof of Claim 3.4 works even in the case when some token f' requires b itself; in fact, the first step of the proof is to conclude that f' requires b). Therefore, we interpret Property (*) in such a way that the locations of f and b also become dead in a flipping move. An important consequence is that a vertex v can participate in at most one flip, as it becomes dead after the first flip and then no other token can move to it with a flip.

For the analysis of the running time of Algorithm 6, we refer to Section 6.1 of [FR06]. Using this algorithm for SCSS as a blackbox, Feldman and Ruhl also design an $n^{O(k)}$ algorithm for DSF. We do no use that algorithm in this thesis, and hence the description is omitted (see Section 5 of [FR06]).

3.1.2 An $2^{O(k\log k)} \cdot n^{O(\sqrt{k})}$ Algorithm for SCSS on Planar Graphs

In this section we design a faster algorithm for the SCSS problem on planar graphs.

Theorem 3.1. *An instance (G,T) of the* STRONGLY CONNECTED STEINER SUBGRAPH *problem with $|G| = n$ and $|T| = k$ can be solved in $2^{O(k\log k)} \cdot n^{O(\sqrt{k})}$ time, when the underlying undirected graph of G is planar (or more generally, H-minor-free for any fixed graph H).*

This algorithm presents a major improvement over Feldman-Ruhl algorithm [FR06] for SCSS in general graphs which runs in $n^{O(k)}$ time (see Section 3.1.1.2). Before describing the algorithm formally, we first give a very high-level intuition. The algorithm of Feldman-Ruhl for SCSS is based on defining a game with $2k$ tokens and costs associated with the moves of the tokens such that the minimum cost of the game is equivalent to the minimum cost of a solution of the SCSS problem; then the minimum cost of the game can be computed by exploring a state space of size $n^{O(k)}$. We slightly generalize this game by introducing *supermoves*, which are sequences of certain types of moves. The generalized game still has a state space of $n^{O(k)}$, but it has the advantage that we can now give a bound of $O(k)$ on the number of supermoves required for the game (such a bound is not possible for the original version of the game). We define a *description* of Feldman-Ruhl game: it is essentially a running commentary of the moves that occur in Feldman-Ruhl game in the sense that we report each move as it happens. As we can bound the length of the description by $O(k)$, we can guess the description (types and order of moves etc.), with the exception of the actual location of the vertices appearing in the description. We then need to map each of the $O(k)$ vertices appearing in the description to an actual vertex of the planar graph; trying all possibilities by brute force would still need $n^{O(k)}$ time. This is the point where planarity comes into play. With each description Γ we associate a graph D_Γ where the edges are added according to the moves in the description. Since the number of supermoves was bounded by $O(k)$, we are able to conclude that there is a description of Feldman-Ruhl game whose associated graph is also planar and has $O(k)$ non-isolated vertices. It is well-known that a planar graph with $O(k)$ vertices has treewidth $O(\sqrt{k})$, hence the treewidth of the graph D_Γ associated with the description is $O(\sqrt{k})$. Therefore, we can use an embedding theorem given in Marx and Klein [KM] to find in time $n^{O(\sqrt{k})}$ a minimum cost mapping of the vertices in the description and obtain a minimum-cost subgraph corresponding to the given description of the Feldman-Ruhl game. Our algorithm uses planarity in a very robust way: the only result on planarity we need is the planar grid minor theorem; we argue that D_Γ is planar by showing that it is a minor of the input graph. This allows transparent generalization to the case when the underlying undirected graph is H-minor-free. All we need to do is to use the grid minor theorem for H-minor-free graphs due to Demaine and Hajiaghayi [DH05b], which implies for any fixed graph H, every H-minor-free graph G has treewidth $O(\sqrt{|V(G)|})$.

3.1.2.1 Another Look at Moves of the Feldman-Ruhl game

In this section, we introduce notation describing the moves of the Feldman-Ruhl game in more detail. This will allow us to prove our $2^{O(k^2)} \cdot n^{O(\sqrt{k})}$ algorithm for SCSS on planar (and more generally H-minor-free) graphs. Recall that the legal moves for SCSS are defined in Section 3.1.1.1.

3.1.2.1.1 The Flipping Move

For ease of notation, we call the Flipping move as Flip(f,b,u,v,F',B'), which is used to denote that the forward token f located at v flips with the backward token b located at v, and the sets of vertices F' and B' that denote the locations of the forward and backward tokens, respectively, are picked up. Let P be the shortest $u \to v$ walk in G which goes through all the vertices where the tokens from $F' \cup B'$ are located. Then the cost of this move is the number of vertices on P which do not have a token from the set $F' \cup B' \cup \{f,b\}$.

We have two cases: either the set $F' \cup B'$ is empty, or not.

- If $F' \cup B' = \emptyset$, then we call this move an **EmptyFlip**(f,b,u,v) move.
- Otherwise $F' \cup B' \neq \emptyset$, and we call this move a **NonEmptyFlip**(f,b,u,v,F',B') move. In particular, we use **NonEmptyFlip**$(f,b,u,v,g_1,g_2,\ldots,g_\ell,w_1,w_2,\ldots,w_\ell)$ to denote the NonEmptyFlip that picks up the tokens g_i at vertex w_i for each $1 \leq i \leq \ell$.

3.1.2.1.2 Single Moves for F and B tokens

We define various types of possible moves of the type Single Move for an F-token. The discussion for B tokens is similar, and we do not repeat it again. For ease of notation, we call the "Single Move for F-token" as Forward(f,u,v) if the forward token f located at u moves forward to the vertex v along the edge (u,v) in this move. Similarly we call the "Single Move for B-token" as Backward(b,u,v) if the backward token b located at vertex u moves to vertex v backward along the edge (v,u) in this move.

For the Forward(f,u,v) move, the cost of this move is 1 if there is a token from $F\cup B$ present on v, and 0 otherwise. We have three cases:

- If there was no token at v, then the cost of this move is 1. We call this move a **SingleForwardAlone**(f,u,v) move since at the end of this move the token originally located at f does not encounter any token.
- If there was no forward token at v, but there was a backward token b, then again the cost of this move is 0. We call this move a **SingleForwardMeet**(f,u,v,b) move since after this move the forward token f meets the backward token b at the vertex v. We follow the convention that every SingleForwardMeet is followed by an EmptyFlip (of length 0) at vertex v; as this does not move the tokens at all, it does not influence the solution. This convention simplifies some of the arguments later in Section 3.1.2.5.
- If there was a forward token f' and a backward token b at v, then the cost of this

move is 0. We call this move a **SingleForwardAbsorbAndMeet**(f,u,v,f',b) move since after this move the forward token f absorbs the forward token f'. However, in this case we do not require an EmptyFlip of length 0 to occur.

- If there was a forward token f' (but no backward) at v, then the cost of this move is 0. We call this move a **SingleForwardAbsorb**(f,u,v,f') move since after this move the forward token f absorbs the forward token f'.

Similarly, we can also define the SingleBackwardAlone move, the SingleBackward-Absorb move, and the SingleBackwardMeet move.

3.1.2.2 A Bird's-eye View of the Feldman-Ruhl game

In this section, we take a bird's-eye view of the Feldman-Ruhl game for SCSS. More formally, we introduce new "supermoves" for their game. The Feldman-Ruhl game takes place in a sequence of moves: each "supermove" is nothing but a collection of contiguous moves from the Feldman-Ruhl game. The advantage is that we are able to show that there is a solution for the Feldman-Ruhl game that can be partitioned into $O(k)$ supermoves, where k is the number of terminals in the SCSS instance. We now define the supermoves. Let H be an optimum solution of the Feldman-Ruhl game satisfying (*), and let the moves in H be $M_1, M_2, \ldots, M_p$.

3.1.2.2.1 Forward, Backward and Flip Supermoves

First, we define the supermoves associated with forward tokens. Let f be a forward token. Consider a contiguous sequence of moves $H_{i_1}, H_{i_2}, \ldots, H_{i_{j-1}}, H_{i_j}$ (with $i_1 < i_2 < \ldots < i_j$) such that

- H_{i_s} is the move SingleForwardAlone(f, v_s, v_{s+1}) for each $1 \leq s \leq i_{j-1}$.
- H_{i_j} is a single move involving f which takes f from v_{i_j} to $v_{i_{j+1}}$
- The only moves between H_{i_1} and H_{i_j} involving f are $H_{i_1}, H_{i_2}, \ldots, H_{i_{j-1}}, H_{i_j}$

Then we can delete the moves $H_{i_1}, H_{i_2}, \ldots, H_{i_{j-1}}$. Depending on the type of the move H_{i_j}, we replace it with a "supermove" M as follows:

- If there was no token at $v_{i_{j+1}}$, then the cost of M is i_j. We call M as the **ForwardAlone**$(f, v_{i_1}, v_{i_{j+1}})$ supermove, and set `involved`$(M) = \{f\}$.
- If there was a backward token b (but no forward token) at the vertex $v_{i_{j+1}}$, then the cost of M is $i_j - 1$. We call this the **ForwardMeet**$(f, v_{i_1}, v_{i_{j+1}}, b)$ supermove, and set `involved`$(M) = \{f, b\}$. For ease of notation, we say that an EmptyFlip (of length 0) occurs at this point, freeing the tokens f, b to move along their subsequent paths.
- If there was both a forward token f' and a backward token b at the vertex $v_{i_{j+1}}$, then the cost of M is $i_j - 1$ and f absorbs the token f'. We call this as the **ForwardAbsorbAndMeet**$(f, v_{i_1}, v_{i_{j+1}}, f', b)$ supermove, and set `involved`$(M) = \{f, f', b\}$. However, in this case we do not require an EmptyFlip of length 0 to occur.

- If there was a forward token f' (but no backward token) at the vertex $v_{i_{j+1}}$, then the cost of M is $i_j - 1$ and f absorbs the token f'. We call this as the **Forward-Absorb**$(f, v_{i_1}, v_{i_{j+1}}, f')$ supermove, and set $\texttt{involved}(M) = \{f, f'\}$.

We also define $\texttt{corner}(M) = \{v_{i_1}, v_{i_{j+1}}\}$ and $\texttt{internal}(M) = \{v_{i_2}, v_{i_3}, \ldots, v_{i_j}\}$.

Similarly, we can also define the BackwardAlone supermoves, the BackwardAbsorb supermoves, the BackwardAbsorbAndMeet and the BackwardMeet supermoves. By Alone supermoves, we refer to the union of BackwardAlone supermoves and the ForwardAlone supermoves. The Absorb supermoves, AbsorbAndMeet and Meet supermoves are also defined similarly.

The NonEmptyFlip moves defined in Section 3.1.2.1.1 are also included in the set of supermoves[1].

If $M = \text{NonEmptyFlip}(f, b, u, v, g_1, g_2, \ldots, g_\ell, w_1, w_2, \ldots, w_\ell)$, then we define

- $\texttt{involved}(M) = \{f, b, g_1, g_2, \ldots, g_\ell\}$,
- $\texttt{corners}(M) = \{u, v\}$, and
- $\texttt{internal}(M) = P \setminus \{u, v, w_1, w_2, \ldots, w_\ell\}$, where P is a shortest $u \to v$ walk in G passing through each w_i.

[1]The NonEmptyFlip is considered both as a move and as a supermove.

3.1.2.2.2 MultipleFlips

Our final supermove is called MultipleFlip. Let H be an optimum solution of the Feldman-Ruhl game satisfying (*), and let the moves in H be $H_1, H_2, \ldots H_\ell$.

Definition 3.1. *Let f and b be forward and backward tokens, respectively. Consider a consecutive sequence of moves $H_i, H_{i+1}, \ldots, H_{j-1}, H_j$ such that*

- *There exists $i \leq s \leq j$ such that H_s is an EmptyFlip involving f and b (potentially of length 0)*
- *For each $i \leq r \leq j$, the move H_r is of one of the following types:*
  - *EmptyFlip move involving f and b.*
  - *SingleForwardAlone move involving f.*
  - *SingleBackwardAlone move involving b.*
  - *SingleForwardMeet or SingleBackwardMeet move involving both f and b.*[2]

Let v_1, w_1 be the initial locations of f, b before H_i occurs are v_1, w_1 respectively. Similarly, let the final locations of f, b after H_i occurs are v_2, w_2 respectively. Then we define M= MultipleFlip$(f, b, v_1, v_2, w_1, w_2)$ as the supermove which is given by the sequence of consecutive moves $H_i, H_{i+1}, \ldots, H_j$. We say that the $H_i, H_{i+1}, \ldots, H_j$ are the components *of M.*

[2]Recall from Section 3.1.2.2.1 that every SingleForwardMeet or SingleBackward Meet move must be followed by an EmptyFlip of length 0.

Note that an EmptyFlip is a special case of a MultipleFlip with just one component which is an EmptyFlip, and also $v_1 = w_2$ and $v_2 = w_1$. For the supermove $M =$ MultipleFlip $(f, b, v_1, v_2, w_1, w_2)$, we define the following sets:

- $\texttt{involved}(M) = \{f, b\}$
- $\texttt{corners}(M) = \{v_1, v_2, w_1, w_2\}$
- $\texttt{internal}(M) = \left(\bigcup_{H \in M} \texttt{corner}(H) \cup \texttt{internal}(H)\right) \setminus \{v_1, v_2, w_1, w_2\}$, where $H \in M$ means that H is a component of the MultipleFlip M.

The following property of a MultipleFlip will be helpful for our algorithm:

Definition 3.2. *Let M be given by MultipleFlip $(f, b, v_1, v_2, w_1, w_2)$. Then $\texttt{corners}(M)$ is given by $\{v_1, v_2, w_1, w_2\}$. We say that M is a* **clean** *MultipleFlip if either*

- $|\texttt{corners}(M)| = 2$*, or*
- $|\texttt{corners}(M)| \geq 3$ *and $\texttt{internal}(M)$ is connected (in the undirected sense), and adjacent to every vertex of $\texttt{corner}(M)$*

Note that if M is an EmptyFlip, then $|\texttt{corners}(M)| = 2$ and it is clean by definition.

3.1.2.2.3 List of all supermoves

The final set of supermoves that we consider are the following:

Final Set of Supermoves

- Alone, Absorb, AbsorbAndMeet and Meet
- NonEmptyFlip
- MultipleFlip

3.1.2.3 Description Associated with a Partition of a Solution to the Feldman-Ruhl Game

Consider a solution H for the Feldman-Ruhl game and a partition $P(H)$ of H into supermoves. Then the description $\Gamma_{P(H)}$ associated with $P(H)$,is essentially a running commentary of the game as it happens, i.e., we list all the supermoves which form the partition $P(H)$.

First there are k entries of the form Location(f_i, b_i, v_i) for $1 \leq i \leq k$ which tell that the initial location of the tokens f_i, b_i is vertex v_i of G. Note that the vertices $v_1, v_2, \ldots, v_k$ are given in the input instance. Then there is a sequence of entries where each entry has one of the following types:

1. ForwardAlone(f, w_1, w_2): The forward token f went from vertex w_1 to w_2 in G and then did not meet any other token at w_2.
2. BackwardAlone(b, w_1, w_2): The backward token b went from vertex w_2 to w_1 in G and then did not meet any other token at w_1.
3. ForwardAbsorb(f_1, w_1, w_2, f_2): The forward token f_1 went from vertex w_1 to w_2 in G and then absorbed another forward token f_2.
4. BackwardAbsorb(b_1, w_2, w_1, b_2): The backward token b_1 went from vertex w_2 to w_1 in G and then absorbed another backward token b_2.
5. ForwardMeet(f, w_1, w_2, b): The forward token f went from w_1 to w_2 in G, and then performed an EmptyFlip (of length 0) with a backward token b at w_2.

6. BackwardMeet(b, w_2, w_1, f): The backward token b went from w_2 to w_1 in G, and then performed an EmptyFlip (of length 0) with a forward token f at w_1.
7. ForwardAbsorbAndMeet(f_1, w_1, w_2, f_2, b): The forward token f_1 went from vertex w_1 to w_2 in G and then absorbed another forward token f_2 at w_2, where a backward token b was also present.
8. BackwardAbsorbAndMeet(b_1, w_2, w_1, b_2, f): The backward token b_1 went from vertex w_2 to w_1 in G and then absorbed another backward token b_2 at w_1, where a forward token f was also present.
9. NonEmptyFlip$(f, b, v_1, v_2, e_1, e_2, \ldots, e_\ell, w_1, w_2, \ldots, w_\ell)$: The tokens f and b were initially located at vertices v_1 and v_2 respectively in G. They then made a NonEmpty flip picking up the tokens e_i which was located at vertex w_i in G along the way, in that order.
10. MultipleFlip$(f, b, v_1, v_2, w_1, w_2)$: The tokens f, b were located initially at vertices v_1, w_1 in G respectively. They then participated in a MultipleFlip and finally were located at vertices v_2, w_2 respectively.

The next theorem is the main combinatorial result that we use in the algorithm. It justifies introducing the supermoves: it shows that there is a solution and a partition of this solution into $O(k)$ supermoves.

Theorem 3.2. *There is an optimum solution H^* of the Feldman-Ruhl game and a partition $P'(H^*)$ of this solution into supermoves such that the total number of entries (i.e., the number of supermoves) in the description of $P'(H^*)$, say X^*_{label}, are $O(k)$. Furthermore,* *every MultipleFlip supermove is clean.*

As the proof of Theorem 3.2 requires a deep analysis of the game, we defer it to Section 3.1.2.5. We observe here the following simple property of an optimum solution:

Lemma 3.3. *Let M, M' be any two supermoves of $P'(H^*)$. Then we have $\texttt{internal}(M) \cap \texttt{internal}(M') = \emptyset$.*

Proof. By definition, each vertex in the set $\texttt{internal}(M)$ is visited by some token. Hence Property (*) implies that if M and M' are any two supermoves then $\texttt{internal}(M) \cap \texttt{internal}(M') \neq \emptyset$.

□

3.1.2.3.1 Unlabeled Descriptions

In this section, we consider *unlabeled* descriptions, i.e., descriptions where we replace the vertices in the descriptions by variables (which will always be denoted by Greek letters). Recall that we have $2k$ tokens. We now show that it is enough to consider $O(k)$ variables to represent the unlabeled descriptions.

Corollary 3.1. *The number of vertices of G (with multiplicities) listed over all entries of the description X^*_{label} is $O(k)$.*

Proof. By Theorem 3.2, we know that the description has $O(k)$ entries. We now refer to Section 3.1.2.3. For the Alone, Absorb, Meet and MultipleFlip type of entries in the description we use a $O(1)$ number of vertices of G per entry. For the NonEmptyFlip case we might add some more vertices in the description (like the $w_1, w_2, \dots, w_\ell$) but their total number is bounded by $2k$, as each such vertex is picked up in the NonEmptyFlip and hence can occur in only one such entry. Therefore, the total number of vertices of G (with multiplicities) listed over all entries of the description X^*_{label} is $O(k)$. □

Our goal is to guess the description X^*_{label}. We will do it as follows: first guess an unlabeled description, and then guess a labeling of the vertices of G to the variables of the unlabeled description. The next lemma bounds the number of distinct unlabeled descriptions having $O(k)$ entries.

Lemma 3.4. *The number of distinct unlabeled descriptions having $O(k)$ entries is* $2^{O(k \log k)}$

Proof. For an unlabeled description, we call each of the following as a *bit* of the description: the names of the supermoves, the listed variables or the listed tokens.

Referring to Section 3.1.2.3, Each supermove (except NonEmptyFlip) contains $O(1)$ variable bits. In addition to two variables corresponding to the endpoints of the flip, each NonEmptyFlip also lists several *internal* variables, each of which corresponds to a token that gets picked up in the NonEmptyFlip. Hence the total number of *internal* variable bits listed by the NonEmptyFlip is at most $2k$. Since the number of NonEmptyFlips is upper bounded by the total number of supermoves, which is $O(k)$, the number of non-internal variable bits listed by NonEmptyFlips is also upper bounded by $2 \times O(k) = O(k)$. Hence the total number of variable bits is $O(k) + 2k + O(k) = O(k)$. By Corollary 3.1, it is enough to consider only $O(k)$ variables in our unlabeled descriptions. Hence, the total number of guesses for the variable bits is $k^{O(k)}$.

We have $O(1) = 10$ choices for the type of the supermove. Since we want to enumerate only unlabeled descriptions with $O(k)$ entries, the number of choices for this is $10^{O(k)}$. Each supermove (except NonEmptyFlip) lists at most 3 tokens. Any non-internal token listed in a NonEmptyFlip does not appear in any other supermove, and hence their number is upper bounded by the total number of tokens which is $2k$. Also we consider only unlabeled descriptions with $O(k)$ entries. Hence the total number of token bits is $O(k)$. Since there are $2k$ tokens, the number of choices for the token bits is $(2k)^{O(k)}$.

Therefore, the total number of distinct unlabeled descriptions with $O(k)$ entries is $10^{O(k)} \times (2k)^{O(k)} \times k^{O(k)} = 2^{O(k \log k)}$.

□

Let the set of all unlabeled descriptions be $\mathcal{X}$. It is easy to see that we can enumerate all elements of $\mathcal{X}$ in time $|\mathcal{X}| = 2^{O(k \log k)}$. We now show how to check if an unlabeled description $X \in \mathcal{X}$ is *valid* or not:

Definition 3.3. *Consider an unlabeled description $X \in \mathcal{X}$. We say that X is* valid *if the following holds:*

- *The first k entries of X are given by Location(f_i, b_i, α_i) for $i \in [k]$ such that $\alpha_i \neq \alpha_j$ for each $i \neq j$.*
- *For every token f, the variables assigned to f in its current supermove is the same variable that ended up being assigned to f at the end of the last supermove in X involving f (if it exists).*
- *Any token which is absorbed (in an Absorb or AbsorbAndMeet supermove) or*

picked up (in a NonEmptyFlip) supermove cannot be involved in any subsequent move in X.

- *At the end of all the supermoves in X, all the alive tokens are assigned to the variable α_k.*

Given an unlabeled description $X \in \mathcal{X}$, it is easy to see that we can check whether X is valid in $O(k)$ time by simply keeping a list of alive tokens are their currently assigned variables. Hence, in $2^{O(k \log k)}$ time, we can build the set $\mathcal{X}'$ of valid unlabeled descriptions.

3.1.2.3.2 Directed Graphs Associated with Descriptions

With each valid unlabeled description $X \in \mathcal{X}'$, we can associate a directed graph $D_X = (V_E, E_X)$. The vertex set V_X contain all the variables listed in X, plus at most one additional variable for each MultipleFlip. By Theorem 3.2 and Corollary 3.1, we have that $|V_X| = O(k)$. The edge set E_X is defined as follows:

The edge set E_X for the digraph D_X corresponding to a valid unlabeled description X

1. ForwardAlone(f,α,α'): Add the edge (α,α').
2. BackwardAlone(b,α,α'): Add the edge (α',α).
3. ForwardAbsorb(f_1,α,α',f_2): Add the edge (α,α').
4. BackwardAbsorb(b_1,α,α',b_2): Add the edge (α',α).
5. ForwardMeet(f,α,α',b): Add the edge (α,α').
6. BackwardMeet(b,α,α',f): Add the edge (α',α).
7. ForwardAbsorbAndMeet(f,α,α',f',b): Add the edge (α,α').
8. BackwardAbsorbAndMeet(b,α,α',b',f): Add the edge (α',α).
9. NonEmptyFlip$(f,b,\alpha,\alpha',e_1,e_2,\ldots,e_\ell,\gamma_1,\gamma_2,\ldots,\gamma_\ell)$: Add the path $\alpha \to \gamma_1 \to \gamma_2 \to \ldots \to \gamma_\ell \to \alpha')$.
10. MultipleFlip$(f,b,\alpha,\alpha',\gamma,\gamma')$: Let L be the (multi)set $\{\alpha,\alpha',\gamma,\gamma'\}$
    - If $|L|=2$ then we know that $\alpha=\alpha'$ and $\gamma=\gamma'$ cannot occur since in this case both tokens do not move at all. So the only two cases are:
      - If $\alpha=\gamma$ and $\alpha'=\gamma'$, then add the edge (α,α') and color it **red**.
      - If $\alpha=\gamma'$ and $\alpha'=\gamma$, then add the edge (α,α') and color it **blue**.
    - If $|L|\geq 3$ then introduce a new vertex δ, and add the edges $E(\delta,1)=(\delta,\alpha), E(\delta,2)=(\delta,\alpha'), E(\delta,3)=(\delta,\gamma)$ and $E(\delta,4)=(\delta,\gamma')$

By the way we defined this directed graph, if the description corresponds to a solution in a graph G, then the graph of the description is a minor of G (and in particular, it is planar if G is planar).

Theorem 3.3. *Let X^*_{label} be as in Theorem 3.2 and let X^* be the corresponding unlabeled description. The underlying undirected graph of the directed graph D_{X^*} is a minor of the underlying undirected graph of G.*

Proof. We construct an undirected graph G' from the underlying undirected graph of G the following way. For every supermove, we do the following:

- In the first 8 cases above, there are two corner vertices. Either the two corner vertices are adjacent in G, or the internal vertices of the supermove give a path between them. In the latter case, we contract this path to make the two corner vertices adjacent.

- In the case of a NonEmptyFlip, by Proposition 3.1, there is a simple $v_1 \to w_1 \to \dots w_\ell \to v_2$ path on the internal vertices of the supermove. Then we contract subpaths of this path to make $v_1 w_1 \dots w_\ell v_2$ a path (i.e., to make these vertices adjacent).
- In the case of a MultipleFlip with two corner vertices, there is a path on the internal vertices between the two corner vertices (note that the case $v_1 = v_2$ and $w_1 = w_2$ need not be considered, since then the two tokens do not move at all). As in the first case, we contract this path to make the two corners adjacent.
- In the case of a MultipleFlip with at least three corner vertices, Theorem 3.3 implies that this MultipleFlip is clean, that is, the internal vertices induce a connected graph that is adjacent to all corners. Then we contract the internal vertices to a single vertex.

By Lemma 3.3, no two supermoves of $P'(H^*)$ share any `internal` vertex, thus these contractions are independent. It is easy to see now that the underlying undirected graph of D_{X^*} is a subgraph of G'. In particular, for every MultipleFlip with at least three corner vertices, the newly introduced vertex δ can be mapped to the vertex obtained by contracting the internal vertices of the supermove. $\square$

Since $|V_X| = O(k)$, a result of Demaine and Hajiaghayi [DH05b] implies that the treewidth of the underlying undirected graph of D_{X^*} is $O(\sqrt{k})$. Therefore, for every valid unlabeled description $X \in \mathcal{X}'$, we check if the treewidth of the underlying undirected graph of D_X is $O(\sqrt{k})$ by using the constant factor approximation algorithm of Bodlaender et al. [BDD+13], which runs in $2^{O(\sqrt{k})} \cdot k$ time. Discard all those unlabeled descriptions $X \in \mathcal{X}'$ for which this does not hold, and let $\mathcal{X}''$ be the resulting set of unlabeled descriptions. Note that we can construct $\mathcal{X}''$ in $|\mathcal{X}'| \times 2^{O(\sqrt{k})} \times k = 2^{O(k \log k)}$ time.

3.1.2.4 Guessing a Labeling for an Unlabeled Description Using Dynamic Programming

For each valid unlabeled description $X \in \mathcal{X}''$, the digraph D_X comes with k special variables, say $\alpha_1, \alpha_2, \ldots, \alpha_k$, that must be mapped to the vertices $v_1, v_2, \ldots, v_k$ where the terminals are placed in G. We try to map the remaining vertices of D_X to elements of $U = V \cup V^4$ so that the unlabelled description coincides with X^*_{label}.

For this purpose, we use the following theorem due to Klein and Marx [KM]:

Theorem 3.4. *Let D be a directed graph, U a set of elements, and functions $\texttt{cv} : V(D) \times U \to \mathbb{Z}^+ \cup \{\infty\}$, $\texttt{ce} : V(D) \times V(D) \times U \times U \to \mathbb{Z}^+ \cup \infty$. In time $|U|^{O(tw(D))}$ we can find a mapping $\phi : V(D) \to U$ that minimizes*

$$B_\phi = \sum_{v \in V(D)} \texttt{cv}(v, \phi(v)) + \sum_{(u,v) \in E(D)} \texttt{ce}(u, v, \phi(u), \phi(v))$$

where $tw(D)$ denotes the treewidth of the underlying undirected graph of D.

Recall that each $X \in \mathcal{X}''$ has treewidth $O(\sqrt{k})$. Note that $|U| = n^{O(1)}$, and hence for any choice of functions $\texttt{ce}$ and $\texttt{cv}$ we will be able to compute the minimum mapping ϕ in time $n^{O(\sqrt{k})}$. Our goal is to now apply Theorem 3.4 for the graph D_X for each $X \in \mathcal{X}''$, and define the functions $\texttt{ce}$ and $\texttt{cv}$ in a way such that the objective value of Theorem 3.4 exactly captures the cost of the labeled description X_{label} obtained by replacing each variable α by the vertex $\phi(\alpha)$.

3.1.2.4.1 Defining the Functions ce and cv

First we see how to compute the minimum cost of a MultipleFlip, since we need it in the $\texttt{cv}$ function. Let $v_1, v_2, \ldots, v_k$ be the vertices of G which have the k terminals of the SCSS instance.

Lemma 3.5. *The minimum cost of MultipleFlip(f_i, b_j) can be found in polynomial time.*

Proof. Let the initial locations of f_i, b_j be u_i, u_j and the final locations be v_i, v_j respectively. We first build a game graph $\tilde{G}$ where the vertex set is $V \times V$. Then we add the weights between the edges similar to Section 6.1 of the Feldman-Ruhl paper [FR06]. Since all the flips in between are empty flips, their cost is just the shortest paths in G. Then we find a shortest path in the game graph $\tilde{G}$ from (u_i, v_i) to (u_j, v_j). □

We now define the $\texttt{cv}$ and $\texttt{ce}$ functions below:

The Function $cv : V(D_X) \times U \to \mathbb{Z}^+ \cup \{\infty\}$

- For $u \in V$ we define

$$\mathtt{cv}(\alpha, u) = \begin{cases} 1 & \text{if } \alpha = \alpha_i \text{ and } u = v_i \text{ for some } i \in [k] \\ \infty & \text{if } \alpha = \alpha_i \text{ and } u \neq v_i \text{ for some } i \in [k] \\ 1 & \text{otherwise} \end{cases}$$

- For $u = (v_1, v_2, w_1, w_2) \in V^4$, let L be the multiset $\{v_1, v_2, w_1, w_2\}$.
  - If $|L| \geq 3$, then define $\mathtt{cv}(\alpha, u) =$ (cost of MultipleFlip(v_1, v_2, w_1, w_2)) $- |X|$, where X is the multiset $= \{v_1, v_2, w_1, w_2\} \setminus \{v_1, w_1\}$
  - Otherwise if $|L| = 2$, then define $cv(*, u) = \infty$ where $*$ denotes any variable in D_X

If $u \in V$, then we make sure that the cost is infinity if a "marked" vertex in D_X is not mapped to the correct vertex having a terminal in G. For all other vertices in D they get a cost of one to be assigned to other vertices in G. If $u \in V^4$, then the cost of mapping $\alpha \in D_X$ to $u = (u_1, u_2, u_3, u_4)$ is the cost of the MultipleFlip between (u_1, u_3) and (u_2, u_4).

The Function $ce : V(D_X) \times V(D_X) \times U \times U \to \mathbb{Z}^+ \cup \infty$

If $(\alpha, \alpha') \notin E(D_X)$ define $\texttt{ce}(\alpha, \alpha', *, *) = \infty$, where $*$ denote any element of U. In the remaining cases below, we assume that $(\alpha, \alpha') \in E(D_X)$.

- For $v, w \in V$
  - If (α, α') is a **red** edge in D_X, then define $\texttt{ce}(\alpha, \alpha', v, w) =$ [cost of the MultipleFlip (v, w, v, w)] $\cdot \texttt{cv}(\alpha, v) \cdot \texttt{cv}(\alpha', w)$
  - If (α, α') is a **blue** edge in D_X, then define $\texttt{ce}(\alpha, \alpha', v, w) =$ [cost of the MultipleFlip (v, w, w, v)] $\cdot \texttt{cv}(\alpha, v) \cdot \texttt{cv}(\alpha', w)$
  - If (α, α') is a edge in D_X with no color, then define $\texttt{ce}(\alpha, \alpha', v, w) = (d_G(v, w) - 1) \cdot \texttt{cv}(\alpha, v) \cdot \texttt{cv}(\alpha', w)$, where $d_G(v, w)$ is the length of the shortest $v \rightsquigarrow w$ path in G.
- For $y = (v_1, v_2, w_1, w_2) \in V^4$ and $x \in V$, let L be the multiset $\{v_1, v_2, w_1, w_2\}$.
  - If $|L| = 2$, then define $\texttt{ce}(*, *, y, x) = \infty$, where $*$ denotes any variable from D_X.
  - If $|L| \geq 3$, then define

$$\texttt{ce}(\alpha, \alpha', y, x) = \begin{cases} 0 & \text{if } x = v_1 \text{ and } (\alpha, \alpha') = E(\alpha, 1) \\ 0 & \text{if } x = v_2 \text{ and } (\alpha, \alpha') = E(\alpha, 2) \\ 0 & \text{if } x = w_1 \text{ and } (\alpha, \alpha') = E(\alpha, 3) \\ 0 & \text{if } x = w_2 \text{ and } (\alpha, \alpha') = E(\alpha, 4) \\ \infty & \text{otherwise} \end{cases}$$

- All others are set to ∞

Recall that we defined a special optimal solution H^* in Theorem 3.2.

Lemma 3.6. *If the cost of the solution H^* for the Feldman-Ruhl game is C, then there is an unlabeled description $X^* \in \mathcal{X}''$ and a mapping $\phi : V(D_{X^*}) \to U$ such that $B_\phi = C$*

Proof. Consider the (labeled) description X^*_{label} corresponding to the solution H^* and partition $P'(H^*)$ given in Theorem 3.2. Consider the corresponding unlabeled description X^* obtained from X^*_{label} by the following procedure: for each vertex v listed in X^*_{label} , replace all its occurrences in X^*_{label} by the same variable (but use distinct variables for distinct vertices). Let us denote the variable which we used to replace v by $\phi'(v)$. By

Theorem 3.3, we have $X^* \in \mathcal{X}''$.

Consider the mapping $\phi : V(D_{X^*}) \to U$ given by $\phi(\alpha) = v$ if and only if $\phi'(v) = \alpha$. This is well-defined since every variable in VD_{X^*} has an unique preimage w.r.t the mapping ϕ'. We know that the sum of the costs of the supermoves $P'(H^*)$ is OPT. We now show that for each supermove M, its contribution to the cost of H^* is the same as the sum of the $\texttt{ce}$ and $\texttt{cv}$ contributions to B_ϕ of the vertices involved in M. Recall that in H^*, we pay for a vertex the first (and only) time we encounter it. Also the $\texttt{cv}$ function pays for a variable the first time it is encountered. Initially, we have to pay a cost of 1 for each of the vertices $v_1, v_2, \ldots, v_{k-1}$ which are the locations of $k-1$ of the tokens: This is paid for by the $\texttt{cv}(\phi'(v_i))$ costs. We have the following choices for M:

1. M= ForwardAlone(f, w_1, w_2): In D_{X^*} we have the edge (β_1, β_2) such that $\phi'(w_i) = \beta_i$ for $i \in [2]$. Hence $\phi(\beta_i) = w_i$ for each $i \in [2]$. Let d be the length of the shortest $w_1 \rightsquigarrow w_2$ path in G. Since it is an Alone supermove the vertex w_2 is being encountered for the first time, but w_1 has been encountered before. Hence the contribution of M to H^* is d. The $\texttt{ce}(\beta_1, \beta_2, w_1, w_2)$ cost is equal to $(d-1) \cdot \texttt{cv}(\beta_1, w_1) \cdot \texttt{cv}(\beta_2, w_2) = (d-1)$. Also $\texttt{cv}(\beta_2, w_2) = 1$ is paid for in B_ϕ since this is the first time we encounter the variable β_2. Hence, the contribution to cost of H^* and to B_ϕ is exactly d.
2. M= BackwardAlone(b, w_1, w_1): This case can be handled similar to the above case.
3. M= ForwardAbsorb(f, w_1, w_2, f'): In D_{X^*} we have the edge (β_1, β_2) such that $\phi'(w_i) = \beta_i$ for $i \in [2]$. Hence $\phi(\beta_i) = w_i$ for each $i \in [2]$. Let d be the length

of the shortest $w_1 \rightsquigarrow w_2$ path in G. Since it is an Absorb supermove, both w_1 or w_2 have been encountered before. Hence the contribution of M to H^* is $d-1$. The $\texttt{ce}(\beta_1, \beta_2, w_1, w_2)$ cost is equal to $(d-1) \cdot \texttt{cv}(\beta_1, w_1) \cdot \texttt{cv}(\beta_2, w_2) = (d-1)$. Also the $\texttt{cv}$ function does not pay for the variables β_1 and β_2 since we have encountered them before. Hence, the contribution to cost of H^* and to B_ϕ is exactly $d-1$.

4. M= BackwardAbsorb(b, w_1, w_2, b'): This case can be handled similar to Case 3.

5. M= ForwardMeet(f, w_1, w_2, b): This case can be handled similar to Case 3.

6. M= BackwardMeet(b, w_1, w_2, f): This case can be handled similar to Case 3.

7. M= ForwardAbsorbAndMeet(f, w_1, w_2, f', b): This case can be handled similar to Case 3.

8. M= BackwardAbsorbAndMeet(b, w_1, w_2, b', f): This case can be handled similar to Case 3.

9. M= NonEmptyFlip$(f, b, w_1, w_2, e_1, e_2, \ldots, e_\ell, x_1, x_2, \ldots, x_\ell)$:In D_{X^*} we have the path $\beta_1 \to \gamma_1 \to \gamma_2 \to \ldots \to \gamma_\ell \to \beta_2$ such that $\phi'(w_i) = \beta_i$ for each $i \in [2]$ and $\phi'(x_j) = \gamma_j$ for each $j \in [\ell]$. Note that each of the vertices $w_1, w_2, x_1, \ldots, x_\ell$ has a token on them, and hence have been encountered before. For the sake of notation, let us denote w_1, w_2 by $x_0, x_{\ell+1}$ respectively and β_1, β_2 by $\gamma_0, \gamma_{\ell+1}$ respectively . For each $0 \leq i \leq \ell$ let the length of shortest $x_i \rightsquigarrow x_{i+1}$ path in G be d_i. Then the contribution of M to the cost of H^* is $(\sum_{i=0}^{\ell} d_i) - (\ell+1)$. For each $0 \leq i \leq \ell$, the $\texttt{ce}(\gamma_i, \gamma_{i+1}, x_i, x_{i+1})$ cost is equal to $(d_i - 1) \cdot \texttt{cv}(\gamma_i, x_i) \cdot \texttt{cv}(\gamma_{i+1}, x_{i+1}) = (d_i - 1)$. Hence the total summation of the $\texttt{ce}$ costs is also $(\sum_{i=0}^{\ell} d_i) - (\ell+1)$. Note that we do not pay any $\texttt{cv}$ costs

since all the variables $\gamma_0, \gamma_1, \ldots, \gamma_\ell, \gamma_{\ell+1}$ have been encountered before.

10. M is a MultipleFlip with $|\texttt{corners}(M)| = 2$. Let $\texttt{corners}(M) = \{v, w\}$. Then we know that both v, w have been encountered before. Hence the `cv` cost is 0, while the `ce` cost for (either the blue or red) edge is exactly equal to the cost of M.

11. M is a MultipleFlip with $|\texttt{corners}(M)| \geq 3$. Let the multiset $\texttt{corners}(M) = \{v_1, w_1, v_2, w_2\}$, and define X to be the multiset $\{v_1, v_2, w_1, w_2\} \setminus \{v_1, w_1\}$. Hence X is exactly the subset of $\texttt{corners}(M)$ that is first discovered by M, and so we pay the `cv` cost for each vertex in X for a total cost of $|X|$. In B_ϕ, we do not pay for the `cv` costs of v_1 or w_1 since they have been encountered before (as they had tokens at the start of M). Here the `ce` cost for the edges is 0, while the `cv` cost for the newly created variable δ is exactly equal to (cost of M)$-|X|$. Hence the contribution of M to B_ϕ is exactly equal to cost of M.

All the above cases together imply that $B_\phi = OPT$.

□

Lemma 3.7. *If there is a valid unlabeled description $X \in \mathcal{X}''$ and a mapping $\phi : V(D_X) \to U$ such that $B_\phi = R$ for some $R < \infty$, then there is a solution H for the Feldman-Ruhl game of cost exactly R.*

Proof. From the unlabeled description X, we first obtain a labeled description X_{label} by replacing each variable $\alpha \in V(D_X)$ by the vertex $\phi(\alpha) \in U$. Since X is a valid unlabeled descriptio, it follows that X_{label} is the description associated with $(H, P(H))$, where H is a solution of the Feldman-Ruhl game and $P(H)$ is a partition of H into supermoves. We now show that the cost of H is at most $B_\phi = R$.

For each supermove $M \in P(H)$, we add some edge(s) to the directed graph D_X (see Section 3.1.2.3.2). In fact, if M was a MultipleFlip with $|\texttt{corners}(M)| \geq 3$, then we also added a new vertex δ and some edges incident to it. Recall that in B_ϕ, we pay the $\texttt{cv}$ cost for a variable the first time we encounter it. We have the following cases for M:

1. M= ForwardAlone(f, w_1, w_2): In D_X we have the edge (β_1, β_2) such that $\phi(\beta_i) = w_i$ for each $i \in [2]$. Let d be the length of the shortest $w_1 \rightsquigarrow w_2$ path in G. Since it is an Alone supermove the vertex w_2 is being encountered for the first time, but w_1 has been encountered before. Hence the contribution of M to H^* is d. The $\texttt{ce}(\beta_1, \beta_2, w_1, w_2)$ cost is equal to $(d-1) \cdot \texttt{cv}(\beta_1, w_1) \cdot \texttt{cv}(\beta_2, w_2) = (d-1)$. Also we pay the cost $\texttt{cv}(\beta_2, w_2) = 1$ in B_ϕ since this is the first time we encounter the variable β_2. Hence, the contribution to cost of H^* and to B_ϕ is exactly d.
2. M= BackwardAlone(b, w_1, w_1): This case can be handled similar to the above case.
3. M= ForwardAbsorb(f, w_1, w_2, f'): In D_{X^*} we have the edge (β_1, β_2) such that $\phi(\beta_i) = w_i$ for each $i \in [2]$. Let d be the length of the shortest $w_1 \to w_2$ path in G. Since it is an Absorb supermove, both w_1 or w_2 have been encountered before. Hence the contribution of M to H^* is $d-1$. The $\texttt{ce}(\beta_1, \beta_2, w_1, w_2)$ cost is equal to $(d-1) \cdot \texttt{cv}(\beta_1, w_1) \cdot \texttt{cv}(\beta_2, w_2) = (d-1)$. Also the $\texttt{cv}$ function does not pay for the variables β_1 and β_2 since we have encountered them before. Hence, the contribution to cost of H^* and to B_ϕ is exactly $d-1$.

4. M= BackwardAbsorb(b, w_1, w_2, b'): This case can be handled similar to Case 3.

5. M= ForwardMeet(f, w_1, w_2, b): This case can be handled similar to Case 3.

6. M= BackwardMeet(b, w_1, w_2, f): This case can be handled similar to Case 3.

7. M= ForwardAbsorbAndMeet(f, w_1, w_2, f', b): This case can be handled similar to Case 3.

8. M= BackwardAbsorbAndMeet(b, w_1, w_2, b', f): This case can be handled similar to Case 3.

9. M= NonEmptyFlip$(f, b, w_1, w_2, e_1, e_2, \ldots, e_\ell, x_1, x_2, \ldots, x_\ell)$:In D_{X^*} we have the path $\beta_1 \to \gamma_1 \to \gamma_2 \to \ldots \to \gamma_\ell \to \beta_2$ such that $\phi(\beta_i) = w_i$ for each $i \in [2]$ and $\phi(\gamma_j) = x_j$ for each $j \in [\ell]$. Note that each of the vertices $w_1, w_2, x_1, \ldots, x_\ell$ has a token on them, and hence have been encountered before. For the sake of notation, let us denote w_1, w_2 by $x_0, x_{\ell+1}$ respectively and β_1, β_2 by $\gamma_0, \gamma_{\ell+1}$ respectively . For each $0 \leq i \leq \ell$ let the length of shortest $x_i \rightsquigarrow x_{i+1}$ path in G be d_i. Then the contribution of M to the cost of H^* is $(\sum_{i=0}^{\ell} d_i) - (\ell+1)$. For each $0 \leq i \leq \ell$, the $\mathtt{ce}(\gamma_i, \gamma_{i+1}, x_i, x_{i+1})$ cost is equal to $(d_i - 1) \cdot \mathtt{cv}(\gamma_i, x_i) \cdot \mathtt{cv}(\gamma_{i+1}, x_{i+1}) = (d_i - 1)$. Hence the total summation of the $\mathtt{ce}$ costs is also $(\sum_{i=0}^{\ell} d_i) - (\ell+1)$. Note that we do not pay any $\mathtt{cv}$ costs since all the variables $\gamma_0, \gamma_1, \ldots, \gamma_\ell, \gamma_{\ell+1}$ have been encountered before.

10. M is a MultipleFlip with $|\mathtt{corners}(M)| = 2$. Let $\mathtt{corners}(M) = \{v, w\}$. Then we know that both v, w have been encountered before. Hence the $\mathtt{cv}$ cost is 0, while the $\mathtt{ce}$ cost for (either the blue or red) edge is exactly equal to the cost of M.

11. M is a MultipleFlip with $|\mathtt{corners}(M)| \geq 3$. Let the multiset $\mathtt{corners}(M) =$

$\{v_1, w_1, v_2, w_2\}$, and define X to be the multiset $\{v_1, v_2, w_1, w_2\} \setminus \{v_1, w_1\}$. Hence X is exactly the subset of $\texttt{corners}(M)$ that is first discovered by M, and so we pay the $\texttt{cv}$ cost for each vertex in X for a total cost of $|X|$. In B_ϕ, we do not pay for the $\texttt{cv}$ costs of v_1 or w_1 since they have been encountered before (as they had tokens at the start of M). Here the $\texttt{ce}$ cost for the edges is 0, while the $\texttt{cv}$ cost for the newly created variable δ is exactly equal to (cost of M)$-|X|$. Hence the contribution of M to B_ϕ is exactly equal to cost of M.

□

Lemma 3.6 and Lemma 3.7 give the correctness of the following algorithm:

Algorithm 7: Computing minimum cost mapping

```
1: Enumerate the set $\mathcal{X}''$
2: for each $X \in \mathcal{X}''$ do
3:     Apply Theorem 3.4 to the graph $D_X$ to get the minimum mapping say $\phi_X$ which
       gives cost $C_X$.
4: Output $\min_{X \in \mathcal{X}''}\{C_X\}$
```

We now analyze the running time. As seen before in Section 3.1.2.3.2, we can compute the set $\mathcal{X}''$ in $2^{O(k \log k)}$ time. For each $X \in \mathcal{X}''$, we can create the graph D_X in $O(k^2)$ time, since it has $O(k)$ vertices. Finally, applying Theorem 3.4 to D_X takes $n^{O(\sqrt{k})}$ time. Hence, the total running time of the algorithm is $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$. This concludes the proof of Theorem 3.1.

3.1.2.5 Bounding the Number of Supermoves: Proof of Theorem 3.2

Consider an optimum solution H of the Feldman-Ruhl game. Let the moves that constitute the solution H be $H_1, H_2, \ldots, H_\ell$, and the number of alive tokens in H after move H_i be t_i for each $i \in [\ell]$. We assign to H the tuple $\lambda(H) = (t_1, t_2, \ldots, t_\ell)$. Let $\preceq_{\texttt{lex}}$ be the standard lexicographic order over tuples of non-negative integers. Now we define an ordering over the solutions of the Feldman-Ruhl game as follows:

Definition 3.4. *Let H, H' be two optimum solutions of the Feldman-Ruhl game. Then we define $H \preceq_{FR} H'$ if $\lambda(H) \preceq_{\texttt{lex}} \lambda(H')$.*

Any solution H of the Feldman-Ruhl game can be partitioned into contiguous sequences of supermoves in various ways.

We fix a pair $(H^*, P(H^*))$, where H^* is an optimum solution for the Feldman-Ruhl game and $P(H^*)$ is a partition of H^* into supermoves, satisfying the following additional property:

H^* is an optimal solution satisfying property (*) and among such solutions, minimal with respect to the ordering $\preceq_{FR}$. Among all partitions of H^* into supermoves, let $P(H^*)$ be a partition that has the minimum number of supermoves. (**)

We now show that the $|P(H^*)| = O(k)$. Recall from Section 3.1.1.1 that if two tokens of same type meet at a common vertex, then we merge them into one token.

Lemma 3.8. *The total number of Absorb supermoves plus AbsorbAndMeet supermoves in $P(H^*)$ is $O(k)$.*

Proof. Each Absorb supermove or AbsorbAndMeet supermove reduces the number of tokens, and hence the total number of such supermoves is $O(k)$. □

By definition, a NonEmptyFlip supermove picks up at least one token along the way, and these tokens no longer need to be considered.

Lemma 3.9. *The total number of NonEmptyFlip supermoves in $P(H^*)$ is $O(k)$.*

Proof. Each NonEmptyFlip supermove reduces the number of tokens, and hence the total number of such supermoves is $O(k)$. □

Next we bound the number of Alone supermoves in the special partition $P(H^*)$.

Lemma 3.10. *The total number of Alone supermoves in $P(H^*)$ is $O(k)$.*

Proof. The proof is by a charging argument: we charge each Alone supermove to either a token or to a NonEmptyFlip supermove.

By symmetry, it is enough to prove the lemma only for the ForwardAlone supermoves. Consider a ForwardAlone supermove $M = \text{ForwardAlone}(f, v_1, v_2)$ in $P(H^*)$ involving a forward token f.

If there is no later supermove in $P(H^*)$ involving f, then we can charge this ForwardAlone move to the token f. Otherwise, let M' be the next supermove after M in $P(H^*)$ which involves f . We have the following cases corresponding to the different possibilities for M'.

1. $M' =$ **ForwardAlone**(f, x_1, x_2)**.** Since M' is the first supermove after M in $P(H^*)$ that involves f, we have $x_1 = v_2$. Delete the supermove M, and replace M' by the

supermove $M'' = \text{ForwardAlone}(f, v_1, x_2)$ to get a new partition $P'(H^*)$.

2. M' = **ForwardAbsorb**(f, x_1, x_2, f')**.** Since M' is the first supermove after M in $P(H^*)$ that involves f, we have $x_1 = v_2$. Delete the supermove M, and replace M' by the supermove $M'' = \text{ForwardAbsorb}(f, v_1, x_2, f')$ to get a new partition $P'(H^*)$.

3. M' = **ForwardMeet**(f, x_1, x_2, b)**.** Since M' is the first supermove after M in $P(H^*)$ that involves f, we have $x_1 = v_2$. Delete the supermove M, and replace M' by the supermove $M'' = \text{ForwardMeet}(f, v_1, x_2, b)$ to get a new partition $P'(H^*)$.

4. M' = **ForwardAbsorbAndMeet**(f, x_1, x_2, f', b)**.** Since M' is the first supermove after M in $P(H^*)$ that involves f, we have $x_1 = v_2$. Delete the supermove M, and replace M' by the supermove $M'' = \text{ForwardAbsorbAndMeet}(f, v_1, x_2, f', b)$ to get a new partition $P'(H^*)$.

5. M' = **MultipleFlip**$(f, b, x_1, x_2, y_1, y_2)$**.** Since M' is the first supermove after M in $P(H^*)$ that involves f, we have $v_2 = x_1$. Delete the supermove M, and replace M' by the supermove $M'' = \text{MultipleFlip}(f, b, v_1, x_2, y_1, y_2)$ to get a new partition $P'(H^*)$.

6. M' **is a NonEmptyFlip**$(fb, , x_1, x_2, g_1, g_2, \ldots, g_\ell, w_1, w_2, \ldots, w_\ell)$**.** In this case, charge M to M'.

To summarize, in each of Cases 1–5 we can construct another partition $P'(H^*)$ that has one less supermove than $P(H^*)$. We now show that $P'(H^*)$ is indeed a valid partition of H^* into supermoves, which contradicts Property (**). Note that the only token whose location is different between the supermoves M and M' is f. But since M' is the first supermove after M in $P(H^*)$ which involves f, we do not create any inconsistencies and do not change the cost of any move between T and T'. Moreover, M'' transports f from its original location before M happened at time T to its final location after M' happened at time T'. Therefore, Cases 1–5 cannot occur.

In Case 6, we can charge the ForwardAlone supermove involving f in $P(H^*)$ to the next supermove in $P(H^*)$, which happens to be a NonEmptyFlip supermove involving f. Note that at most two Alone supermoves are charged to a NonEmptyFlip move of tokens f and b: the last Alone supermoves of f and b before the NonEmptyFlip supermove. There is one initial case when there is no supermove involving f after M in $P(H^*)$: in this case we charge M to the token f. The total number of tokens is $O(k)$, and Lemma 3.9 bounds the number of NonEmptyFlips in $P(H^*)$ to $O(k)$. Hence the total number of ForwardAlone supermoves in $P(H^*)$ is $O(k)$. Similarly, it can be shown that the number of BackwardAlone supermoves in $P(H^*)$ is $O(k)$.

□

Lemma 3.11. *There are no Meet supermoves in $P(H^*)$.*

Proof. Consider a Meet supermove $M = \text{Meet}(f, v_1, v_2, b)$ in $P(H^*)$ involving a forward token f. By our convention, an Meet supermove is always followed by an EmptyFlip of length 0 between tokens f and b; let $E = \text{EmptyFlip}(f, b, v_2, v_2)$ be this move. Let M' be the supermove in $P(H^*)$ that E belongs to. We have the following two cases for M':

1. $\underline{M' = M}$: Delete the supermove M, and replace M' by $M'' = \text{MultipleFlip}(f, b, v_1, v_2, v_2, v_2)$ to get a new partition say $P'(H^*)$.

2. $\underline{M' \neq M}$: In this case, E is an EmptyFlip of length 0 which is neither a supermove in $P(H*)$ nor is it a supermove by itself. Hence M' must be a MultipleFlip with E as its first component, and say $M' = \text{MultipleFlip}(f,b,x_1,x_2,y_1,y_2)$. Also E being the first component of the MultipleFlip M' implies $x_1 = v_2 = y_1$. Delete the supermove M, and replace M' by $M' = \text{MultipleFlip}(f,b,v_1,x_2,v_2,y_2)$ to get a new partition say $P'(H^*)$.

In both the cases, we were able to construct another partition $P'(H^*)$ into supermoves which has one less supermove than $P(H^*)$: by replacing M and M' by a single supermove M''. Note that the next supermove after M in the partition $P(H^*)$ is M'' (since M' occurs immediately after M), and hence we do not change (or reorder) any other supermoves of the partition. $\square$

3.1.2.5.1 Removable Tokens

The following notion will be needed in proofs modifying the solution.

Definition 3.5. *Consider a solution H of the Feldman-Ruhl game satisfying Property (*). Let the moves in H be $H_1, H_2, \dots, H_\ell$. Let $\mathcal{T}$ be the set of alive tokens after move H_i has occurred for some $i \in [\ell]$. Let $\mathcal{M}$ be the sequence of moves that occur in H after H_i, i.e, $\mathcal{M} = \{H_{i+1}, \dots, H_\ell\}$. We say that the token b is* removable *after move H_i has occurred if:*

- *There exists a sequence $\mathcal{M}'$ of moves that moves the tokens in $\mathcal{T} \setminus b$ from their positions after move $H_(i)$ to the root.*
- *The cost of $\mathcal{M}'$ is at most the cost of $\mathcal{M}$.*

- *The sequence $\mathcal{M}'$ satisfies Property (*).*

Intuitively, the meaning of token b being removable after move H_i is that if b somehow magically "disappears" after move H_i, then we can modify the rest of the game without increasing the cost. Therefore, if we modify the solution locally in a way that the only difference after move H_i is that b is no longer present, then the rest of the game can be changed accordingly.

The next lemma states that the tokens that have just performed a flip are removable.

Lemma 3.12. *If b is a token whose last move in the sequence $H_1, \dots, H_t$ was a flip (potentially a flip of length 0), then b is* removable *after move H_t.*

Proof. In this proof, we consider a backward token b, but the arguments for forward tokens are similar. Let $H = H_1, \dots, H_\ell$ be the original solution and suppose that b is at vertex x after move H_t. We define the index q as the least index greater than t such that H_q that is either a SingleBackwardAbsorb (or SingleBackwardAbsorbAndMeet) move where b absorbs another token b', or a SingleBackwardAbsorb (or SingleBackwardAbsorbAndMeet) move where another token b' absorbs b, or a NonEmptyFlip where b together with a forward token f' picks up at least one backward token; we define $q = \ell$ if there is no such move. For $t < i \leq q$, we replace each H_i with a sequence of moves H'_i; we describe this in detail below[3]. For the analysis of how the cost changes by these replacement, let Q be

[3]The following is a brief sketch of why it suffices to only replace moves H_i for $t < i \leq q$: if H_q is a SingleBackwardAlone (or SingleBackwardAbsorbAndMeet) move involving b, then either b gets absorbed (and we do not worry about it) or b absorbs another backward token b' (we can pretend b' is b henceforth).

the set of vertices on which b steps with an SingleBackwardAlone move during the steps H_{t+1}, ..., H_q; each of these steps incur a cost of 1 in the original solution H. We show that the replacement decreases the cost by $|Q|$ (as each of these SingleBackwardAlone moves involving b will be removed) and increases the cost by at most $|Q|$ (as an increase of cost can occur only when a token moves on a vertex of Q). More precisely, we need the following claim:

Claim 3.1. *If for some $t < i \leq q$, move H_i moves some token to a vertex w containing only b, then $w \in Q$.*

Proof. Suppose that b is at vertex w and a token moves to w, which is only possible if w is not dead. We show first that w is not the same vertex where b is after move H_t. Suppose the contrary. Then we know by assumption that b made a flip at w during the moves H_1, ..., H_t. If this flip was not of length 0, then w became dead and hence no token can move to w in move H_i, a contradiction. If this flip was a flip of length 0, then a forward token was at w at some point before move H_{t+1}. Thus before move H_i, either there is still a forward token at w (contradicting that w contains only b), or a forward token already moved off w, in which case w is dead, again a contradiction.

Therefore, b made a move after H_t to reach w. As this move occurred before H_q, it is not a SingleBackwardAbsorb move. As no token moved off w before H_i, the only

In the remaining case when H_q is a NonEmptyFlip with f' which picks up a backward token, then we can again pretend that some other backward token picked up in the NonEmptyFlip will henceforth play the role of b (see Case 10 of Lemma 3.12) possibility is that b reached w with a SingleBackwardAlone. As this move occurred after H_t, it follows that $w \in Q$. □

Now we continue with the proof of Lemma 3.12. The following detailed list shows we modify the moves after H_t.

1. $H_i =$ **SingleBackwardAlone**(b, y, z): Then $z \in Q$. Set $H_i' = \emptyset$; this decreases cost by 1.
2. $H_i =$ **SingleBackwardAbsorb**(b, y, z, b'): This means that $i = q$. At this point, the token b would have eaten up the token b'. Set $H_i' = \emptyset$, and for every $q < i \leq \ell$, replace b with b'. That is, instead of b eating b', we pretend that b' survives and takes on the role of b.
3. $H_i =$ **SingleBackwardAbsorb**(b', z, y, b): This means that $i = q$. At this point the token b' would have eaten up the token b. Let $H_i' =$ SingleBackwardAlone (b', z, y). T his increases cost by 1, but note that Claim 3.1 implies that $y \in Q$.
4. $H_i =$ **SingleBackwardMeet**(b, y, z, f): Set $H_i' = \emptyset$; this modification does not changes the cost.
5. $H_i =$ **SingleForwardMeet**(f, z, y, b): This implies that there is no forward token at y when H_i occurs, otherwise a SingleForwardAlone move would have taken place. Set $H_i' =$ SingleForwardAlone (f, z, y). This modification increases cost by 1, but note that Claim 3.1 implies that $y \in Q$.
6. $H_i =$ **SingleBackwardAbsorbAndMeet**(b, y, z, b', f): This means that $i = q$. At this point, the token b would have eaten up the token b'. Set $H_i' = \emptyset$, and for every

$q < i \leq \ell$, replace b with b'. That is, instead of b eating b', we pretend that b' survives and takes on the role of b.

7. $H_i =$ **SingleBackwardAbsorbAndMeet**(b', z, y, b, f): This means that $i = q$. At this point the token b' would have eaten up the token b. Let $H_i' =$ SingleBackwardMeet(b', z, y, f); this does not change the cost.

8. $H_i =$ **EmptyFlip**(f, b, z, y): Replace this move by a sequence H_i' of ForwardSingleAlone moves, which takes f from z to y. Note that these set of moves have a combined cost of 1 more than the cost of H_i, as they also pay for the vertex y unlike H_i. However, note that Claim 3.1 implies that $y \in Q$.

9. $H_i =$ **NonEmptyFlip**$(f, b, z, y, g_1, g_2, \ldots, g_r, u_1, u_2, \ldots, u_r)$ picking up no backward token: Denote z by u_0 for sake of notation. We replace H_i with the following sequence H_i' of moves. For $1 \leq i \leq r$, add the collection of moves resulting in ForwardAbsorb(f, u_{i-1}, u_i, g_i). Finally, add the collection of moves resulting in ForwardAlone(f, g_r, y). It is easy to see that the cost of H_i' is 1 more than the cost of H_i, since it paid for the vertex y and H_i did not. However, note that by Claim 3.1, we have $y \in Q$.

10. $H_i =$ **NonEmptyFlip**$(f, b, z, y, g_1, g_2, \ldots, g_r, u_1, u_2, \ldots, u_r)$ picking up at least one backward token: Then $i = q$. We replace H_i with the following sequence H_i' of moves. Let $1 \leq j \leq r$ be the largest index such that g_i is a backward token (such an index exists, otherwise we are in the previous case). The sequence H_i' starts with the move NonEmptyFlip$(f, g_j, z, u_j, g_1, g_2, \ldots, g_{j-1}, u_1, u_2, \ldots, u_{j-1})$ if

$j > 1$ and with the move EmptyFlip(f, g_1, z, u_1) if $j = 1$. Then for $j < i \le \ell$, every g_i is a forward token and we add to H_i' the collection of moves resulting in ForwardAbsorb(f, u_{i-1}, u_i, g_i). Finally, we add the collection of moves which result in ForwardAlone(f, g_ℓ, y). Note that H_i moves token f to y, token b to z, and destroys the tokens $\{g_1, \ldots, g_r\}$; whereas H_i' moves token f to y, backward token g_j to z, and destroys the tokens $\{b, g_1, \ldots, g_{j-1}, g_{j+1}, \ldots, g_r\}$. Thus by renaming b to g_j in the moves H_{q+1}, $\ldots$, H_ℓ, we get a consistent sequence of moves. It is easy to see that cost of H_i' is 1 more than the cost of H_i, since it paid for the vertex y and H_i did not. However, by Claim 3.1, we have $y \in Q$.

It can be seen that the modified sequence satisfies Property (*): for every step of the modified sequence where a vertex becomes dead, there is a corresponding move in the original sequence where the same vertex becomes dead. Furthermore, we can see that the cost increases only when a token moves to a vertex in Q containing only b. Note that such a cost increase cannot happen two times for the same vertex w: the first such token arriving to w has to leave before the second such token can enter, but then vertex w is already dead when the second token arrives, contradicting that the modified sequence satisfies Property (*). Therefore, the total increase of the cost is at most $|Q|$, while removing the SingleBackwardAlone moves of b decreases the cost by exactly $|Q|$. $\square$

3.1.2.5.2 Bounding the number of "bad" EmptyFlips in H^*

We now show that the number of MultipleFlip moves in $P(H^*)$ is $O(k)$. First, we define the following special type of EmptyFlip move:

Definition 3.6. *A EmptyFlip(f,b,u,v) in H^* is said to be* **good** *if the next move (if it exists) in the Feldman-Ruhl game, which is not an SingleAlone move involving either f or b, is one of the following:*

- *EmptyFlip involving f and b, or*
- *SingleForwardMeet or SingleBackwardMeet involving both f and b.*

Otherwise, we call the EmptyFlip move as **bad**.

We would now like to give an upper bound on the number of bad EmptyFlip moves in H^*. Let the moves in H^* be $H_1, H_2, \ldots, H_\ell$

Theorem 3.5. *The total number of bad EmptyFlip moves in H^* is $O(k)$.*

Proof. We consider the Feldman-Ruhl game as it progresses and assign time to the sequence of moves as they occur. Let H_i = EmptyFlip(f,b,v_1,v_2) be the first EmptyFlip involving either of these tokens

If there is no other EmptyFlip involving either f or b after H_i, then in the worst case this EmptyFlip is a *bad* one. We charge this (possibly) bad move to the tokens f and b.

Otherwise, w.l.o.g let the first EmptyFlip after H_i involving either f or b be $H_j =$ EmptyFlip (f,b',w_1,w_2).

Let H_r be the first move after H_i that is not a SingleAlone move, and involves at least one of f and b. Hence $i < r \leq j$. We have the following cases for H_r:

1. H_r **is a SingleAbsorb or SingleAbsorbAndMeet move**: This means that H_i is a *bad* EmptyFlip; but we can charge it to this SingleAbsorb or SingleAbsorbAndMeet move H_r.
2. H_r **is a NonEmptyFlip**: This means that H_i is a *bad* EmptyFlip; but we can charge it to this NonEmptyFlip move H_r.
3. $H_r =$ **EmptyFlip**(f, b', x_1, x_2): If $b' = b$, then H_i is a good EmptyFlip. Otherwise, if $b' \neq b$, then we arrive at a contradiction to Property (**) in Section 3.1.2.5.
4. $H_r =$ **SingleForwardMeet**(f, b', x_1, x_2): If $b' = b$, then H_i is a good EmptyFlip. Otherwise, if $b' \neq b$, then we arrive at a contradiction to Property (**) in Section 3.1.2.5.
5. $H_r =$ **SingleBackwardMeet**(b', f, x_2, x_1): If $b' = b$, then H_i is a good EmptyFlip. Otherwise, if $b' \neq b$, then we arrive at a contradiction to Property (**) in Section 3.1.2.5.

To summarize, if H_i is the last EmptyFlip involving either f or b, then we can charge this (possibly) bad EmptyFlip to the tokens f or b. In Case 1, we charge the bad EmptyFlip H_i to a SingleAbsorb or SingleAbsorbAndMeet move H_r that is the next move in H^* involving either f or b. Observe that we can charge at most two bad EmptyFlip moves to one SingleAbsorb move, and at most three bad EmptyFlips to one SingleAbsorbAndMeet move: two EmptyFlips charged to a SingleAbsorb or SingleAbsorbAnd-

Meet move cannot share a token (otherwise it would not be true that the SingleAbsorb or SingleAbsorbAndMeet is the next move involving these tokens), and the SingleAbsorb move involves 2 tokens while the SingleAbsorbAndMeet move involves 3 tokens. Now Lemma 3.8 implies that only $O(k)$ bad EmptyFlip moves can be charged to SingleAbsorb or SingleAbsorbAndMeet moves.

In Case 2, we charge the bad EmptyFlip H_i to the NonemptyFlip H_r that is the next move in H^* involving either f or b. Again, if two bad EmptyFlip moves are charged to the same NonemptyFlip H_i, then the EmptyFlip moves cannot share any tokens. Therefore, the number of bad EmptyFlips that can be charged to H_r is at most $|\texttt{involved}(H_r)|$. The sum of involved vertices in all NonemptyFlip moves can be upper bounded by $3 \cdot 2k$: the factor 3 comes from the fact that a NonEmptyFlip involving 3 tokens picks up only a single token. Therefore, only $O(k)$ bad EmptyFlip moves can be charged to NonemptyFlip moves.

In Lemma 3.13, Lemma 3.14 and Lemma 3.15 we show that Case 3, Case 4 and Case 5 respectively cannot occur by arriving at a contradiction. Therefore, the number of bad EmptyFlip moves is $O(k)$. □

Lemma 3.13. *Case 3 from Theorem 3.5 leads to a contradiction.*

Proof. Suppose that $H_r = \text{EmptyFlip}(f, b', x_1, x_2)$. Then $j = r$ and $w_1 = x_1$ and $w_2 = x_2$. If $b' = b$, then H_i is a good EmptyFlip. Hence suppose $b' \neq b$. By the definition of H_r, the only possible moves involving f or b between H_i and H_r. Hence after move H_r, the token b must still be present in the game: let it be located at say vertex v_3.

We know that f travels from v_2 to x_1 between time T_0 and T. So currently the moves in H^* are as follows:

1. The move $H_i = \text{EmptyFlip}(f, b, v_1, v_2)$ takes place.
2. Between the moves H_i and H_r, the tokens f and b complete a series of SingleAlone moves to travel $v_2 \rightarrow x_1$ and $v_1 \leftarrow v_3$, respectively.
3. Then the move $H_r = \text{EmptyFlip}(f, b', x_1, x_2)$ takes place.

Therefore, after move H_r, the tokens b, b' and f are at the vertices v_3, x_1, and x_2, respectively. By Lemma 3.12, the token b' is removable after move H_r, since its last move was a flip. Let $\mathcal{M}'$ be the sequence of moves given by Definition 3.5 that can be used after H_r if we can make b' "disappear."

We now construct another solution H^{**} for the game, which violates Property (**), as follows:

1. Until the move H_i, the solution H^{**} has the same moves as H^*.
2. Delete the move H_i
3. Between the moves H_i to H_r, any move in H^* that does not involve f or b is included in H^{**}.
4. Replace H_r by the move $\text{NonEmptyFlip}(f, b', v_1, x_2, b, v_2)$ followed by the SingleBackwardAlone moves that take b' from $v_1 \leftarrow v_3$ (the same ones that b used in H^* to go from $v_1 \leftarrow v_3$). Rename b' as b, i.e., henceforth b' plays the role of b.
5. Observe now that the tokens b and f are at vertices v_3 and x_2, respectively, which

is exactly the same situation as in H^*. The token b' no longer appears. Therefore, we can finish the solution H^{**} with the sequence $\mathcal{M}'$ guaranteed by Definition 3.5.

Observe that the cost of the solution did not increase with these modifications and it still satisfies property (*). In H^*, the tokens b, b' and f were all present after move H_r. In H^{**}, let NonEmptyFlip$(f, b', v_1, x_2, b, v_2)$ be the z-th move. Then it is easy to see that $z < r$, since we replaced the two EmptyFlips H_i and H_r by a single NonEmptyFlip in H^{**} (also the bunch of SingleBackwardAlone moves of b occur before H_r in H^*, and after the NonEmptyFlip in H^{**}). Hence it is easy to see that $H^{**} \prec_{FR} H^*$, a contradiction to Property (**). □

Lemma 3.14. *Case 4 from Theorem 3.5 leads to a contradiction.*

Proof. Suppose $H_r = \text{SingleForwardMeet}(f, b', x_1, x_2)$. If $b' = b$, then H_i is a good EmptyFlip. Hence suppose $b' \neq b$. By definition of H_r the only possible moves involving f or b between H_i to H_r are SingleAlone moves. Hence after move H_r in H^*, the token b must still be present in the game: let it be located at say vertex v_3.

Also between the moves H_i and H_r, the token f makes a bunch of SingleAlone moves and travels from v_2 to x_1. So currently the moves in H^* are as follows:

1. The move $H_i = \text{EmptyFlip}(f, b, v_1, v_2)$ takes place.
2. Between the moves H_i and H_r, the tokens f, b complete a bunch of SingleAlone moves to travel from $v_2 \to x_1$ and $v_1 \leftarrow v_3$ respectively.
3. After move H_{r-1}, the move $H_r = \text{SingleForwardMeet}(f, b', x_1, x_2)$ takes place. Both

the tokens f, b' are located at x_2, and then an EmptyFlip of length 0 takes place at x_2.

Therefore, after move H_r, the tokens b, b' and f are at the vertices v_3, x_2, and x_2, respectively. By Lemma 3.12, the token b' is removable after move H_r, since its last move was a flip (in fact an EmptyFlip on length 0). Let $\mathcal{M}'$ be the sequence of moves given by Definition 3.5 that can be used after H_r if we can make b' "disappear." We now construct another solution H^{**} for the game, which violates Property (**), as follows:

1. Until the move H_i, the solution H^{**} has the same moves as H^*.
2. Delete the move H_i
3. Between the moves H_i to H_r, any move in H^* that does not involve f or b is included in H^{**}.
4. Replace H_r by the move NonEmptyFlip$(f, b', v_1, x_2, b, v_2)$ followed by the Single-BackwardAlone moves that take b' from $v_1 \leftarrow v_3$ (the same ones that b used in H^* to go from $v_1 \leftarrow v_3$). Rename b' as b, i.e., henceforth b' plays the role of b.
5. Observe now that the tokens b and f are at vertices v_3 and x_2, respectively, which is exactly the same situation as in H^*. The token b' no longer appears. Therefore, we can finish the solution H^{**} with the sequence $\mathcal{M}'$ guaranteed by Definition 3.5.

Observe that the cost of the solution did not increase with these modifications and it still satisfies property (*). In H^*, the tokens b, b' and f were all present after move H_r. In H^{**}, let NonEmptyFlip$(f, b', v_1, x_2, b, v_2)$ be the z-th move. Then it is easy to see that

$z < r$, since we replaced the two moves H_i and H_r by a single NonEmptyFlip in H^{**} (also the bunch of SingleBackwardAlone moves of b occur before H_r in H^*, and after the NonEmptyFlip in H^{**}). Hence it is easy to see that $H^{**} \prec_{FR} H^*$, a contradiction to Property (**). □

Lemma 3.15. *Case 5 from Theorem 3.5 leads to a contradiction.*

Proof. Suppose $H_r = \text{SingleBackwardMeet}(b', f, x_2, x_1)$. If $b' = b$, then H_i is a good EmptyFlip. Hence suppose $b' \neq b$. By definition of H_r the only possible moves involving f or b between the moves H_i and H_r are SingleAlone moves. Hence after move H_r in H^*, the token b must still be present in the game: let it be located at say vertex v_3.

Also between H_i and H_r, the token f makes a bunch of SingleAlone moves and travels from v_2 to x_1. So currently the moves in H^* are as follows:

1. The move $H_i = \text{EmptyFlip}(f, b, v_1, v_2)$ takes place.
2. Between the moves H_i and H_r, the tokens f, b complete a bunch of SingleAlone moves to travel from $v_2 \rightarrow x_1$ and $v_1 \leftarrow v_3$ respectively.
3. After move H_{r-1}, the move $H_r = \text{SingleBackwardMeet}(b', f, x_2, x_1)$ takes place. Both the tokens f, b' are located at x_1, and then an EmptyFlip of length 0 takes place at x_1.

Therefore, after move H_r, the tokens b, b' and f are at the vertices v_3, x_1, and x_1, respectively. By Lemma 3.12, the token b' is removable after move H_r, since its last move was a flip (in fact an EmptyFlip on length 0). Let $\mathcal{M}'$ be the sequence of moves given by

Definition 3.5 that can be used after H_r if we can make b' "disappear." We now construct another solution H^{**} for the game, which violates Property (**), as follows:

1. Until the move H_i, the solution H^{**} has the same moves as H^*.
2. Delete the move H_i
3. Between the moves H_i to H_r, any move in H^* that does not involve f or b is included in H^{**}.
4. Replace H_r by the move SingleBackwardAlone(b', x_2, x_1) followed by the move NonEmptyFlip$(f, b', v_1, x_1, b, v_2)$ and then the bunch of SingleBackwardAlone moves that take b' from $v_1 \leftarrow v_3$ (the same ones that b used in H^* to go from $v_1 \leftarrow v_3$). Rename b' as b, i.e., henceforth b' plays the role of b.
5. Observe now that the tokens b and f are at vertices v_3 and x_1, respectively, which is exactly the same situation as in H^*. The token b' no longer appears. Therefore, we can finish the solution H^{**} with the sequence $\mathcal{M}'$ guaranteed by Definition 3.5.

Observe that the cost of the solution did not increase with these modifications and it still satisfies property (*). In H^*, the tokens b, b' and f were all present after move H_r. In H^{**}, let NonEmptyFlip$(f, b', v_1, x_2, b, v_2)$ be the z-th move. Then it is easy to see that $z \leq r$, since we replaced the two moves H_i and H_r by a SingleBackwardAlone move and a NonEmptyFlip in H^{**} (also the bunch of SingleBackwardAlone moves of b occur before H_r in H^*, and after the NonEmptyFlip in H^{**}). Hence it is easy to see that $H^{**} \prec_{FR} H^*$, a contradiction to Property (**).

$\square$

3.1.2.5.3 Bounding the number of MultipleFlips in $P(H^*)$

Theorem 3.6. *If M is a MultipleFlip in $P(H^*)$ involving f and b such that no EmptyFlip component of M is bad, then there is no further MultipleFlip in $P(H^*)$ involving either f or b. As a corollary, we obtain that the total number of MultipleFlips in $P(H^*)$ is $O(k)$.*

Proof. Let $M = \text{MultipleFlip}(f,b,v_1,v_2,w_1,w_2)$ and its components be $H_i, H_{i+1}, \ldots, H_j$. We are given that all EmptyFlips of M are good. Let H_ℓ be the last EmptyFlip component of M. Hence $i \leq \ell \leq j$. If there is no further MultipleFlip in $P(H^*)$ involving either f or b, then we have nothing to prove. Hence, suppose there is such a MultipleFlip and w.l.o.g let $M' = \text{MultipleFlip}(f,b',x_1,x_2,y_1,y_2)$ be the next MultipleFlip in $P(H^*)$ involving f or b. The next claim shows that in fact M' is the next supermove in $P(H^*)$ which involves f or b.

Claim 3.2. *M' is the next supermove involving f or b after M in $P(H^*)$.*

Proof. Suppose to the contrary that there is another supermove (involving f or b) after M and before M' in $P(H^*)$: let $M'' \neq M'$ be the first such supermove after M in $P(H^*)$. We consider the following choices for M''.

- M'' is an Absorb, NonEmptyFlip or AbsorbAndMeet supermove: In this case, Definition 3.6 implies that the move H_ℓ is a bad EmptyFlip, which is a contradiction since no EmptyFlip of M is bad.
- M'' is an Alone supermove: W.l.o.g let the Alone supermove be $F = \text{ForwardAlone}(f,v_2,v_3)$. Note that M'' is the next supermove (involving f or b) after M in $P(H^*)$. Also,

M'' pays for each vertex from the set $\texttt{internal}(F) \cup \{v_3\}$. If there is any supermove M''' in between M and M'' such that $\Big(\texttt{internal}(M''') \cup \texttt{corners}(M''')\Big) \cap \Big(\texttt{internal}(F) \cup \{v_3\}\Big) \neq \emptyset$, then both M''' and M'' would pay for any vertex in the intersection , which is a contradiction. Hence we can reorder the partition $P(H^*)$ so that the next supermove (not necessarily involving f or b) after M in $P(H^*)$ is M''. But then we can merge the two supermoves M and F into a single MultipleFlip, which reduces the number of supermoves in $P(H^*)$ and contradicts Property (**).

- M'' is a Meet supermove: By Lemma 3.11, there are no Meet supermoves in $P(H^*)$ and hence this case does not need to be considered.
- M'' is a MultipleFlip: This implies that $M'' = M'$, a contradiction.

In each of the cases, we were able to derive a contradiction. Hence M' is indeed the next supermove involving f or b after M in $P(H^*)$. $\square$

We now continue with the proof of Theorem 3.6. There are now two cases depending on whether $b' = b$, or not:

- $\underline{b' = b}$: Note that M' is the next supermove (involving f or b) after M in $P(H^*)$. Also, M' pays for each vertex from the set $\texttt{internal}(F) \cup \{v_2, w_2\} \setminus \{v_1, w_1\}$. If there is any supermove M^* in between M and M' such that $\Big(\texttt{internal}(M^*) \cup \texttt{corners}(M^*)\Big) \cap \Big(\texttt{internal}(F) \cup \{v_2, w_2\} \setminus \{v_1, w_1\}\Big) \neq \emptyset$, then both M^* and M' would pay for all the vertices in the intersection, which is a contradiction. Hence we can reorder the partition $P(H^*)$ so that the next supermove (not necessarily involving f or b) after M in $P(H^*)$ is M'. But then we can merge the two supermoves M and M' into a single MultipleFlip, which reduces the number of supermoves in $P(H^*)$ and contradicts Property (**).

- $\underline{b' \neq b}$: In this case, the first component of M' which involves f and is not a SingleAlone move[4], leads to H_ℓ being a bad EmptyFlip, which is a contradiction since no EmptyFlip of M is bad.

We now show that the total number of MultipleFlips in $P(H^*)$ is $O(k)$ by a charging argument: we charge each MultipleFlip in $P(H^*)$ to either a bad EmptyFlip, or to both the tokens involved in the MultipleFlip. Note that the number of tokens is $2k$, and Theorem 3.5 bounds the number of bad EmptyFlips in H^* by $O(k)$. Hence the claim follows.

□

Theorem 3.7. *The total number of supermoves in $P(H^*)$ is $O(k)$.*

Proof. Follows from Lemma 3.8, Lemma 3.9, Lemma 3.10, Lemma 3.11 and Theorem 3.6. □

3.1.2.5.4 Clean MultipleFlips

We have seen in the previous section that the partition $P(H^*)$ has $O(k)$ supermoves. Now we define a special property of MultipleFlip supermoves, which will be useful for us later.

[4]We are guaranteed that such a move exists since one of the components of M' is the EmptyFlip move involving f and b'.

Now we state a lemma which shows how to decompose a MultipleFlip into other types of supermoves.

Lemma 3.16. *Let M be a MultipleFlip involving f and b that contains EmptyFlips $E_1, E_2, \ldots, E_r$. Then we can subpartition M into $O(r)$ supermoves such that each supermove belongs to the set {Alone, MultipleFlip, Meet}, and each of the newly created MultipleFlips is clean (in fact is an EmptyFlip).*

Proof. Since M is a MultipleFlip, we know that $r \geq 1$. Also M contains at most $r+1$ SingleMeet moves involving f and b, since each such move is followed by an EmptyFlip between f and b of length 0 (it may happen that the last EmptyFlip of length 0 might not be included in M, but all the previous ones must be included). We consider each of these SingleMeet moves as a Meet supermove. Finally, note that we can merge all the SingleAlone moves of both f and b, which lie in between any two moves from the set $\{\text{EmptyFlip}(f,b), \text{Meet}(f,b)\}$, into Alone moves of the respective tokens. Hence the total number of SingleAlone moves is $2(r+(r+1)+1) = 4r+4$: there are a total of $r+(r+1)+1$ gaps formed in between, before and after the moves from the set $\{\text{EmptyFlip}(f,b), \text{Meet}(f,b)\}$. Therefore, by considering each EmptyFlip as a (clean) MultipleFlip, we can subpartition M into $r+(r+1)+(4r+4) = 6r+5 = O(r)$ supermoves such that each supermove belongs to the set {Alone, MultipleFlip, Meet} and each newly created MultipleFlip is clean. □

We now show the following lemma which says that no vertex can be involved in more than two flips.

Lemma 3.17. *No vertex v can be the* `corner`[5] *of more than two flips*

Proof. We show the following equivalent statement: if a vertex v is a `corner` in at least two flips, then it becomes dead after the second flip (which implies that v cannot be a corner of any other flip henceforth). If v is a `corner` of at most flip, then we are fine. So suppose that v is a `corner` of at least two flips: let the first two of these flips be F_1 and F_2. If F_1 is a flip (Empty or NonEmpty) of non-zero length, then the vertex v becomes dead after F_1 occurs (refer to end of Section 3.1.1.2). Therefore, we can assume that F_1 is an EmptyFlip of length 0. Now if F_2 is a flip (Empty or NonEmpty) of non-zero length, then the vertex v becomes dead after F_2 occurs.

So, the only case to be considered is when both F_1 and F_2 are EmptyFlips of length 0. Let H_1, H_2 be the Meet moves which precede the EmptyFlips F_1, F_2 respectively. Let the two tokens involved in H_1 be f and b. So after F_1 occurs the two tokens f and b will be located at the vertex v. If either of this tokens leaves v before H_2 occurs, then v becomes dead and then F_2 cannot occur. Hence both the tokens f and b remain at v when H_2 occurs. But this implies that H_2 must be a AbsorbAndMeet move instead of a Meet move, since v has both a forward and a backward token. This leads to a contradiction. $\square$

From Theorem 3.7, we know that there is a partition $P(H^*)$ of H^* into $O(k)$ supermoves. We want to prove that the MultipleFlip supermoves can be assumed to be clean:

Theorem 3.8. *There is a partition $P'(H^*)$ of H^* into $O(k)$ supermoves, such that each*

[5]Recall the definition of `corners` of EmptyFlip and NonEmptyFlips from Section 3.1.2.2.1

MultipleFlip in $P'(H^*)$ is clean.

We now consider various cases that can occur in a MultipleFlip entry in a description of the game. Let the MultipleFlip entry be $M = \text{MultipleFlip}(f, b, v_1, v_2, w_1, w_2)$. We have three cases depending on the number of different elements in the multiset $L = \{v_1, v_2, w_1, w_2\} = \texttt{corners}(M)$. Recall that the MultipleFlip move takes the token f from v_1 to v_2 and the token b from w_1 to w_2.

First we define a special type of EmptyFlip that we will use in our proofs:

Definition 3.7. *Let $M = MultipleFlip(f, b, v_1, v_2, w_1, w_2)$. Then we say that an EmptyFlip E is* independent *if $\texttt{corners}(E) \cap \texttt{corners}(M) = \emptyset$.*

We now consider two possibilities for the number of distinct elements in L.

L has 3 distinct elements

By symmetry we only need to consider the following three cases:

Case 1: Only $v_1 = v_2$ and w_1, w_2 are distinct

The initial and final location of f is the same implies that there is always a token at vertex v_1: otherwise we would violate Property (*). Hence v_1 is an endpoint of every EmptyFlip that is contained in the Multiple M. By Lemma 3.17, M has at most two EmptyFlips. By Lemma 3.16, we can subpartition M into $O(1)$ supermoves such that each supermove is from the set {Alone, MultipleFlip, Meet} and each newly created MultipleFlip is clean. □

Case 2: Only $v_1 = w_2$ and v_2, w_1 are distinct

Since the final location of b is $w_2 = v_1$, which is the initial location of f, we know that there is always a token at $v_1 = w_2$ throughout the MultipleFlip, i.e., every EmptyFlip in M has v_1 as one endpoint. By Lemma 3.17, M has at most two EmptyFlips. By Lemma 3.16, we can subpartition M into $O(1)$ supermoves such that each supermove is from the set $\{$Alone, MultipleFlip, Meet$\}$ and each newly created MultipleFlip is clean. □

Case 3: Only $v_1 = w_1$ and v_2, w_2 are distinct

If M does not have an independent EmptyFlip, then by Lemma 3.17 the number of EmptyFlips is M is upper bounded by 6. By Lemma 3.16, we can subpartition M into $O(1)$ supermoves such that each supermove is from the set $\{$Alone, MultipleFlip, Meet$\}$ and each newly created MultipleFlip is clean.

Hence suppose that M contains an independent EmptyFlip say E, and let $\texttt{corners}(E) = \{v_3, w_3\}$. Let G^*_M be the underlying undirected graph of the induced graph $G[\texttt{corners}(M) \cup \texttt{internal}(M)]$ By the definition of a MultipleFlip, every vertex of G^*_M lies on a $v_1 - v_3$ path or a $v_3 - v_2$ path or a $w_3 - w_1$ path. Also there is a $v_3 - w_3$ path in G^*_M. Hence the induced graph $G^*_M[\texttt{internal (M)}]$ is connected: each vertex can reach either v_3 or w_3, which are themselves connected via a path. It is clear that $v_1 = 2_1$ has a path to both v_3 and w_3 that does not go through any other vertex of $\texttt{corners}(M)$. Let the (undirected) $v_3 - v_2, w_3 - w_2$ paths be P_1, P_2 respectively. We have the following four possibilities:

- If neither P_1 passes through w_2 or P_2 passes through v_2, then $\texttt{internal}(M)$ is adjacent to each vertex of L, and hence M is clean.

- P_1 passes through w_2 and P_2 passes through v_2: In this case, the last component of M is an EmptyFlip. Consider this EmptyFlip separately as a (clean) MultipleFlip say M''. Since M contains an independent EmptyFlip, the remaining components of M form a MultipleFlip, say $M' =$ MultipleFlip (f,b,v_1,w_2,v_1,v_2). Note that $\texttt{internal}(M') \subseteq \texttt{internal(M)}$ and $\texttt{corners}(M) = \texttt{corners(M')}$. Observe now that the set $\texttt{internal}(M')$ is adjacent in $G^*_{M'}$ to each vertex of $\texttt{corners}(M')$. Also $G^*_M[\texttt{internal(M')}]$ remains connected since we know that $G^*_M[\texttt{internal(M)}]$ is connected and there is a $v_2 - w_2$ path in $G^*_{M'}$[6]. Since the conditions of Definition 3.2 are satisfied, we get that M' is clean.

- P_1 passes through w_2 but P_2 does not pass through v_2: In this case, the last component say C of M is an SingleForwardAlone(f,w_2,v_2) move.[7] By considering C separately as an Alone supermove say M'', the rest of M is reduced to $M' =$ MultipleFlip(f,b,v_1,w_2,v_1,w_2). Since $|\texttt{corners}(M')| = 2$, by Definition 3.2 we get that M' is clean.

- P_1 does not pass through w_2 but P_2 passes through v_2: In this case, the last component say C of M is an SingleBackwardAlone(b,v_2,w_2) move.[8] By considering

[6]This path is a concatenation of $v_2 - w_3$ subpath of P_2 followed by $w_3 - v_3$ path given by the EmptyFlip followed by the $v_3 - w_2$ subpath of P_1

[7]The last move cannot be a EmptyFlip (since otherwise P_2 would also pass through v_2), a SingleMeet move (since $v_2 \neq w_2$) or a SingleBackward move (since then P_2 would pass through v_2).

[8]The last move cannot be a EmptyFlip (since otherwise P_1 would also pass through w_2), a SingleMeet move (since $v_2 \neq w_2$) or a SingleBackward move (since then P_1 would pass through w_2).

C separately as an Alone supermove say M'', the rest of M is reduced to $M' =$ MultipleFlip$(f, b, v_1, v_2, v_1, v_2)$. Since $|\texttt{corners}(M')| = 2$, by Definition 3.2 we get that M' is clean.

To summarize, if either P_1 passes through w_2 or P_2 passes through v_2, then we can either split M into two clean MultipleFlips, or into one Alone supermove and one clean MultipleFlip. $\square$

L has 4 distinct elements

Case 4: There is an *independent* EmptyFlip E in the MultipleFlip M

Let $\{v_3, w_3\} = \texttt{corners}(E)$. Let G^*_M be the underlying undirected graph of the induced graph $G[\texttt{corners}(M) \cup \texttt{internal}(M)]$ By the definition of a MultipleFlip, every vertex of G^*_M lies on a $v_1 - v_3$ path or a $v_3 - v_2$ path or a $w_1 - w_3$ path or a $w_3 - w_1$ path[9]. Also there is a $v_3 - w_3$ path in G^*_M. Hence the induced graph $G^*_M[\texttt{internal (M)}]$ is connected: each vertex can reach either v_3 or w_3, which are themselves connected via a path. Let the $v_1 - v_3, w_1 - w_3$ paths in G^*_M be P_1, P_2 respectively. Similarly, let the $v_3 - v_2, w_3 - w_2$ paths in G^*_M be P_3, P_4 respectively. We now set the following notation:

[9]Observe that the $v_1 - v_3$ path may go through w_1, but cannot go through v_2 or w_2 since these vertices are only discovered after the EmptyFlip between v_3 and w_3 occurs. That is, the set vertices discovered until the EmptyFlip between v_3 and w_3 occurs is disjoint from the set of vertices discovered after the EmptyFlip between v_3 and w_3

let $x_1 = w_1, x_2 = v_1, x_3 = w_2, x_4 = v_2$. Now a four tuples from the set $\{0,1\}^4$ has its i-th co-ordinate as 0 if P_i does not pass through x_i, and 1 otherwise.

We consider the following possibilities corresponding the 16 tuples:

- $\underline{(0,0,0,0)}$: In this case, $\texttt{internal}(M)$ is adjacent in G^*_M to each vertex of the set $\texttt{corners}(M)$, and hence M is clean.

- $\underline{(0,0,1,1)}$: In this case, the last component of M is an EmptyFlip, and $\texttt{internal}(M)$ is adjacent in G^*_M to both v_1 and w_1. Consider this EmptyFlip separately as a (clean) Multiple say M''. Since M has an independent flip, the remaining components of M form a MultipleFlip given by $M' = \text{MultipleFlip}(f, b, v_1, w_2, w_1, v_2)$. Note that $\texttt{internal}(M') \subseteq \texttt{internal(M)}$ and $\texttt{corners}(M) = \texttt{corners(M')}$. Observe now that the set $\texttt{internal}(M')$ is adjacent in $G^*_{M'}$ to each vertex of $\texttt{corners}(M)$. Also $G^*_{M'}[\texttt{internal(M')}]$ remains connected since we know that $G^*_M[\texttt{internal(M)}]$ is connected and there is a $v_2 - w_2$ path in $G^*_{M'}$[10]. Since the conditions of Definition 3.2 are satisfied, we get that M' is clean. Hence we can subpartition M into two clean MultipleFlips M' and M''

- $\underline{(1,1,0,0)}$: In this case, the first component of M is an EmptyFlip, and $\texttt{internal}(M)$ is adjacent in G^*_M to both v_2 and w_1. Similar to the above case, we can subpartition M into two clean MultipleFlips M' and M''.

- $\underline{(1,1,1,1)}$: In this case, the first component say M'' and last component say M''' of

[10]This path is a concatenation of $v_2 - w_3$ subpath of P_2 followed by $w_3 - v_3$ path given by the EmptyFlip followed by the $v_3 - w_2$ subpath of P_1

M are EmptyFlips. Consider M'' and M''' as (clean) MultipleFlips. Since M has an independent flip, the remaining components of M form a MultipleFlip given by $M' = \text{MultipleFlip}(f, b, w_1, w_2, v_1, v_2)$. Note that $\texttt{internal}(M') \subseteq \texttt{internal(M)}$ and $\texttt{corners}(M) = \texttt{corners(M')}$. Observe now that the set $\texttt{internal}(M')$ is adjacent in $G^*_{M'}$ to each vertex of $\texttt{corners}(M)$. Also $G^*_{M'}[\texttt{internal(M')}]$ remains connected since $G^*_M[\texttt{internal(M)}]$ is connected and there is a $v_2 - w_2$ path and a $v_1 - w_1$ path in $G^*_{M'}$ (by arguments similar to those in above cases). Since the conditions of Definition 3.2 are satisfied, we get that M' is clean. Hence, we can subpartition M into three clean MultipleFlips M', M'' and M'''

- For all the other 12 tuples either the first two co-ordinates or the last two coordinates form $(0,1)$ or $(1,0)$. Without loss of generality, let us consider the case when the tuple has first two co-ordinates as $(1,0)$. In this case, the first component say C of M is an SingleForwardAlone(f, v_1, w_1) move.[11] By considering C separately as an Alone supermove say M'', the rest of M is reduced to $M' =$ MultipleFlip$(f, b, w_1, v_2, w_1, w_2)$. Since $|\texttt{corners}(M')| = 3$, we get reduced to Cases 1–3.

□

Case 5: There is no *independent* EmptyFlip in the MultipleFlip

[11]The last move cannot be a EmptyFlip (since otherwise P_2 would also pass through v_1), a SingleMeet move (since $v_1 \neq w_1$) or a SingleBackward move (since then P_2 would pass through v_1).

This means that at least one endpoint of each EmptyFlip belongs to the set $\{v_1, v_2, w_1, w_2\}$. By Lemma 3.17, M has at most 8 EmptyFlips. By Lemma 3.16, we can subpartition M into $O(1)$ supermoves such that each supermove belongs to the set {Alone, MultipleFlip, Meet} and each newly created MultipleFlip is clean. □

Finally we are now ready to prove Theorem 3.8, which is restated below:

Theorem 3.8 . *There is a partition $P'(H^*)$ of H^* into $O(k)$ supermoves, such that each MultipleFlip in $P'(H^*)$ is clean.*

Proof. We analyze what happens in each of the 5 cases considered previously. Recall that every EmptyFlip is a clean Multiple Flip.

- In Cases 1–2, it was shown that the MultipleFlip can be subpartitioned into $O(1)$ supermoves such that each supermove belongs to the set {Alone, MultipleFlip, Meet} and each of the newly created MultipleFlips is clean.
- In Case 3, if the MultipleFlip does not contain a independent EmptyFlip then it was shown that the MultipleFlip can be subpartitioned into $O(1)$ supermoves such that each supermove belongs to the set {Alone, MultipleFlip, Meet} and each of the newly created MultipleFlips is clean. In the case when the MultipleFlip contains an independent EmtpyFlip, then it was shown that one of the following must occur:
  - M is itself clean
  - M can be subpartitioned into two clean MultipleFlips M' and M''

  - M can be subpartitioned into a clean MultipleFlip M' and an Alone supermove M''

- In Case 4, it was shown that one of the following must occur:
  - M is itself clean
  - M can be subpartitioned into two clean MultipleFlips M' and M''
  - M can be subpartitioned into three clean MultipleFlips M', M'' and M'''
  - We can reduce to Cases 1–3.

- In Case 5, it was shown that the MultipleFlip can be decomposed into $O(1)$ supermoves such that each supermove belongs to the set $\{$Alone, MultipleFlip, Meet$\}$ and each of the newly created MultipleFlips is clean.

In each of the cases, at the expense of a $O(1)$ blowup in the number of supermoves, we can convert $P(H^*)$ into a partition $P'(H^*)$ such that each MultipleFlip is clean. Since Theorem 3.7 gives a partition $P(H^*)$ of H^* into $O(k)$ supermoves, we get that $P'(H^*)$ also has $O(k)$ supermoves. $\square$

3.2 Lower Bounds

First we show that the unweighted vertex version of both SCSS and DSF problems are harder than the edge versions with integer weights. In this thesis all our hardness reductions are for edge weighted graphs with integer weights, and hence they also carry over to the unweighted vertex versions.

3.2.1 Vertex Versions are harder than Integer Weighted Edge Versions

For both the SCSS and DSF problems we show that the edge version with integer weights can be solved using the unweighted vertex version. Hence all our hardness results of Theorem 3.17, Theorem 3.14 and Theorem 3.10 holds for unweighted vertex versions as well.

We give a formal proof for the DSF problem; the proof for the SCSS problem is similar. Consider an instance $I_1 = (G,T)$ of Edge DSF with integer weights where $T = \{(s_i,t_i) \mid i \in [k]\}$. Replace each edge of weight ℓ by $n\ell$ internal vertices where $|G| = n$. Let the new graph be G'. Consider the instance I_2 of unweighted Vertex DSF where the set of terminals is the same as in I_1.

Theorem 3.9. *There is a solution I_1 of weight at most C if and only if there is a solution for Vertex DSF to I_2 with weight at most $Cn+n$ vertices.*

Proof. Suppose there is a solution E_1 for I_1 of weight at most C. For each edge in E_1 pick all its internal vertices and two endpoints in E_2. Clearly E_2 is a solution for I_2. The weight of E_2 is $Cn+\gamma$ where γ is the number of vertices of G incident to the edges in E_1. Since $\gamma \leq n$ we are done.

Suppose there is a (minimal) solution E_2 for I_2 of weight at most $Cn+n$. For any edge $e \in G$ of weight c we need to pick all the cn internal vertices (plus the two endpoints of e) in E_2 if we actually want to use e in a solution for I_1. So for every edge $e \in E$ we know that E_2 contains either or none of the internal vertices obtained after splitting up e according to its weight in G. Let the set of edges of G all of whose internal vertices are in E_2 be $E_1 = \{e_1, e_2, \dots, e_r\}$ and theirs weights be $c_1, c_2, \dots, c_r$. Since E_2 is a solution for I_2 we have E_1 is a solution for I_1. Let S be the union of set of endpoints of the edges in E_1. Therefore $Cn + n \geq |S| + n(\sum_{i=1}^{r} c_i)$. Since $|S| \geq 0$ we have $C \geq \sum_{i=1}^{r} c_i$, i.e., $|E_1|$ has weight at most C. □

3.2.2 A Tight Lower Bound for SCSS Planar Graphs

After designing faster algorithm for SCSS planar graphs from Section 3.1.2, the next natural question is whether we can obtain a better speedup than the improvement from $O(k)$ to $O(\sqrt{k})$ in the exponent of n? Our main hardness result matches our algorithm: it shows that $O(\sqrt{k})$ is best possible.

Theorem 3.10. *The* STRONGLY CONNECTED STEINER SUBGRAPH *problem restricted to the case when the underlying undirected graph is planar is W[1]-hard parameterized by the number of terminals, and cannot be solved in time $f(k) \cdot n^{o(\sqrt{k})}$ unless ETH[12] fails, where f is any computable function, k is the number of terminals, and n is the number of vertices in the instance.*

This also answers the question of Guo, Niedermeier and Suchy [GNS11], who showed the W[1]-hardness of these problems on general graphs and left the fixed-parameter

[12]Recall that ETH can be stated as the assumption that n-variable 3SAT cannot be solved in time $2^{o(n)}$ [IP01, IPZ01].

tractability status on planar graphs as an open question. Note that there are relatively few parameterized problems that are W[1]-hard on planar graphs [BLP09, CFJR07, EFG+09, Mar12a]. The reason for the scarcity of such hardness results is mainly because for most problems, the fixed-parameter tractability of finding a solution of size k in a planar graph can be reduced to a bounded-treewidth problem by standard layering techniques. However, in our case the parameter k is the number of terminals, hence such a simple reduction to the bounded-treewidth case does not seem to be possible. Our reduction is from the GRID TILING problem formulated by Marx [Marc, Mar12a], which has turned out to be a convenient starting point for parameterized reductions for planar problems.

GRID TILING
Input : Integers k, n, and k^2 non-empty sets $S_{i,j} \subseteq [n] \times [n]$ where $1 \leq i, j \leq k$
Question: For each $1 \leq i, j \leq k$ does there exist a value $\gamma_{i,j} \in S_{i,j}$ such that

- If $\gamma_{i,j} = (x, y)$ and $\gamma_{i,j+1} = (x', y')$ then $x = x'$.
- If $\gamma_{i,j} = (x, y)$ and $\gamma_{i+1,j} = (x', y')$ then $y = y'$.

Chen et al. [CHKX06] showed that under ETH the k-CLIQUE problem cannot be solved in $f(k) \cdot n^{o(k)}$ time for any computable function f. Marx [Marc] gave a reduction from k-CLIQUE to $k \times k$ GRID TILING. Composing these two reductions together implies that under ETH the problem of $k \times k$ GRID TILING cannot be solved in time $f(k) \cdot n^{o(k)}$, for any computable function f.

To prove Theorem 3.10, we give a reduction which transforms the problem of $k \times k$ GRID TILING into an instance of SCSS with $O(k^2)$ terminals. We design two types of gadgets: the *connector gadget* and the *main gadget* and arrange them in a grid-like fashion (see Figure 3.2). Each cell of the grid with a copy of the main gadget, with a connector gadget between main gadgets that are adjacent either horizontally or vertically. The main technical part of the reduction is the structural results regarding the existence and construction of particular types of connector gadgets and main gadgets (Lemma 3.18 and Lemma 3.19). Interestingly, the construction of the connector gadget poses a greater challenge: here we exploit in a fairly delicate way the fact that the $t_i \rightsquigarrow t_j$ and the $t_j \rightsquigarrow t_i$ paths appearing in the solution subgraph might need to share edges to reduce the weight.

The proof of Theorem 3.10 is divided into the following steps: In Sections 3.2.2.1 we introduce the connector gadget and main gadget. Lemma 3.18 proves the existence of a particular type of connector gadget and Lemma 3.19 proves the existence of a particular type of main gadget. Using Lemmas 3.18 and 3.19 as a blackbox, we prove Theorem 3.10 in Section 3.2.2.3. The proofs of Lemmas 3.18 and Lemma 3.19 are given later in Sections 3.2.2.4 and 3.2.2.5 respectively.

3.2.2.1 Existence of Connector And Main Gadgets

Connector Gadgets: A connector gadget CG_n is an embedded planar graph with $O(n^2)$ vertices and weights on its edges. It has a total of $2n+2$ distinguished vertices divided into the following 3 types:

- The vertices p, q are called *internal-distinguished* vertices
- The vertices $p_1, p_2, \dots, p_n$ are called *source-distinguished* vertices
- The vertices $q_1, q_2, \dots, q_n$ are called *sink-distinguished* vertices

Let $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$. The vertices $P \cup Q$ appear in the order $p_1, \ldots, p_n, q_n, \ldots, q_1$ on the boundary of the gadget. In the connector gadget CG_n, every vertex in P is a source and has exactly one outgoing edge. Also every vertex in Q is a sink and has exactly one incoming edge.

Definition 3.8. *We say an edge set $E' \subseteq E(CG_n)$ satisfies the* **connectedness** *property if each of the following four conditions hold for the graph $CG_n[E']$:*

1. *p can be reached from some vertex in P*
2. *q can be reached from some vertex in P*
3. *p can reach some vertex in Q*
4. *q can reach some vertex in Q*

Definition 3.9. *We say an edge set E' satisfying the connectedness property* ***represents*** *an integer $i \in [n]$ if in E' the only outgoing edge from P is the one incident to p_i and the only incoming edge into Q is the one incident to q_i.*

The next lemma shows we can construct a particular type of connector gadgets:

Lemma 3.18. *Given an integer n one can construct in polynomial time a connector gadget CG_n and an integer C^*_n such that the following two properties hold* [13]*:*

1. *For every $i \in [n]$, there is an edge set $E_i \subseteq E(CG_n)$ of weight C^*_n such that E_i satisfies the connectedness property and represents i. Note that, in particular, E_i contains a $p_i \rightsquigarrow q_i$ path (via p or q).*

[13]We use the notation C^*_n to emphasize that C^* depends only on n

2. *If there is an edge set $E' \subseteq E(CG_n)$ such that E' has weight at most C_n^* and E' satisfies the connectedness property, then E' has weight exactly C_n^* and it represents some $\beta \in [n]$.*

Main Gadgets: A main gadget MG is an embedded planar graph with $O(n^3)$ vertices and weights on its edges. It has $4n$ distinguished vertices given by the following four sets:

- The set $L = \{\ell_1, \ell_2, \ldots, \ell_n\}$ of *left-distinguished* vertices.
- The set $R = \{r_1, r_2, \ldots, r_n\}$ of *right-distinguished* vertices.
- The set $T = \{t_1, t_2, \ldots, t_n\}$ of *top-distinguished* vertices.
- The set $B = \{b_1, b_2, \ldots, b_n\}$ of *bottom-distinguished* vertices.

The distinguished vertices appear in the order t_1, …, t_n, r_1, …, r_n, b_n, …, b_1, ℓ_n, …, ℓ_1 on the boundary of the gadget. In the main gadget MG, every vertex in $L \cup T$ is a source and has exactly one outgoing edge. Also each vertex in $R \cup B$ is a sink and has exactly one incoming edge.

Definition 3.10. *We say an edge set $E' \subseteq E(MG)$ satisfies the* **connectedness** *property if each of the following four conditions hold for the graph $MG[E']$:*

1. *There is a directed path from some vertex in L to $R \cup B$*
2. *There is a directed path from some vertex in T to $R \cup B$*
3. *Some vertex in R can be reached from $L \cup T$*
4. *Some vertex in B can be reached from $L \cup T$*

Definition 3.11. *An edge set $E' \subseteq E(MG)$ satisfying the connectedness property **represents** a pair $(i, j) \in [n] \times [n]$ if each of the following four conditions holds:*

- *The only edge of E' leaving L is the one incident to ℓ_i*
- *The only edge of E' entering R is the one incident to r_i*
- *The only edge of E' leaving T is the one incident to t_j*
- *The only edge of E' entering B is the one incident to b_j*

The next lemma shows we can construct a particular type of connector gadgets:

Lemma 3.19. *Given a subset $S \subseteq [n] \times [n]$, one can construct in polynomial time a main gadget MG_S and an integer M_n^* such that the following three properties hold* [14]*:*

1. *For every $(x,y) \in S$ there is an edge set $E_{x,y} \subseteq E(MG_S)$ of weight M_n^* such that $E_{x,y}$ satisfies the connectedness property and represents (x,y). Moreover, $E_{x,y}$ contains a $t_y \rightsquigarrow b_y$ path and a $\ell_x \rightsquigarrow r_x$ path.*
2. *If there is an edge set $E' \subseteq E(MG_S)$ such that E' has weight at most M_n^* and satisfies the connectedness connectivity property, then E' has weight exactly M_n^* and represents some $(\alpha, \beta) \in S$.*

3.2.2.2 Constructing the SCSS instance

In order to prove Theorem 3.10, we reduce from the GRID TILING problem. The following assumption will be helpful in handling some of the border cases of the gadget

[14]We use the notation M_n^* to emphasize that M^* depends only on n, and not on the set S

construction. We may assume that $1 < x, y < n$ holds for every $(x, y) \in S_{i,j}$: indeed, we can increase n by two and replace every (x, y) by $(x+1, y+1)$ without changing the problem.

Given an instance of GRID TILING, we construct an instance of SCSS the following way (see Figure 3.2):

- We introduce a total of k^2 main gadgets and $2k(k+1)$ connector gadgets.
- For every non-empty set $S_{i,j}$ in the GRID TILING instance, we construct a main gadget $MG_{i,j}$ using Lemma 3.19 for the subset $S_{i,j}$.
- Half of the connector gadgets have the same orientation as in Figure 3.3, and we call them *HCG* to denote *horizontal connector gadgets*. The other half of the connector gadgets are rotated anti-clockwise by 90 degrees with respect to the orientation of Figure 3.3, and we call them *VCG* to denote *vertical connector gadgets*. The internal-distinguished vertices of the connector gadgets are shown in Figure 3.2.
- For each $1 \leq i, j \leq k$, the main gadget $MG_{i,j}$ is surrounded by the following four connector gadgets:

  1. The *horizontal connector gadgets* $HCG_{i,j}$ are on the top and $HCG_{i+1,j}$ are on the bottom. Identify (or glue together) each sink-distinguished vertex of $HCG_{i,j}$ with the top-distinguished vertex of $MCG_{i,j}$ of the same index. Similarly identify each source-distinguished vertex of $HCG_{i+1,j}$ with the bottom-distinguished vertex of $MCG_{i,j}$ of the same index.
  2. The *vertical connector gadgets* $VCG_{i,j}$ are on the left and $VCG_{i,j+1}$ are on the right. Identify (or glue together) each sink-distinguished vertex of $VCG_{i,j}$

with the left-distinguished vertex of $MCG_{i,j}$ of the same index. Similarly identify each source-distinguished vertex of $VCG_{i,j+1}$ with the right-distinguished vertex of $MCG_{i,j}$ of the same index.

- We introduce to special vertices x^*, y^* and an edge (x^*, y^*) of weight 0.
- For each $1 \leq i \leq k$, consider the horizontal connector gadget $HCG_{1,i}$ and collapse all its source-distinguished vertices into y^*.
- For each $1 \leq j \leq k$, consider the vertical connector gadget $VCG_{j,1}$ and collapse all its source-distinguished vertices into y^*.
- For each $1 \leq i \leq k$, consider the horizontal connector gadget $HCG_{k+1,i}$ and collapse all its sink-distinguished vertices into x^*.
- For each $1 \leq j \leq k$, consider the vertical connector gadget $VCG_{j,k+1}$ and collapse all its sink-distinguished vertices into x^*.
- For each $i \in [k+1], j \in [k]$, denote the internal-distinguished vertices of $HCG_{i,j}$ by $\{p^h_{i,j}, q^h_{i,j}\}$
- For each $i \in [k], j \in [k+1]$, denote the internal-distinguished vertices of $VCG_{i,j}$ by $\{p^v_{i,j}, q^v_{i,j}\}$
- The set of terminals T^* for the SCSS instance on G^* is $\{x^*, y^*\} \cup \{p^h_{i,j}, q^h_{i,j} \mid 1 \leq i \leq k+1, 1 \leq j \leq k\} \cup \{p^v_{i,j}, q^v_{i,j} \mid 1 \leq i \leq k, 1 \leq j \leq k+1\}$.
- We note that the total number of terminals is $|T^*| = 4k(k+1) + 2 = O(k^2)$
- The edge set of G^* is a disjoint union of edge sets of all main gadgets, vertical connector gadgets, horizontal gadgets, and the edge (x^*, y^*).

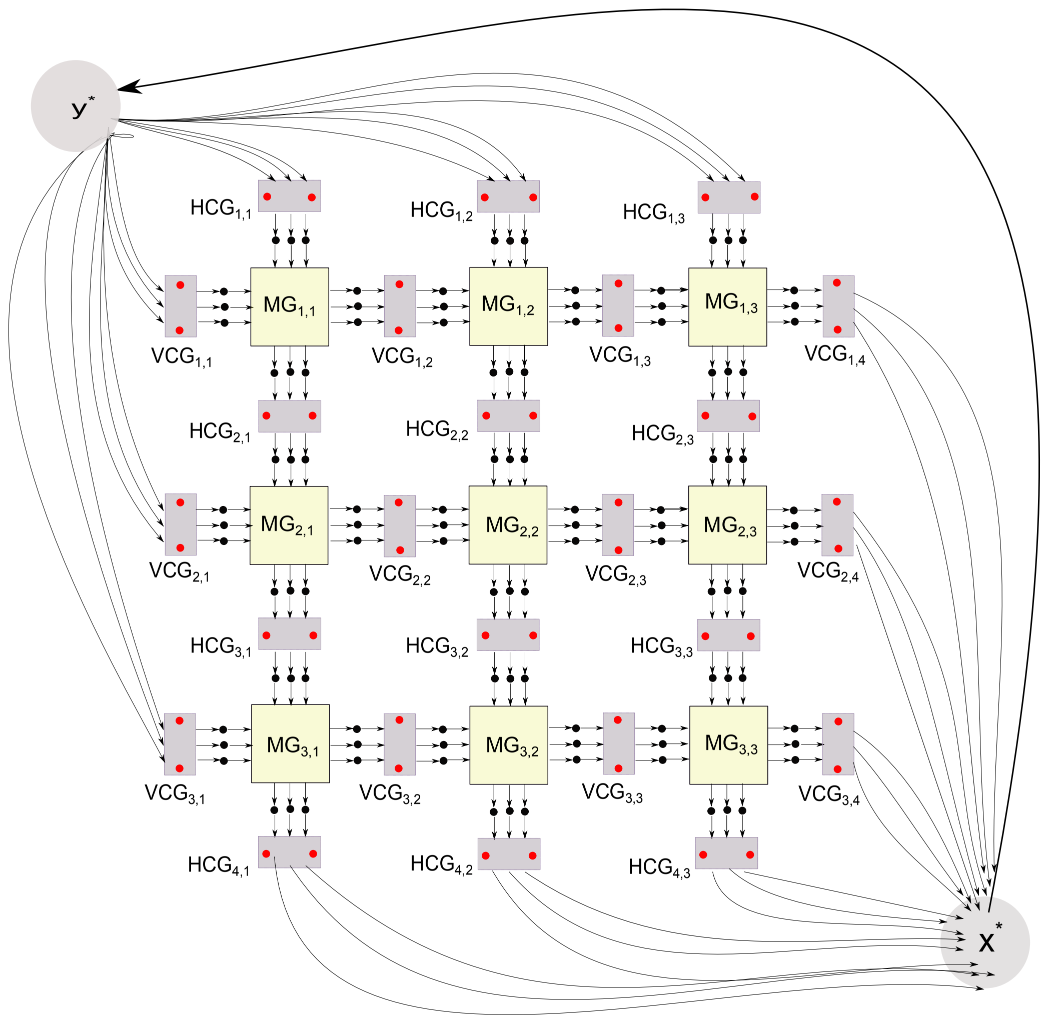


Figure 3.2: The figure for reduction from GRID TILING to SCSS on planar graphs.

Define the following quantity [15]:

$$W_n^* = k^2 \cdot M_n^* + 2k(k+1) \cdot C_n^*. \tag{3.1}$$

In the next two sections, we show that GRID TILING has a solution if and only if the SCSS instance (G^*, T^*) has a solution of weight at most W_n^*.

3.2.2.3 **Finishing the Reduction**

First we show the easy direction: GRID TILING has a solution implies that the SCSS instance (G^*, T^*) constructed in Section 3.2.2.2 has a solution of weight at most W_n^*.

Lemma 3.20. *If the* GRID TILING *instance has a solution, then the SCSS instance (G^*, T^*) has a solution of weight at most W_n^*.*

Proof. Since GRID TILING has a solution, for each $1 \leq i, j \leq k$ there is a value $(x_{i,j}, y_{i,j}) = \gamma_{i,j} \in S_{i,j}$ such that

- For every $i \in [k]$, we have $x_{i,1} = x_{i,2} = x_{i,3} = \ldots = x_{i,k} = \alpha_i$
- For every $j \in [k]$, we have $y_{1,j} = y_{2,j} = y_{3,j} = \ldots = y_{k,j} = \beta_j$

We build a solution E^* for the SCSS instance (G^*, T^*) and show that it has weight at most W_n^*. In the edge set E^*, we take the following edges:

1. The edge (x^*, y^*) which has weight 0.

[15]We use the notation W_n^* to emphasize that W^* depends only on n

2. For each $1 \leq i, j \leq k$ for the main gadget $MG_{i,j}$, use Lemma 3.19(1) to generate a solution $E^M_{i,j}$ which has weight M^*_n and represents (α_i, β_j).
3. For each $1 \leq j \leq k$ and $1 \leq i \leq k+1$ for the horizontal connector gadget, use Lemma 3.18(1) to generate a solution $E^{HC}_{i,j}$ of weight C^*_n which represents β_j.
4. For each $1 \leq i \leq k$ and $1 \leq j \leq k+1$ for the vertical connector gadget, use Lemma 3.18(1) to generate a solution $E^{VC}_{i,j}$ of weight C^*_n which represents α_i.

The weight of E^* is $k^2 \cdot M^*_n + k(k+1) \cdot C^*_n + k(k+1) \cdot C^*_n = W^*_n$. It remains to show that E^* is a solution for the SCSS instance (G^*, T^*). Since we have already picked up the edge (x^*, y^*), it is enough to show that for any terminal $t \in T^* \setminus \{x^*, y^*\}$, both $t \rightsquigarrow x^*$ and $y^* \rightsquigarrow t$ paths exist in E^*. Then for any two terminals t_1, t_2, there is a $t_1 \rightsquigarrow t_2$ path given by $t_1 \rightsquigarrow x^* \rightarrow y^* \rightsquigarrow t_2$.

We only show that the existence of both a $t \rightsquigarrow x^*$ path and a $y^* \rightsquigarrow t$ path in E^* when t is a terminal, say $p^h_{i,j}$, in a horizontal connector gadget. The case when t is a terminal in a vertical connector gadget is similar.

- <u>Existence of $p^h_{i,j} \rightsquigarrow x^*$ path in E^*</u>: By Lemma 3.18(1) $p^h_{i,j}$ can reach the sink-distinguished vertex of $HCG_{i,j}$ which has the index β_j. This vertex is the top-distinguished vertex of the index β_j of the main gadget $MG_{i,j}$. By Lemma 3.19(1) there is a path from this vertex to the bottom-distinguished vertex of the index β_j of the main gadget $MG_{i,j}$. However this vertex is exactly the same as the source-distinguished vertex of the index β_j of $HCG_{i+1,j}$. By Lemma 3.18(1), the source-distinguished vertex of the index β_j of $HCG_{i+1,j}$ can reach the sink-distinguished vertex of the index β

of $HCG_{i+1,j}$. This vertex is exactly the top-distinguished vertex of $MG_{i+1,j}$. Continuing in this way we can reach the source-distinguished vertex of the index β_j of $HCG_{k+1,j}$. By Lemma 3.18(1), this vertex can reach the sink-distinguished vertex of the index β_j of $HCG_{k+1,j}$. But this sink-distinguished vertex was identified with x^*, and hence there is a $p^h_{i,j} \rightsquigarrow x^*$ path in E^*.

- Existence of $y^* \rightsquigarrow p^h_{i,j}$ path in E^*: There is an edge from y^* to the source-distinguished vertex of the index β_j of $HCG_{1,j}$. If $i = 1$, then by Lemma 3.18(1) there is a path from this vertex to $p^h_{1,j}$. If $i \geq 2$, then by Lemma 3.18(1) there is a path from source-distinguished vertex of the index β_j of $HCG_{1,j}$ to the sink-distinguished vertex of the index β_j of $HCG_{1,j}$. But this is the top-distinguished vertex of $MG_{1,j}$ of the index β_j. By Lemma 3.19(1), from this vertex we can reach the bottom-distinguished vertex of the index β_j of $MG_{1,j}$. Continuing this way we can reach the source-distinguished vertex of the index β_j of $HCG_{i,j}$. By Lemma 3.18(1), from this vertex we can reach $p^h_{i,j}$. Hence there is a $y^* \rightsquigarrow p^h_{i,j}$ path in E^*.

$\square$

Now we show the harder direction: the SCSS instance (G^*, T^*) constructed in Section 3.2.2.2 has a solution of weight at most W^*_n implies that GRID TILING has a solution. First we show that the following preliminary claim:

Claim 3.3. *Let E' be any solution to the SCSS instance (G^*, T^*). Then*

- *E' restricted to each connector gadget satisfies the connectedness property (see Definition 3.8).*

- *E' restricted to each main gadget satisfies the connectedness property (see Definition 3.10).*

Proof. First we show that the edge set E' restricted to each connector gadget satisfies the connectedness property. Consider a horizontal connector gadget $HCG_{1,i}$ for some $i \in [k]$. The only incoming edges into $HCG_{1,i}$ are from y^*, and hence y^* must reach both terminals inside $HCG_{1,i}$. Since the only outgoing edges from $HCG_{1,i}$ are the ones to its sink-distinguished vertices, both the terminals inside $HCG_{1,i}$ must be able to reach some sink-distinguished vertex, i.e., $HCG_{1,i}$ satisfies the connectedness property. By very similar reasoning, one can see that all the following types of connector gadgets also satisfy the connectedness property

1. All horizontal connector gadgets $HCG_{k+1,j}$ for $1 \leq j \leq k$: We use the argument that the only outgoing edges for each such gadget are into x^*, and the only incoming edges are from $MG_{k,i}$.
2. All vertical connector gadgets $VCG_{i,1}$ for $1 \leq i \leq k$: We use the argument that the only incoming edges for each such gadget are from y^*, and the only outgoing edges are into $MG_{1,i}$.
3. All vertical connector gadgets $VCG_{j,k+1}$ for $1 \leq j \leq k$: We use the argument that the only outgoing edges for each such gadget are into x^*, and the only incoming edges are from $MG_{j,k}$.

For any connector gadget other than of the four types described above, the only incoming edges are from its source-distinguished vertices and the only outgoing edges are from its sink-distinguished vertices. Hence E' restricted to this connector gadget must satisfy the connectedness property since the two terminals within this connector gadget need to be strongly connected to other terminals.

Now we argue that E' restricted to each main gadget satisfies the connectedness property. Consider a main gadget $MG_{i,j}$. The outgoing edges from the terminals in $HCG_{i,j}$ enter $MG_{i,j}$ via its top-distinguished vertices. These edges can leave $MG_{i,j}$ only through the bottom-distinguished or right-distinguished vertices of $MG_{i,j}$. Hence the first condition of Definition 3.10 is satisfied. Similarly it can be shown the other three conditions of Definition 3.10 also hold, and hence E' restricted to each main gadget satisfies the connectedness property. $\square$

Now we are ready to prove the following lemma:

Lemma 3.21. *If the SCSS instance (G^*, T^*) has a solution say E'' of weight at most W_n^*, then the* Grid Tiling *instance has a solution.*

Proof. By Claim 3.3, the edge set E'' restricted to any connector gadget satisfies the connectedness property and the edge set E'' restricted to any main gadget satisfies the connectedness property. Let $\mathcal{C}$ and $\mathcal{M}$ be the sets of connector and main gadgets respectively. Recall that $|\mathcal{C}| = 2k(k+1)$ and $|\mathcal{M}| = k^2$. Recall that we have defined W_n^* as $k^2 \cdot M_n^* + 2k(k+1)C_n^*$. Let $\mathcal{C}' \subseteq \mathcal{C}$ be the set of connector gadgets that have weight at most C_n^* in E''. By Lemma 3.18(2), each connector gadget from the set $\mathcal{C}'$ has weight exactly C_n^*. Since all weights are positive integers, each connector gadget from the set $\mathcal{C} \setminus \mathcal{C}'$ has weight at least $C_n^* + 1$. Similarly, let $\mathcal{M}' \subseteq \mathcal{M}$ be the set of main gadgets which have weight at most M_n^* in E''. By Lemma 3.19(2), each main gadget from the set $\mathcal{M}'$ has weight exactly M_n^*. Since all weights are positive integers, each main gadget from the set $\mathcal{M} \setminus \mathcal{M}'$ has weight at least $M_n^* + 1$. As any two gadgets are pairwise edge-disjoint, we have

$$\begin{aligned}
W_n^* &= k^2 \cdot M_n^* + 2k(k+1)C_n^* \\
&\geq |\mathcal{M} \setminus \mathcal{M}'| \cdot (M_n^* + 1) + |\mathcal{M}'| \cdot M_n^* + |\mathcal{C} \setminus \mathcal{C}'| \cdot (C_n^* + 1) + |\mathcal{C}'| \cdot C_n^* \\
&= |\mathcal{M}| \cdot M_n^* + |\mathcal{C}| \cdot C_n^* + |\mathcal{M} \setminus \mathcal{M}'| + |\mathcal{C} \setminus \mathcal{C}'| \\
&= k^2 \cdot M_n^* + 2k(k+1) \cdot C_n^* + |\mathcal{M} \setminus \mathcal{M}'| + |\mathcal{C} \setminus \mathcal{C}'| \\
&= W_n^* + |\mathcal{M} \setminus \mathcal{M}'| + |\mathcal{C} \setminus \mathcal{C}'|.
\end{aligned}$$

This implies $|\mathcal{M} \setminus \mathcal{M}'| = 0 = |\mathcal{C} \setminus \mathcal{C}'|$. However, we had $\mathcal{M}' \subseteq \mathcal{M}$ and $\mathcal{C}' \subseteq \mathcal{C}$. Therefore, $\mathcal{M}' = \mathcal{M}$ and $\mathcal{C}' = \mathcal{C}$. Hence in E'', each connector gadget has weight C_n^* and each main gadget has weight M_n^*.

From Lemma 3.18(2) and Lemma 3.19(2), we have

- For each horizontal connector gadget $HCG_{i,j}$, the restriction of the edge set E'' to $HCG_{i,j}$ represents an integer say $\beta_{i,j} \in [n]$ where $i \in [k+1], j \in [k]$.
- For each vertical connector gadget $VCG_{i,j}$, the restriction of the edge set E'' to $VCG_{i,j}$ represents an integer say $\alpha_{i,j}$ where $i \in [k], j \in [k+1]$.
- For each main gadget $MG_{i,j}$, the restriction of the edge set E'' to $MG_{i,j}$ represents an ordered pair say $(\alpha'_{i,j}, \beta'_{i,j}) \in S_{i,j}$ where $i, j \in [k]$.

Consider the main gadget $MG_{i,j}$ for any $1 \leq i, j \leq k$. We can make the following observations:

- $\underline{\beta_{i,j} = \beta'_{i,j}}$: By Lemma 3.18(2) and Definition 3.9, the terminal vertices in $HCG_{i,j}$ can exit the horizontal connector gadget only via the unique edge entering the sink-distinguished vertex of index $\beta_{i,j}$. By Lemma 3.19(2) and Definition 3.11, the only edge in E'' incident to any top-distinguished vertex of $MG_{i,j}$ is the unique edge leaving the top-distinguished vertex of the index $\beta'_{i,j}$. Hence if $\beta_{i,j} \neq \beta'_{i,j}$ then the terminals in $HCG_{i,j}$ will not be able to reach other terminals.

- $\underline{\beta'_{i,j} = \beta_{i+1,j}}$: By Lemma 3.18(2) and Definition 3.9, the unique edge entering $HCG_{i+1,j}$ is the edge entering the source-distinguished vertex of the index $\beta_{i+1,j}$. By Lemma 3.19(2) and Definition 3.11, the only edge in E'' incident to any bottom-distinguished vertex of $MG_{i,j}$ is the unique edge entering the bottom-distinguished vertex of index $\beta'_{i,j}$. Hence if $\beta'_{i,j} \neq \beta_{i+1,j}$, then the terminals in $HCG_{i+1,j}$ cannot be reached from the other terminals.

- $\underline{\alpha_{i,j} = \alpha'_{i,j}}$: By Lemma 3.18(2) and Definition 3.9, the paths starting at the terminal vertices in $VCG_{i,j}$ can leave the vertical connector gadget only via the unique edge entering the sink-distinguished vertex of index $\alpha_{i,j}$. By Lemma 3.19(2) and Definition 3.11, the only edge in E'' incident to any left-distinguished vertex of $MG_{i,j}$ is the unique edge leaving the left-distinguished vertex of the index $\alpha'_{i,j}$. Hence if $\alpha_{i,j} \neq \alpha'_{i,j}$ then the terminals in $VCG_{i,j}$ will not be able to reach other terminals.

- $\underline{\alpha'_{i,j} = \alpha_{i,j+1}}$: By Lemma 3.18(2) and Definition 3.9, the unique edge entering $VCG_{i,j+1}$ is the edge entering the source-distinguished vertex of index $\alpha_{i,j+1}$. By

Lemma 3.19(2) and Definition 3.11, the only edge in E'' incident to any right-distinguished vertex of $MG_{i,j}$ is the unique edge entering the bottom-distinguished vertex of index $\alpha'_{i,j}$. Hence if $\alpha'_{i,j} \neq \alpha_{i,j+1}$, then the terminals in $VCG_{i,j+1}$ cannot be reached from the other terminals.

We claim that for $1 \leq i,j \leq k$, the values $(\alpha'_{i,j}, \beta'_{i,j}) \in S_{i,j}$ form a solution for the GRID TILING instance. For this we need to check two conditions:

- $\underline{\alpha'_{i,j} = \alpha'_{i,j+1}}$: This holds because $\alpha_{i,j} = \alpha'_{i,j} = \alpha_{i,j+1} = \alpha'_{i,j+1}$.
- $\underline{\beta'_{i,j} = \beta'_{i+1,j}}$: This holds because $\beta_{i,j} = \beta'_{i,j} = \beta_{i+1,j} = \beta'_{i+1,j}$.

This completes the proof of the lemma. □

Finally we are ready to prove Theorem 3.10, which is restated below. Recall that Marx [Marc] showed the W[1]-hardness of GRID TILING parameterized by k.

Theorem 3.10 . *The* STRONGLY CONNECTED STEINER SUBGRAPH *problem restricted to the case when the underlying undirected graph is planar is W[1]-hard parameterized by the number of terminals, and cannot be solved in time $f(k)n^{o(\sqrt{k})}$ unless ETH fails, where f is any computable function, k is the number of terminals, and n is the number of vertices in the instance.*

Proof. We note that the number of terminals in the SCSS instance is $2k(k+1)+2 = O(k^2)$. By Lemmas 3.18 and 3.19, the connector and main gadgets are constructed in polynomial time, hence their size can be bounded by a polynomial in n. It follows that the constructed instance has polynomial size. It is easy to see the underlying undirected graph of G^* constructed in Figure 3.2 is planar, since the underlying graph of each connector gadget (see Figure 3.3) and each main gadget (see Figure 3.4) is planar. Lemma 3.20 and Lemma 3.21 together imply the W[1]-hardness of SCSS parameterized by the number of terminals, even when the underlying graph is planar.

Chen et al. [CHKX06] showed for any function f an $f(k) \cdot n^{o(k)}$ algorithm for CLIQUE implies ETH fails. Marx [Marc] gave a reduction that transforms the problem of finding a k-clique into a $k \times k$ GRID-TILING instance. Lemma 3.20 and Lemma 3.21 together give a reduction which transforms the problem of $k \times k$ GRID TILING into an instance of SCSS with $O(k^2)$ terminal pairs. Composing the two reductions, we obtain that, under ETH, there is no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm for SCSS (even when the underlying undirected graph is planar) for any function f. This shows that the $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$ algorithm for SCSS given in Theorem 3.1 is essentially optimal. $\square$

3.2.2.4 Proof of Lemma 3.18: Constructing the Connector Gadget

We prove Lemma 3.18 in this section, by constructing a connector gadget satisfying the specifications of Section 3.2.2.1.

3.2.2.4.1 Different types of edges in connector gadget

Before proving Lemma 3.18, we first describe the construction of the connector gadget in more detail (see Figure 3.3). The connector gadget has $2n+4$ rows denoted by $R_0, R_1, R_2, \ldots, R_{2n+3}$ and $4n+1$ columns denoted by $C_0, C_1, \ldots, C_{4n}$. Let us denote the vertex at the intersection of row R_i and column C_j by v_i^j. We now describe the different kinds of edges present in the connector gadget.

1. **Source Edges**: For each $i \in [n]$, there is an edge (p_i, v_0^{2i-1}). These edges are together called as the source edges.

2. **Sink Edges**: For each $j \in [n]$, there is an edge $(v_{2n+3}^{2n+2i-1}, q_i)$. These edges are together called as the sink edges.

3. **Terminal Edges**: We call the union of the sets of edges incident to the terminals p or q as terminal edges. The set of edges incident on p is $\{(v_{2i}^0, p) \ : \ 0 \leq i \leq n+1\} \bigcup \{(p, v_{2i+1}^0) \ : \ 0 \leq i \leq n+1\}$. The set of edges incident on q is $\{(v_{2i}^{4n}, q) \ : \ 0 \leq i \leq n+1\} \bigcup \{(q, v_{2i+1}^{4n}) \ : \ 0 \leq i \leq n+1\}$.

4. **Inrow Edges**:

   - Inrow Up Edges: For each $0 \leq i \leq n+1$, we call the $\uparrow$ edges connecting vertices of row R_{2i+1} to R_{2i} as inrow up edges. An example of an inrow up edge is (v_1^0, v_0^0).

   - Inrow Left Edges: For each $0 \leq i \leq 2n+3$, we call the $\leftarrow$ edges connecting vertices of row R_i as inrow left edges. An example of an inrow left edge is (v_3^1, v_3^0).

   - Inrow Right Edges: For each $0 \leq i \leq 2n+3$, we call the $\rightarrow$ edges connecting vertices of row R_i as inrow right edges. An example of an inrow right edge is (v_0^0, v_0^1).

   - Inrow Down Edges: For each $0 \leq i \leq n+1$, we call the $\downarrow$ edges connecting vertices of row R_{2i} to R_{2i+1} as inrow down edges. An example of an inrow down edge is (v_0^1, v_1^1).

5. **Interrow Edges**: For each $i \in [n+1]$ and each $j \in [2n]$, we subdivide the edge $(v_{2i-1}^{2j-1}, v_{2i}^{2j-1})$ by introducing a new vertex $w_{i,j}$ and adding the edges (v_{2i-1}^{2j-1}, w_i^j) and (w_i^j, v_{2i}^{2j-1}). All these edges are together called as interrow edges. Note that there are $4n(n+1)$ interrow edges.

6. **Shortcuts**: There are $2n$ shortcut edges, namely $e_1, e_2, \ldots, e_n$ and $f_1, f_2, \ldots, f_n$. They are drawn as follows:

   - The edge e_i is given by $(v_{2n-2i+2}^{2i-2}, w_{n-i+1}^{i})$.
   - The edge f_i is given by $(w_{n-i+2}^{n+i}, v_{2n-2i+3}^{2n+2j})$.

3.2.2.4.2 Assigning Weights in the Connector Gadget

Fix the quantity $B = 18n^2$. We assign weights to the edges as follows

1. For $i \in [n]$, the source edge (p_i, v_0^{2i-1}) has weight $B^5 + (n-i+1)$.
2. For $i \in [n]$, the sink edge $(v_{2n+3}^{2n+2i-1}, q_i)$ has weight $B^5 + i$.
3. Each terminal edge has weight B^4.
4. Each inrow up edge has weight B^3.
5. For each $i \in [n+1]$ and each $j \in [2n]$, the edge (v_{2i-1}^{2j-1}, w_i^j) has weight 0, and the edge (w_i^j, v_{2i}^{2j-1}) has weight B^2.
6. Each inrow right edge has weight B.

7. For each $i \in [n]$, the edge e_i has weight $n \cdot i$.

8. For each $j \in [n]$, the edge f_j has weight $n(n-j+1)$.

9. Each inrow left edge and inrow down edge has weight 0.

Now we define the quantity C_n^* stated in statement of Lemma 3.18:

$$C_n^* = 2B^5 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-2)B + (n+1)^2. \tag{3.2}$$

In the next two sections, we prove the two statements of Lemma 3.18.

3.2.2.4.3 $i \in [n] \Rightarrow \exists$ a solution E_i of weight C_n^* that satisfies connectedness property & represents i

Let E_i be the union of the following sets of edges:

- Select the edges (p_i, v_0^{2i-1}) and $(v_{2n+3}^{2n+2i-1}, q_i)$. This incurs a weight of $B^5 + (n-i+1) + B^5 + i = 2B^5 + (n+1)$.
- The two terminal edges $(p, v_{2n-2i+2}^0)$ and $(v_{2n-2i+1}^0, p)$. This incurs a weight of $2B^4$.
- The two terminal edges $(q, v_{2n-2i+2}^{4n})$ and $(v_{2n-2i+1}^{4n}, q)$. This incurs a weight of $2B^4$.
- All the $2n+1$ inrow up edges that are between vertices of $R_{2n-2i+2}$ and $R_{2n-2i+3}$. These edges are given by $(v_{2n-2i+3}^{2j}, v_{2n-2i+2}^{2j})$ for $0 \le j \le 2n$. This incurs a weight of $(2n+1)B^3$.
- The vertically downward path $P_1 = v_0^{2i-1} \to v_1^{2i-1} \to \ldots \to v_{2n-2i+1}^{2i-1} \to w_{n-i+1}^i \to v_{2n-2i+2}^{2i-1} \to v_{2n-2i+3}^{2i-1}$ and the vertically downward path $P_2 = v_{2n-2i+2}^{2n+2i-1} \to v_{2n-2i+3}^{2n+2i-1} \to$

$h_i \to v_{2n-2i+4}^{2n+2i-1} \to \ldots \to v_{2n+2}^{2n+2i-1} \to v_{2n+3}^{2n+2i-1}$. These two paths incur a weight of $(n+1)B^2$, since the inrow down edges have weight 0.

- All $2n$ inrow right edges and $2n$ inrow left edges which occur between vertices of $R_{2n-2i+2}$. This incurs a weight of $2n \cdot B$ since inrow left edges have weight 0 and each inrow right edge has weight B.
- All inrow right edges and inrow left edges which occur between vertices of $R_{2n-2i+3}$. This incurs a weight of $2n \cdot B$ since inrow left edges have weight 0 and there are $2n$ such inrow right edges each having weight B.
- The edges e_i and f_i. This incurs a weight of $n \cdot i + n(n-i+1) = n(n+1)$.
- All $2n$ inrow down edges that occur between vertices of row $R_{2n-2i+2}$ and row $R_{2n-2i+3}$. This incurs a weight of 0, since the each inrow down edge has weight 0.

Finally, remove the two inrow right edges $(v_{2n-2i+2}^{2i-2}, v_{2n-2i+2}^{2i-1})$ and $(v_{2n-2i+3}^{2n+2i-1}, v_{2n-2i+3}^{2n+2i})$ from E'. This saves a weight of $2B$. From the above paragraph and Equation 3.2 it follows that the total weight of E_i is exactly C_n^*. Note that we can still travel from $v_{2n-2i+2}^{2i-2}$ to $v_{2n-2i+2}^{2i-1}$ as follows: take the path $v_{2n-2i+2}^{2i-2} \to w_{n-i+1}^{i} \to v_{2n-2i+2}^{2i-1}$. Similarly, even though we removed the edge $(v_{2n-2i+3}^{2n+2i-1}, v_{2n-2i+3}^{2n+2i})$ we can still travel from $v_{2n-2i+3}^{2n+2i-1}$ to $v_{2n-2i+3}^{2n+2i}$ in E' via the path $v_{2n-2i+3}^{2n+2i-1} \to w_{n-i+2}^{n+i} \to v_{2n-2i+3}^{2n+2i}$.

It remains to show that E_i satisfies the connectedness property and it represents i. It is easy to see E_i represents i since the only edge in E_i which is incident to P is the edge leaving p_i. Similarly, the only edge in E_i incident to Q is the one entering q_i. We show that the connectedness property holds as follows:

1. There is a $p_i \rightsquigarrow p$ path in E_i by starting with the source edge leaving p_i and then following downward path P_1 from $v_0^{2i-1} \rightsquigarrow v_{2n-2i+3}^{2i-1}$. Then travel towards the left from $v_{2n-2i+3}^{2i-1}$ to p by using inrow left, inrow up and inrow down edges from rows $R_{2n-2i+2}$ and $R_{2n-2i+3}$. Finally, use the edge $(v_{2n-2i+2}^{0}, p)$

2. For the existence of a $p_i \rightsquigarrow q$ path in E_i, we have seen above that there is a $p_i \rightsquigarrow v_{2n-2i+3}^{2i-1}$ path. Then travel towards the right from $v_{2n-2i+2}^{2i-1}$ to q by using inrow right, inrow up and inrow down edges from rows $R_{2n-2i+2}$ and $R_{2n-2i+3}$ (and one shortcut edge as well). Finally, use the edge $(v_{2n-2i+2}^{4n}, q)$.

3. For the existence of a $p \rightsquigarrow q_i$ path in E_i, first use the edge $(p, v_{2n-2i+3}^{0})$. Then travel towards the right by using inrow right, inrow up and inrow down edges from rows $R_{2n-2i+2}$ and $R_{2n-2i+3}$ (and one shortcut edge as well) until you reach the vertex $v_{2n-2i+2}^{2n+2i-1}$. Then take the downward path P_2 from $v_{2n-2i+2}^{2n+2i-1}$ to $v_{2n+3}^{2n+2i-1}$. Finally, use the sink edge incident to q_i.

4. For the existence of a $q \rightsquigarrow q_i$ path in E_i, first use the terminal edge $(q, v_{2n-2i+3}^{4n})$. Then travel towards the left by using inrow left, inrow up and inrow down edges from rows $R_{2n-2i+2}$ and $R_{2n-2i+3}$ until you reach the vertex $v_{2n-2i+2}^{2n+2i-1}$. Then take the downward path P_2 from $v_{2n-2i+2}^{2n+2i-1}$ to $v_{2n+3}^{2n+2i-1}$. Finally, use the sink edge incident to q_i.

Therefore, E_i indeed satisfies the connectedness property.

3.2.2.4.4 E' satisfies connectedness property and has weight $\leq C_n^* \Rightarrow E'$ represents some $\beta \in [n]$ and has weight exactly C_n^*

Next we show that if a set of edges E' satisfies the connectedness property and has weight at most C_n^*, then in fact the weight of E' is exactly C_n^* and it represents some $\beta \in [n]$. We do this via the following series of claims and observations.

Claim 3.4. *E' contains exactly one source edge and one sink edge.*

Proof. Since E' satisfies the connectedness property it must contain at least one source edge and at least one sink edge. Without loss of generality, suppose that there are at least two source edges in E'. Then the weight of E' is a least the sum of the weights of these two source edges plus the weight of at least one sink edge.

Thus if E' contains at least two source edges, then its weight is at least $3B^5$. However, from Equation 3.2 we get that

$$\begin{aligned} C_n^* &= 2B^5 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-2)B + (n+1)^2 \\ &\leq 2B^5 + 4n \cdot B^4 + 3n \cdot B^4 + 2n \cdot B^4 + 4n \cdot B^4 + 4n \cdot B^4 \\ &\leq 2B^5 + 17n \cdot B^4 \\ &< 3B^5, \end{aligned}$$

since $B = 18n^2 > 17n$.

□

Thus we know that E' contains exactly one source edge and exactly one sink edge. Let the source edge be incident to $p_{i'}$ and the sink edge be incident to $q_{j'}$.

Claim 3.5. *E' contains exactly four terminal edges.*

Proof. Since E' satisfies the connectedness property, it must contain at least one incoming and one outgoing edge for both p and q. Therefore, we need at least four terminal edges. Suppose we have at least five terminal edges in E'. We already know that the source and sink edges contribute at least $2B^5$ to weight of E', hence the of E' is at least $2B^5 + 5B^4$. However, from Equation 3.2, we get that

$$\begin{aligned}
C_n^* &= 2B^5 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-2)B + (n+1)^2 \\
&\leq 2B^5 + 4B^4 + 3n \cdot B^3 + 2n \cdot B^3 + 4n \cdot B^3 + 4n \cdot B^3 \\
&= 2B^5 + 4B^4 + 13n \cdot B^3 \\
&< 2B^5 + 5B^4,
\end{aligned}$$

since $B = 18n^2 > 13n$. □

Hence we know that E' contains exactly four terminal edges.

Claim 3.6. *E' contains exactly $2n+1$ inrow up edges.*

Proof. Observe that for each $1 \leq j \leq 2n-1$, the inrow up edges in column C_{2j} form a cut between vertices from columns C_{2j-1} and C_{2j+1}. Since E' must have a $p_{i'} \rightsquigarrow p$ path, we need to use at least one inrow up edge from each of the columns $C_0, C_2, \ldots, C_{2i'-2}$. Since E' must have a $p_{i'} \rightsquigarrow q$, path we need to use at least one inrow up edge from each of the columns $C_{2i'}, C_{2i'+2}, \ldots, C_{4n}$. Hence E' has at least $2n+1$ inrow up edges, as we require at least one inrow up edge from each of the columns $C_0, C_2, \ldots, C_{4n}$.

Suppose E' contains at least $2n+2$ inrow up edges. We already know that E' has a contribution of $2B^5+4B^4$ from source, sink, and terminal edges. Hence the weight of E' is at least $2B^5+4B^4+(2n+2)B^3$. However, from Equation 3.2, we get that

$$\begin{aligned}
C_n^* &= 2B^5+4B^4+(2n+1)B^3+(n+1)B^2+(4n-2)B+(n+1)^2 \\
&\leq 2B^5+4B^4+(2n+1)B^3+2n\cdot B^2+4n\cdot B^2+4n\cdot B^2 \\
&= 2B^5+4B^4+(2n+1)B^3+10n\cdot B^2 \\
&< 2B^5+4B^4+(2n+2)B^3,
\end{aligned}$$

since $B=18n^2>10n$. □

Therefore, we know that E' contains exactly one inrow edge per column C_{2i} for every $0\leq i\leq 2n$. By Claim 3.5, we know that exactly two terminal edges incident to p are selected in E'. Observe that the terminal edge leaving p should be followed by an inrow up edge, and similarly, the terminal edge entering p follows an inrow up edge. Since we select exactly one inrow up edge from column C_0, it follows that the two terminal edges in E' incident to p must be incident to the rows $R_{2\ell+1}$ and $R_{2\ell}$ respectively for some $\ell\in[n]$. Similarly, the two terminal edges in E' incident to q must be incident to the rows $R_{2\ell'+1}$ and $R_{2\ell'}$ for some $\ell'\in[n]$. We summarize this in the following claim:

Observation 3.11. *There exist integers $\ell,\ell'\in[n]$ such that*

- *the only two terminal edges in E' incident to p are $(p,v^0_{2\ell+1})$ and $(v^0_{2\ell},p)$, and*
- *the only two terminal edges in E' incident to q are $(q,v^{4n}_{2\ell'+1})$ and $(v^{4n}_{2\ell'},q)$.*

Definition 3.12. *For $i \in [n+1]$, we call the $4n$ interrow edges which connect vertices from row R_{2i-1} to vertices from R_{2i} as Type(i) interrow edges. We can divide the Type(i) interrow edges into $2n$ pairs of adjacent interrow edges given by (v_{2i-1}^{2j-1}, w_i^j) and (w_i^j, v_{2i}^{2j-1}) for each $1 \leq j \leq 2n$*

Note that there are a total of $n+1$ types of interrow edges. Moreover, we can assume that if we select one interrow edge, then we also pick up the other adjacent interrow edge in the "pair," otherwise there is no use for selecting the first interrow edge (even though it has weight 0).

Claim 3.7. *E' contains exactly one pair of interrow edges of Type(r) for each $r \in [n+1]$.*

Proof. Observation 3.11 implies that we cannot avoid using interrow edges of any type by, say, going into p via an edge from some R_{2i} and then exiting p via an edge to some R_{2j+1} for any $j > i$ (similarly for q). By the connectedness property, set E' contains a $p_{i'} \rightsquigarrow p$ path. By Observation 3.11, the only edge entering p is $(v_{2\ell}^0, p)$. Hence E' must contain at least one pair of interrow edges of Type(r) for $1 \leq r \leq \ell$. Similarly E' contains a $p \rightsquigarrow q_i$ path and the only outgoing edge from p is $(p, v_{2\ell+1}^0)$. Hence E' must contain at least one pair of interrow edges of Type(r) for each $\ell+1 \leq r \leq n+1$. Therefore, set E' contains at least one pair of interrow edges of each type.

Suppose E' contains at least two pairs of interrow edges of some Type(r) for some $r \in [n+1]$. This means E' gets a weight of least $(n+2) \cdot B^2$ from the interrow edges. We have already seen E' has contribution of $2B^5 + 4B^4 + (2n+1)B^3$ from source, sink, terminal, and inrow up edges. Hence the weight of E' is at leas $2B^5 + 4B^4 + (2n+1)B^3 +$

$(n+2)\cdot B^2$. However, from Equation 3.2, we get that

$$\begin{aligned} C_n^* &= 2B^5+4B^4+(2n+1)B^3+(n+1)B^2+(4n-2)B+(n+1)^2 \\ &\leq 2B^5+4B^4+(2n+1)B^3+(n+1)B^2+4n\cdot B+4n\cdot B \\ &= 2B^5+4B^4+(2n+1)B^3+(n+1)B^2+8n\cdot B \\ &< 2B^5+4B^4+(2n+1)B^3+(n+2)B^2, \end{aligned}$$

since $B = 18n^2 > 8n$. □

Claim 3.8. *For each $r \in [n+1]$, let the unique pair of interrow edges in E' (guaranteed by Claim 3.7) of Type(r) belong to column C_{2i_r-1}. If the unique source and sink edges in E' (guaranteed by Claim 3.4) are incident to $p_{i'}$ and $q_{j'}$, respectively, then we have $i' \leq i_1 \leq i_2 \leq \ldots \leq i_{n+1} \leq n+j'$.*

Proof. Observation 3.11 implies the only way to get from row R_{2i} to R_{2i+1} is to use a pair of interrow edges of Type(i). By Claim 3.7, we use exactly one pair of interrow edges of each type. Recall that there is a path $P = p_{i'} \rightsquigarrow p \rightsquigarrow q_{j'}$ in E', and this path must use each of these interrow edges.

First we show that $i_1 \geq i'$. Suppose $i_1 < i' \leq n$. Since we use the source edge incident to $p_{i'}$, we must reach vertex $v_0^{2i'-1}$. Since $i' > i_1$, to use the pair of interrow edges to travel from $v_1^{2i_1-1}$ to $v_2^{2i_1-1}$, the path P must contain a $v_0^{2i'-1} \rightsquigarrow v_1^{2i_1-1}$ subpath say P'. By the construction of the connector gadget this subpath P' must use the inrow up edge $(v_1^{2i'-2}, v_0^{2i'-2})$. Now the path P has to reach column $C_{2n+2j'-1}$ from column C_{2i_1-1}, and so it must use another inrow edge from column $C_{2i'-2}$, which contradicts Claim 3.6.

Now we prove $i_{n+1} \leq n + j'$. Suppose to the contrary that $i_{n+1} > n + j'$. Then by reasoning similar to that of above one can show that at least two inrow up edges from column $C_{2n+2j'}$ are used, which contradicts Claim 3.6.

Finally suppose there exists $r \in [n]$ such that $i_r > i_{r+1}$. We consider the following three cases:

- $\underline{i_{r+1} < i_r \leq n}$: Then by using the fact that there is a $p_{i'} \rightsquigarrow q_{j'}$ path in E' we get at least two inrow up edges are used from column C_{2i_r-2}, which contradicts Claim 3.6.
- $\underline{n < i_r \leq n + j'}$: Then we need to use at least two inrow up edges from column C_{2i_r-2}, which contradicts Claim 3.6.
- $\underline{i_r > n + j'}$: Then we need to use at least two inrow up edges from column $C_{2n+2j'}$, which contradicts Claim 3.6.

□

Claim 3.9. *E' contains at most two shortcut edges.*

Proof. For the proof we will use Claim 3.8. We will show that we can use at most one e-shortcut. The proof for f-shortcut is similar.

Suppose we use two e-shortcuts viz. e_x and e_y such that $x > y$. Note that it makes sense to include a shortcut into E' only if we use the interrow edge that continues it. Hence $i_x = x$ and $i_y = y$. By Claim 3.8, we have $y = i_y \geq i_x = x$, which is a contradiction.

□

Claim 3.10. *E' contains exactly $4n - 2$ inrow right edges.*

Proof. E' contains a $p \rightsquigarrow q_{j'}$ path. Thus E' has an inrow right edge between columns C_i, C_{i+1} for each $0 \le i \le 2n + 2j' - 2$. Similarly E' contains a $p_{i'} \rightsquigarrow q$ path. So E' has an inrow right edge between columns C_j, C_{j+1} for each $2i' - 1 \le i \le 4n - 1$.

Since $2n + 2j' - 2 \ge 2n$ and $2i' - 1 \le 2n$ it follows E' must contain at least $4n$ inrow right edges. However, as we have seen before, a shortcut can replace an inrow right edge as follows: for example the inrow edge (v_2^{2n-2}, v_2^{2n-1}) can be replaced by the path $v_2^{2n-2} \to w_1^n \to v_2^{2n-1}$. But Claim 3.9 implies we can use at most two shortcuts. Therefore we need to use at least $4n-2$ inrow right edges. Suppose E' contains at least $4n-1$ edges. We have already seen the contribution of source, sink, terminal, inrow up and interrow edges is $2B^5 + 4B^4 + (2n+1)B^3 + (n+1)B^2$. If E' contains at least $4n-1$ inrow right edges, then the weight of E' is at least $2B^5 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-1)B$.

However, from Equation 3.2, we get that

$$\begin{aligned} C_n^* &= 2B^5 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-2)B + (n+1)^2 \\ &= 2B^5 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-2)B + 4n^2 \\ &< 2B^5 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-1)B, \end{aligned}$$

since $B = 18n^2 > 4n^2$.

□

From Claim 3.9 and the proof of Claim 3.10, we can make the following observation:

Observation 3.12. *E' contains exactly two shortcuts.*

Let the shortcuts used be $e_{i''}$ and $f_{j''}$. Recall that Claim 3.4 implies that at most one edge incident to P and at most one edge incident to Q is used in E'. Therefore, if we show that $i' = j'$, then it follows that E' represents $\beta = i' = j'$.

Claim 3.11. *The following inequalities hold:*

- $i'' \geq i'$ *and* $j'' \leq j'$
- $i'' \geq j''$

Proof. To use the shortcut $e_{i''}$, we need to use the lower half of a pair of interrow edge from column $C_{2i''-1}$. Claim 3.8 implies $i' \leq i_1$ and the pairs of interrow edges used are monotone from left to right. Hence $i'' \geq i'$. Similarly, to use the shortcut $f_{j''}$, we need to use the upper half of an interrow edge from Column $C_{2n+2j''-1}$. Claim 3.8 implies $n + j' \geq i_{n-1} \geq n + j''$. Hence $j'' \leq j'$.

Now we prove $i'' \geq j''$. Since both $i'', j'' \in [n]$, if either $i'' = n$ or $j'' = 1$, then we are done. So suppose $i'' \leq n-1$ and $j'' \geq 2$. To use the shortcut edge $e_{i''}$ after entering the connector gadget through the source edge incident to $p_{i'}$, we must use pairs of interrow edges of Type(1), Type(2), …, Type$(n - i'')$. To use the shortcut edge $f_{j''}$ and exit the connector gadget through the sink edge incident to $b_{j'}$, we must use pairs of interrow edges of Type$(n+1)$, Type(n), …, Type$(n - j'' + 1)$. If $i'' < j''$, then this implies we use at least two pairs of interrow edges of Type$(n - i'')$, which contradicts Claim 3.7. Therefore, $i'' \geq j''$ holds. □

Theorem 3.13. *The weight of E' is exactly C_n^*, and E' represents some integer $\beta \in [n]$.*

Proof. As argued above it is enough to show that $i' = j'$. We have already seen E' has the following contribution to its weight:

- The source edge incident to $p_{i'}$ has weight $B^5 + (n - i' + 1)$ by Claim 3.4.
- The sink edge incident to $q_{j'}$ has weight $B^5 + j'$ by Claim 3.4.
- The terminal edges incur weight $4B^4$ by Claim 3.5.
- The inrow up edges incur weight $(2n+1)B^3$ by Claim 3.6.
- The interrow edges incur weight $(n+1)B^2$ by Claim 3.7.
- The inrow right edges incur weight $(4n-2)B$ by Claim 3.10.
- The shortcut $e_{i''}$ incurs weight $n \cdot i''$ and $f_{j''}$ incurs weight $n(n - j'' + 1)$ by Claim 3.12.

Thus we already have a weight of

$$C^{**} = (2B^5 + (n - i' + j' + 1)) + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-2)B + n(n - j'' + i'' + 1) \tag{3.3}$$

Observe that adding any edge of non-zero weight to E' (other than the ones mentioned above) increases the weight C^{**} by at least B, since Claim 3.9 does not allow us to use any more shortcuts. Equation 3.2 and Equation 3.3 imply $C^{**} + B - C_n^* = B - n(i' - j') - (j'' - i'') \geq 20n^3 - n(i' - j') - (j'' - i'') \geq 0$, since $i', i'', j', j'' \in [n]$. This implies that the weight of E' is exactly C^{**}. We now show that in fact $C^{**} - C_n^* \geq 0$, which will imply that $C^{**} = C_n^*$. From Equation 3.2 and Equation 3.3, we have $C^{**} - C_n^* = (j' - i') + n(i'' - j'')$. We now show that this quantity is non-negative. Recall that from Claim 3.11, we have $i'' \geq j''$.

- If $i'' > j''$ then $n(i''-j'') \geq n$. Since $j', i' \in [n]$, we have $j'-i' \geq 1-n$. Therefore, $(j'-i')+n(i''-j'') \geq n+(1-n) = 1$
- If $i'' = j''$ then by Claim 3.11 we have $i' \leq i'' = j'' \leq j'$. Hence $(j'-i') \geq 0$ and so $(j'-i')+n(i''-j'') \geq 0$.

Therefore $C^{**} = C_n^*$, since we were given that the weight of E' is C^{**}. However $C_n^* = C^{**}$ implies

$$j'-i' \leq n(j''-i'') \tag{3.4}$$

Since $i', j' \in [n]$ we have $n-1 \geq j'-i' \geq 1-n$. Therefore Equation 3.4 implies $j'' = i''$ and $j' = i'$.

□

3.2.2.5 Proof of Lemma 3.19: Constructing the Main Gadget

We prove Lemma 3.19 in this section, by constructing a main gadget satisfying the specifications of Section 3.2.2.1. Recall that, as discussed in Section 3.2.2.2, we may assume that $1 < x, y < n$ holds for every $(x,y) \in S_{i,j}$.

3.2.2.5.1 Different Types of Edges in Main Gadget

Before proving Lemma 3.19, we first describe the construction of the main gadget in more detail (see Figure 3.4). The main gadget has n^2 rows denoted by $R_1, R_2, \ldots, R_{n^2}$ and $2n+1$ columns denoted by $C_0, C_1, \ldots, C_{2n+1}$. Let us denote the vertex at intersection of row R_i and column C_j by v_i^j. We now describe the various different kinds of edges in the main gadget gadget.

1. **Left Source Edges**: For every $i \in [n]$, the edge (ℓ_i, ℓ'_i) is a left source edge.

2. **Right Sink Edges**: For every $i \in [n]$, the edge (r'_i, r_i) is a right sink edge.

3. **Top Source Edges**: For every $i \in [n]$, the edge (t_i, v_1^i) is a top source edge.

4. **Bottom Sink Edges**: For every $i \in [n]$, the edge $(v_{n^2}^{n+i}, b_i)$ is a bottom sink edge.

5. **Source Internal Edges**: It is the set of n^2 edges of the form (ℓ'_i, v_j^0) for $i \in [n]$ and $n(i-1)+1 \leq j \leq n \cdot i$. Number the source internal edges from top to bottom, i.e., the edge (ℓ'_i, v_j^0) is called the j^{th} source internal edge, where $i \in [n]$ and $n(i-1)+1 \leq j \leq n \cdot i$.

6. **Sink Internal Edges**: It is the set of n^2 edges of the form (v_j^{2n+1}, r'_i) for $i \in [n]$ and $n(i-1)+1 \leq j \leq n \cdot i$. Number the sink internal edges from top to bottom, i.e., the edge (v_j^{2n+1}, r'_i) is called the j^{th} sink internal edge, where $i \in [n]$ and $n(i-1)+1 \leq j \leq n \cdot i$.

7. **Bridge Edges**: It is the set of n^2 edges of the form (v_i^n, v_i^{n+1}) for $1 \leq i \leq n^2$. We number the bridge edges from top to bottom, i.e., the edge (v_i^n, v_i^{n+1}) is called the i^{th} bridge edge.

8. **Inrow Right Edges**: For each $i \in [n^2]$ we call the $\rightarrow$ edges (except the bridge edge (v_i^n, v_i^{n+1})) connecting vertices of row R_i as inrow right edges. An example of an inrow right edge is (v_1^0, v_1^1).

9. **Interrow Down Edges**: For each $i \in [n^2 - 1]$ we call the $2n \downarrow$ edges connecting vertices of row R_i to R_{i+1} as interrow down edges. The $2n$ edges interrow edges between row R_i and R_{i+1} are (v_i^j, v_{i+1}^j) for each $1 \leq j \leq 2n$.

10. **Shortcut Edges**: There are $2|S|$ shortcut edges, namely $e_1, e_2, \ldots, e_{|S|}$ and $f_1, f_2, \ldots, f_{|S|}$. The shortcut edge for a $(x,y) \in S$ for some $1 < x, y < n$ is defined the following way:

    - Introduce a new vertex g_x^y at the middle of the edge $(v_{n(x-1)+y-1}^y, v_{n(x-1)+y}^y)$ to create two new edges $(v_{n(x-1)+y-1}^y, g_x^y)$ and $(g_x^y, v_{n(x-1)+y}^y)$. Then the edge $e_{x,y}$ is $(v_{n(x-1)+y}^{y-1}, g_x^y)$.
    - Introduce a new vertex h_x^y at the middle of the edge $(v_{n(x-1)+y}^{n+y}, v_{n(x-1)+y+1}^{n+y})$ to create two new edges $(v_{n(x-1)+y}^{n+y}, h_x^y)$ and $(h_x^y, v_{n(x-1)+y+1)}^{n+y})$. Then the edge $f_{x,y}$ is $(h_x^y, v_{n(x-1)+y}^{n+y+1})$.

3.2.2.5.2 Assigning Weights in the Main Gadget

Define $B = 11n^2$. We assign weights to the edges as follows:

1. Each left source edge has weight B^4.
2. Each right sink edge has weight B^4.
3. For every $1 \leq i \leq n$, the ith top source edge (t_i, v_1^i) has weight B^4.
4. For every $1 \leq i \leq n$, the ith bottom sink edge $(v_{n^2}^{n+i}, b_i)$ has weight B^4.
5. For each $i \in [n^2]$, the ith bridge edge (v_i^n, v_i^{n+1}) has weight B^3.
6. For each $i \in [n^2]$, the ith source internal edge has weight $B^2(n^2 - i)$.
7. For each $j \in [n^2]$, the jth sink internal edge has weight $B^2 \cdot j$.

8. Each inrow right edge has weight $3B$.

9. For each $(x,y) \in S$, both the shortcut edges $e_{x,y}$ and $f_{x,y}$ have weights B each.

10. Each interrow down edge that does not have a shortcut incident to it has weight 2. If an interrow edge is split into two edges by the shortcut incident to it, then we assign a weight 1 to each of the two parts.

Now we define the quantity M_n^* stated in statement of Lemma 3.19:

$$M_n^* = 4B^4 + B^3 + B^2n^2 + B(6n-4) + 2(n^2-1). \tag{3.5}$$

Next we are ready to prove the statements of Lemma 3.19.

3.2.2.5.3 $(x,y) \in S \Rightarrow \exists$ a solution $E_{x,y}$ of weight M_n^* that represents (x,y)

For $(x,y) \in S \subseteq [n] \times [n]$ define $z = n(x-1)+y$. Let $E_{x,y}$ be the union of the following sets of edges:

- The x^{th} left source edge and x^{th} right sink edge. This incurs a weight of $2B^4$.
- The y^{th} top source edge and the y^{th} bottom sink edge. This incurs a weight of $2B^4$.
- The z^{th} bridge edge. This incurs a weight of B^3.
- The z^{th} source internal edge and z^{th} sink internal edge. This incurs a weight of B^2n^2.
- All inrow right edges from row R_z except (v_z^{y-1}, v_z^y) and (v_z^{n+y}, v_z^{n+y+1}). This incurs a weight of $3B \cdot (2n-2)$.
- The shortcut edges $e_{x,y}$ and $f_{x,y}$. This incurs a weight of $2B$.
- The vertically downward path $v_1^y \to v_2^y \to \ldots \to v_z^y$. This incurs a weight of $2(z-1)$.

- The vertically downward path $v_z^{n+y} \to v_{z+1}^{n+y} \to \ldots \to v_{n^2}^{n+y}$. This incurs a weight of $2(n^2 - z)$.

From the above paragraph and Equation 3.5, it follows the total weight of $E_{x,y}$ is exactly M_n^*. Note that we can still travel from $(v_z^{y-1}$ to $v_z^y)$ as follows: take the path $(v_z^{y-1} \to g_x^y \to v_z^y)$. Similarly, even though the edge (v_z^{n+y}, v_z^{n+y+1}) is not present in $E_{x,y}$, we can still travel from $(v_z^{n+y}$ to $v_z^{n+y+1})$ in $E_{x,y}$ via the path $(v_z^{n+y} \to h_{x,y} \to v_z^{n+y+1})$.

It remains to show that $E_{x,y}$ satisfies the connectedness property and it represents $(x,y) \in S$. $E_{x,y}$ represents (x,y) because

- In $E_{x,y}$ the only outgoing edge from L is the one incident to ℓ_x
- In $E_{x,y}$ the only incoming edge to R is the one incident to r_x
- In $E_{x,y}$ the only outgoing edge from T is the one incident to t_y
- In $E_{x,y}$ the only incoming edge to B is the one incident to b_y

We now show that the connectedness property follows:

1. There is a $\ell_x \rightsquigarrow r_x$ path in $E_{x,y}$ obtained by taking the edges in the following order:
    - the left source edge (ℓ_x, ℓ'_x),
    - the source internal edge (ℓ'_x, v_z^0),
    - the horizontal path $v_z^0 \to v_z^1 \to \ldots v_z^n$ given by inrow right edges in row R_z,
    - the bridge edge (v_z^n, v_z^{n+1}),
    - the horizontal path $v_z^{n+1} \to v_z^{n+2} \to \ldots v_z^{2n+1}$ given by inrow right edges in row R_z,

- the sink internal edge (v_z^{2n+1}, r'_x), and
- the right sink edge (r'_x, r_x).

2. There is a $t_y \rightsquigarrow b_y$ path in $E_{x,y}$ obtained by taking the edges in the following order:

   - the top source edge (t_y, v_1^y),
   - the downward path $v_1^y \to v_2^y \to \dots v_z^y$ given by interrow down edges in column C_y,
   - the horizontal path $v_z^y \to v_z^{y+1} \to \dots v_z^n$ given by inrow right edges in row R_z,
   - the bridge edge (v_z^n, v_z^{n+1}),
   - the horizontal path $v_z^{n+1} \to v_z^{n+2} \to \dots v_z^{n+y}$ given by inrow right edges in row R_z,
   - the downward path $v_z^{n+y} \to v_{z+1}^{n+y} \to \dots v_{n^2}^{n+y}$ given by interrow down edges in column C_{n+y}, and
   - the bottom sink edge $(v_{n^2}^{n+y}, b_y)$.

Therefore, $E_{x,y}$ indeed satisfies the connectedness property.

3.2.2.5.4 E' satisfies connectedness property and has weight $\leq M_n^* \Rightarrow E'$ represents some $(\alpha, \beta) \in S$ and has weight exactly M_n^*

In this section we show that if a set of edges E' satisfies the connectedness property and has weight M_n^*, then it represents some $(\alpha, \beta) \in S$. We do this via the following series of claims and observations.

Claim 3.12. *E' contains*

- *exactly one left source edge,*
- *exactly one right sink edge,*
- *exactly one top source edge, and*
- *exactly one bottom sink edge.*

Proof. Since E' satisfies the connectedness property, it must contain at least one edge of each of the above types. Without loss of generality, suppose we have at least two left source edges in E'. Then the weight of the edge set E' is greater than or equal to the sum of weights of these two left source edges and the weight of a right sink edge, the weight of a top source edge, and the weight of a bottom sink edge. Thus if E' contains at least two left source edges, then its weight is at least $5B^4$. However, from Equation 3.5, we get that

$$\begin{aligned} M_n^* &= 4B^4 + B^3 + B^2 n^2 + B(6n-4) + 2(n^2-1) \\ &\leq 4B^4 + n \cdot B^3 + n \cdot B^3 + 6n \cdot B^3 + 2n \cdot B^3 \\ &= 4B^4 + 10n \cdot B^3 \\ &< 5B^4, \end{aligned}$$

since $B = 11n^2 > 10n$.

□

Therefore, we can set up the following notation:

- Let $i_L \in [n]$ be the unique index such that the left source edge in E' is incident to ℓ_{i_L}.

- Let $i_R \in [n]$ be the unique index such that the right sink edge in E' is incident to r_{i_R}.
- Let $i_T \in [n]$ be the unique index such that the top source edge in E' is incident to t_{i_T}.
- Let $i_B \in [n]$ be the unique index such that the bottom sink edge in E' is incident to b_{i_B}.

Claim 3.13. *The edge set E' contains exactly one bridge edge.*

Proof. To satisfy the connectedness property, we need at least one bridge edge, since the bridge edges form a cut between the top-distinguished vertices and the right-distinguished vertices as well as between the top-distinguished vertices and the bottom-distinguished vertices. Suppose that the edge set E' contains least two bridge edges. This contributes a weight of $2B^3$. We already have a contribution on $4B^4$ to weight of E' from Claim 3.12. Therefore, the weight of E' is at least $4B^4 + 2B^3$. However, from Equation 3.5, we get that

$$\begin{aligned}
M_n^* &= 4B^4 + B^3 + B^2n^2 + B(6n-4) + 2(n^2-1) \\
&\leq 4B^4 + B^3 + B^2n^2 + 6n \cdot B + 2n^2 \\
&\leq 4B^4 + B^3 + B^2n^2 + 6n^2B^2 + 2n^2B^2 \\
&= 4B^4 + B^3 + 9B^2n^2 \\
&< 4B^8 + 2B^3,
\end{aligned}$$

since $B = 11n^2 > 9n^2$. □

Let the index of the unique bridge edge in E' (guaranteed by Claim 3.13) be $\gamma \in [n^2]$. The connectedness property implies that we need to select at least one source internal edge incident to ℓ'_{i_L} and at least one sink internal edge incident to r'_{i_R}. Let the index of the source internal edge incident to ℓ'_{i_L} be j_L and the index of the sink internal edge incident to r'_{i_R} be j_R.

Claim 3.14. *$i_L = i_R$ and $j_L = j_R = \gamma$.*

Proof. By the connectedness property, there is a path from ℓ_{i_L} to some vertex in $r_{i_R} \cup b_{i_B}$. This path starts with $\ell_{i_L} \to \ell'_{i_L} \to v^1_{j_L}$ and has to use the γ^{th} bridge edge. By the construction of the main gadget (all edges are either downwards or towards the right), this path can never reach any row R_ℓ for $\ell < j_L$. Therefore, $\gamma \geq j_L$. By similar logic, we get $j_R \geq \gamma$. Therefore $j_R \geq j_L$.

If $j_R > j_L$, then the weight of the source internal edge and the sink internal edge is $B^5(n^2 - j_L + j_R) \geq B^5(n^2+1)$. We already have a contribution of $4B^4 + B^3$ to the weight of E' from Claim 3.12 and Claim 3.13. Therefore, the weight of E' is at least $4B^4 + B^3 + B^2(n^2+1)$. However, from Equation 3.5, we get that

$$\begin{aligned}
M_n^* &= 4B^4 + B^3 + B^2n^2 + B(6n-4) + 2(n^2-1) \\
&\leq 4B^4 + B^3 + B^2n^2 + 6n \cdot B + 2n^2 \\
&\leq 4B^4 + B^3 + B^2n^2 + 6n^2 \cdot B + 2n^2 \cdot B \\
&= 4B^4 + B^3 + B^2n^2 + 8n^2 \cdot B \\
&< 4B^4 + B^3 + B^2(n^2+1),
\end{aligned}$$

since $B = 11n^2 > 8n^2$. Hence $j_R = j_R = \gamma$. Observing $i_L = \lfloor \frac{j_L}{n} \rfloor$ and $i_R = \lfloor \frac{j_R}{n} \rfloor$, we obtain $i_L = i_R$. $\square$

Let $i_L = i_R = \alpha$ and $\gamma = n(\alpha - 1) + \beta$. We will show that below E' represents the pair (α, β). For this, we need to show three conditions:

1. The only left source edge in E' is the one incident to ℓ_α and the only right sink edge in E' is the one incident to r_α.
2. The only top source edge in E' is the one incident to t_β and the only bottom sink edge in E' is the one incident to b_β.
3. The pair (α, β) is in S.

The first statement above follows from Claim 3.12 and Claim 3.14. To prove the second claim it is sufficient to show that $i_T = i_B = \beta$, since Claim 3.12 implies E' contains exactly one top source edge and exactly one bottom sink edge. We now continue with the proof of the last two statements mentioned above:

Claim 3.15. *E' contains exactly $2n - 2$ inrow right edges, all of them from row R_γ. As a corollary, we get that there are two shortcuts incident to row R_γ, i.e., $(\alpha, \beta) \in S$ and also that E' uses both these shortcuts.*

Proof. Claim 3.14 implies $j_L = j_R = \gamma$. Hence the $\ell_\alpha \rightsquigarrow r_\alpha \cup b_{i_B}$ path in E' contains a $v_\gamma^0 \rightsquigarrow v_\gamma^n$ subpath P_1. By the construction of the main gadget, we cannot reach an upper row from a lower row. Hence this subpath P_1 must be the path $v_\gamma^0 \to v_\gamma^2 \to \ldots \to v_\gamma^n$. This path P_1 can at most use the unique shortcut edge incident to row R_γ and column C_β to replace an inrow right edge. Hence P_1 uses at least $n - 1$ inrow right edges, with equality only if R_γ has a shortcut incident to it.

Similarly, the $\ell_\alpha \cup t_{i_T} \rightsquigarrow r_\alpha$ path in E' contains a $v_\gamma^{n+1} \rightsquigarrow v_\gamma^{2n+1}$ subpath P_2. By the construction of the main gadget, we cannot reach an upper row from a lower row. Hence this subpath P_2 must be the path $v_\gamma^{n+1} \to v_\gamma^{n+2} \to \ldots \to v_\gamma^{2n+1}$. This path P_2 can at most use the unique shortcut edge incident to row R_γ and column $C_{n+\beta}$ to replace an inrow right edge. Hence P_2 uses at least $n-1$ inrow right edges, with equality only if R_γ has a shortcut incident to it.

Clearly, the sets of inrow edges used by P_1 and P_2 are disjoint, and hence E' contains at least $2n-2$ inrow right edges from row R_γ. Suppose E' contains at least $2n-1$ inrow right edges. Then it incurs a weight of $3B \cdot (2n-1)$. From Claim 3.12, Claim 3.13 and Claim 3.14 we already have a contribution of $4B^4 + B^3 + B^2n^2$. Therefore the weight of E' is at least $4B^4 + B^3 + B^2n^2 + 3B \cdot (2n-1)$.

However, from Equation 3.5, we get that

$$\begin{aligned} M_n^* &= 4B^4 + B^3 + B^2n^2 + B(6n-4) + 2(n^2-1) \\ &\leq 4B^4 + B^3 + B^2n^2 + B(6n-4) + 2n^2 \\ &< 4B^4 + B^3 + B^2n^2 + 3B \cdot (2n-1), \end{aligned}$$

since $B = 11n^2 > 2n^2$. Therefore, E' can only contain at most $2n-2$ inrow right edges. Hence there must be two shortcut edges incident to row R_γ, which are both used by E'. Since $\gamma = n(\alpha-1)+\beta$, the fact that row R_γ has shortcut edges incident to it implies $(\alpha, \beta) \in S$. $\square$

Note that the remaining budget left for the weight of E' is at most $2(n^2-1)$.

Claim 3.16. $i_T = i_B = \beta$

Proof. Recall that the only bridge edge used is the one on row R_γ. So the $t_{i_T} \rightsquigarrow r_\alpha \cup b_{i_B}$ path in E' contains a $v_1^{i_T} \rightsquigarrow v_\gamma^n$ subpath P_3 (of interrow edges). By Claim 3.15, all inrow right edges are only from row R_γ. As the only remaining budget is $2(n^2-1)$, we cannot use any other shortcuts or inrow right edges since $B = 11n^2 > 2(n^2-1)$. Therefore, P_3 contains another $v_1^{i_T} \rightsquigarrow v_\gamma^{i_T}$ subpath P_3'. If $i_T \neq \beta$, then P_3' incurs weight $2(\gamma-1)$. Note that we also pay a weight of 1 to use half of the interrow edge when we use the shortcut edge which is incident to row R_γ and column C_β.

Similarly, the $\ell_\alpha \cup t_{i_T} \rightsquigarrow b_{i_B}$ path in E' contains a $v_\gamma^{n+1} \rightsquigarrow v_{n^2}^{n+i_B}$ subpath P_4 (of interrow edges). By Claim 3.15, all inrow horizontal edges are only from row R_γ. As the only remaining budget is $2(n^2-1)$, we cannot use any other shortcuts or inrow right edges. Therefore, P_4 contains another $v_\gamma^{n+i_B} \rightsquigarrow v_{n^2}^{n+i_B}$ subpath say P_4'. If $i_B \neq \beta$, then P_4' incurs weight $2(n^2-\gamma)$. Note that we also pay a weight of 1 to use half of the interrow edge when we use the shortcut edge which is incident to row R_γ and column C_β.

Suppose without loss of generality that $i_T \neq \beta$. Then P_3' incurs a weight of $2(\gamma-1)$, and the half interrow edge used incurs an additional weight of 1. Also P_4' incurs a weight of $2(n^2-\gamma)$. Hence the total weight incurred is $2(\gamma-1)+1+2(n^2-\gamma) = 2(n^2-1)+1$ which is greater than our allowed budget. Hence $i_T = \beta$. It can be shown similarly that $i_B = \beta$. □

Claim 3.12, Claim 3.14, Claim 3.15 and Claim 3.16 together imply that E' represents $(\alpha,\beta) \in S$. We now show that weight of E' is exactly M_n^*.

Lemma 3.22. *Weight of E' is exactly M_n^**

Proof. Claim 3.12 contributes a weight of $4B^8$ to E'. Claim 3.13 contributes a weight of B^7 to E'. From the proof of Claim 3.14, we can see that E' incurs weight B^5n^2 from the source internal edge and sink internal edge. Claim 3.15 implies that E' contains exactly $2n-2$ inrow right edges from row $R_{\gamma'}$ and also both shortcuts incident to row R_{γ}. For each of the shortcuts we need to use an additional half interrow edge of weight 1. Hence this incurs a weight of $3B^3 \cdot (2n-2) + 2B^3 + 2$. By reasoning similar to Claim 3.16, E' contains at least $n^2 - 1$ interrow edges. Out of these at most 2 half interrow edges could have been counted in the above step. Hence this incurs a weight of $2(n^2-1)-2$.

Therefore, we have weight of $E' \geq 4B^8 + B^7 + B^5n^2 + \Big(3B^3 \cdot (2n-2) + 2B^3 + 2\Big) + \Big(2(n^2-1)-2\Big) = M_n^*$. Hence the weight of E' is exactly M_n^*. □

This completes the proof of the second statement of Lemma 3.19.

3.2.3 (Almost) Tight Lower Bound for SCSS on General Graphs

Given our speedup for SCSS in planar graphs (see Section 3.1.2), one may ask if it is possible to get any similar speedup in general graphs. Our next result shows that the $n^{O(k)}$ algorithm of Feldman-Ruhl is *almost optimal* in general graphs:

Theorem 3.14. *The* STRONGLY CONNECTED STEINER SUBGRAPH *problem cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$ where f is an arbitrary function, k is the number of terminals and n is the number of vertices in the instance, unless ETH fails.*

Our proof is similar to the W[1]-hardness proof of Guo et al. [GNS11]. They showed the W[1]-hardness of SCSS on general graphs parameterized by the number k

of terminals by giving a reduction from k-CLIQUE. However, this reduction uses "edge selection gadgets" and since a k-clique has $\Theta(k^2)$ edges, the parameter is increased at least to $\Theta(k^2)$. Combining with the result of Chen et al. [CHKX06] regarding the non-existence of an $f(k) \cdot n^{o(k)}$ algorithm for k-CLIQUE under ETH, this gives a lower bound of $f(k) \cdot n^{o(\sqrt{k})}$ for SCSS on general graphs. To avoid the quadratic blowup in the parameter and thereby get a stronger lower bound, we use the COLORED SUBGRAPH ISOMORPHISM (CSI) problem as the source problem of our reduction. For this problem, Marx [Mar10] gave a $f(k) \cdot n^{o(k/\log k)}$ lower bound under ETH, where $k = |E(G)|$ is the number of edges of the subgraph G to be found in graph H. The reduction of Guo et al. [GNS11] from CLIQUE can be turned into a reduction from CSI which uses only $|E(G)|$ edge selection gadgets, and hence the parameter is $\Theta(|E(G)|)$. Then the lower bound of $f(k) \cdot n^{o(k/\log k)}$ transfers from CSI to SCSS.

We now define the COLORED SUBGRAPH ISOMORPHISM problem formally:

COLORED SUBGRAPH ISOMORPHISM (CSI
Input : Undirected graphs $G = (V_G = \{g_1, g_2, \ldots, g_\ell\}, E_G)$ and $H = (V_H, E_H)$, and a partition of V_H into disjoint subsets $H_1, H_2, \ldots, H_\ell$
Question: Is there an injection $\phi : V_G \to V_H$ such that

1. For every $i \in [\ell]$ we have $\phi(g_i) \in H_i$.
2. For every edge $\{g_i, g_j\} \in E_G$ we have $\{\phi(g_i), \phi(g_j)\} \in E_H$.

Marx [Mar10] showed the following hardness result:

Theorem 3.15. COLORED SUBGRAPH ISOMORPHISM *(CSI) cannot be solved in time $f(r) \cdot n^{o(r/\log r)}$ where f is an arbitrary function, r is the number of edges in G and n is the number of vertices in H, unless ETH fails.*

By giving a reduction from CSI to SCSS where $k = O(|E_G|)$ we will get a $n^{o(k \log k)}$ hardness for SCSS under the ETH, where k is the number of terminals. Consider an instance (G,H) of CSI. We now construct an instance of SCSS as follows:

- $B = \{b_i \mid i \in [\ell]\}$
- $C = \{c_v \mid v \in V_H\}$
- $C' = \{c'_v \mid v \in V_H\}$
- $D = \{d_{uv'} \cup d_{vu'} \mid \{u,v\} \in E_H\}$
- $A = \{a_{uv'} \cup a_{vu'} \mid \{u,v\} \in E_H\}$
- $F = \{f_{ij} \mid 1 \leq i,j \leq \ell \mid g_i g_j \in E_G\}$
- $V^* = B \cup C \cup C' \cup D \cup A \cup F$
- $E_1 = \{(c_v, b_i) \mid v \in H_i, 1 \leq i \leq \ell\}$
- $E_2 = \{(b_i, c'_v) \mid v \in H_i, 1 \leq i \leq \ell\}$
- $E_3 = \{(c'_v, c_v) \mid v \in V_H\}$
- $E_4 = \{(c_v, d_{vu'}) \mid \{u,v\} \in E_H\}$
- $E_5 = \{(a_{vu'}, c'_u) \mid \{u,v\} \in E_H\}$
- $E_6 = \{(d_{vu'}, a_{vu'}) \mid \{u,v\} \in E_H\}$
- $E_7 = \{(f_{ij}, d_{vu'}) \cup (a_{vu'}, f_{ij}) \mid \{u,v\} \in E_H; v \in H_i; u \in H_j; 1 \leq i,j \leq \ell\}$
- $E^* = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6 \cup E_7$

An illustration of the construction for a small graph in given in Figure 3.5. We imagine that the edges of E_4 are colored red to help us argue the proof. Let the terminals be $T = B \cup F$. In the instance of COLORED SUBGRAPH ISOMORPHISM we can assume the graph G is connected, otherwise we can solve the problem for each connected component. Therefore $k = |T| = \ell + 2|E_G| = O(|E_G|)$.

Theorem 3.16. *The instance (G,H) of CSI answers YES if and only if there is a solution for the SCSS instance (V^*, E^*, T) of size $3\ell + 10|E_G|$.*

Proof.

Suppose the instance (G,H) of CSI answers YES and let ϕ be the injection from $V_G \to V_H$. Then we claim the following set M' of $3\ell + 10|E_G|$ edges forms a solution for the SCSS instance:

- $M_1 = \{(c'_{\phi(g_i)}, c_{\phi(g_i)}) \mid i \in [\ell]\}$
- $M_2 = \{(b_i, c'_{\phi(g_i)}) \mid i \in [\ell]\}$
- $M_3 = \{(c_{\phi(g_i)}, b_i) \mid i \in [\ell]\}$
- $M_4 = \{(c_{\phi(g_i)}, d_{\phi(g_i)\phi'(g_j)}) \cup (d_{\phi(g_i)\phi'(g_j)}, a_{\phi(g_i)\phi'(g_j)}) \cup (a_{\phi(g_i)\phi'(g_j)}, c'_{\phi(g_j)}) \mid g_i g_j \in E_G; 1 \leq i, j \leq \ell\}$.
- $M_5 = \{(f_{ij}, d_{\phi(g_i)\phi'(g_j)}) \cup (a_{\phi(g_i)\phi'(g_j)}, f_{ij}) \mid g_i g_j \in E_G; 1 \leq i, j \leq \ell\}$.
- $M' = M_1 \cup M_2 \cup M_3 \cup M_4 \cup M_5$

First consider $i \neq j$ such that $g_i g_j \in E_G$. Then there is a $b_i \rightsquigarrow b_j$ path in M', namely $b_i \to c'_{\phi(g_i)} \to c_{\phi(g_i)} \to d_{\phi(g_i)\phi'(g_j)} \to a_{\phi(g_i)\phi'(g_j)} \to c'_{\phi(g_j)} \to c_{\phi(g_j)} \to b_j$. Generalizing this and observing G is connected we can see any two terminals in B are strongly connected. Now consider two terminals f_{ij} and b_q such that $1 \leq i, j, q \leq \ell$. The existence of the terminal f_{ij} implies $g_i g_j \in E_G$ and hence $\phi(g_i)\phi(g_j) \in E_H$. There is a path in M' from f_{ij} to b_q: use the path $f_{ij} \rightsquigarrow d_{\phi(g_i)\phi'(g_j)} \to a_{\phi(g_i)\phi'(g_j)} \to c'_{\phi(g_j)} \to c_{\phi(g_j)} \to b_j$ followed by the $b_j \rightsquigarrow b_q$ path (which is shown to exist above). Similarly there is a path in M' from b_q to f_{ij}: use the $b_q \rightsquigarrow b_i$ path (which is shown to exist above) followed by the path $b_i \rightsquigarrow c'_{\phi(g_i)} \to c_{\phi(g_i)} \to d_{\phi(g_i)\phi'(g_j)} \to a_{\phi(g_i)\phi'(g_j)} \to f_{ij}$. Hence each terminal in B can reach every terminal in F and vice versa. Finally consider any two terminals in F, say f_{ij} and f_{st}: f_{ij} can first reach b_i and we have seen above that b_i can reach any terminal in F. This shows M' forms a solution for the SCSS instance.

Now we claim if there is solution of size $3\ell + 10|E_G|$ to the SCSS instance, then the instance (G,H) of CSI answers YES. Consider a terminal $f_{ij} \in F$. The only out-neighbors of f_{ij} are vertices from D and so we need to pick an edge say $(f_{ij}, d_{vu'})$ such that $v \in H_i$ and $u \in H_j$. Now the only neighbor of $d_{vu'}$ is $a_{vu'}$ and hence we need to pick this edge as well. Finally, we also need one incoming edge into f_{ij} since we desire strong connectivity. So for each f_{ij}, we need three "private" edges in the sense that every terminal in F needs three such edges in any optimum solution. This uses up $6|E_G|$ of the budget. Referring to Figure 3.5, we can see any $f_{ij} \in F$ needs two "private" red edges: one edge coming out of some vertex in A and some edge going into a vertex of D. This uses up $4|E_G|$ more from the budget leaving us with only 3ℓ edges.

Consider b_i for $i \in [\ell]$. First we claim at least three edges must be selected for b_i to have incoming and outgoing paths to the other terminals. The only outgoing edge from b_i is to vertices of C', and hence we need to pick an edge say (b_i, c'_v) such that $v \in H_i$.

Since the only out-neighbor of c'_v is c_v, we need to pick this edge as well. We need at least one incoming edge into b_i to account for incoming paths from other terminals to b_i. So each b_i needs to have at least three edges selected in order to have incoming and outgoing paths to other terminals. Moreover, all these edges are clearly ”private”, i.e., different for each b_i. But as seen in last paragraph, our remaining budget is exactly 3ℓ, and hence we must selected exactly three such edges for each b_i. In other words, once we select the edges (b_i, c'_v) and c'_v, c_v such that $v \in H_i$ then we must select the edge (c_v, b_i). Else to have paths incoming to b_i, if we select the edge (c_w, b_i) for some $w \in H_i, w \neq v$ then since c'_w is the only neighbor of c_w we would need to select this edge also causing four edges to be selected for b_i, a contradiction. So for every $i \in [\ell]$, there is a vertex $v_i \in H_i$ such that the edges $(b_i, c'_{v_i}), (c'_{v_i}, c_{v_i})$ and (c_{v_i}, b_i) are selected in the solution for the SCSSS instance. Further these are the only chosen bold black edges corresponding to b_i (refer to Figure 3.5). It also follows for each $f_{ij} \in F$ we can select at most three of the dotted black edges.

Define $\phi : V_G \to V_H$ by $\phi(g_i) = v_i$ for each $i \in [\ell]$. Since $v_i \in H_i$ and since the $H'_i s$ form a disjoint partition of V_H the function ϕ is an injection. Consider any edge $g_i g_j \in E_G$. We have seen above that we only pick three black dotted edges per f_{ij}. Suppose for f_{ij} we have picked the edges $(f_{ij}, d_{vu'}), (d_{vu'}, a_{vu'})$ and $(a_{vu'}, f_{ij})$ for some $v \in H_i, u \in H_j$. The only incoming path for f_{ij} is via $d_{vu'}$. Also the only outgoing path from b_i is via c_{v_i}. If $v_i \neq v$ then we will need two other black dotted edges to reach f_{ij}, which is a contradiction since have already picked the allocated budget of three such edges. Similarly $v_j = u$ and the existence of $d_{vu'}$ vertex implies $vu' \in E_H$, i.e., $\phi(g_i)\phi(g_j) \in E_H$.

□

Finally we are now ready to prove Theorem 3.14, which we restate below:

Theorem 3.14 . *The* STRONGLY CONNECTED STEINER SUBGRAPH *(SCSS) problem cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$ where f is an arbitrary function, k is the number of terminals and n is the number of vertices in the instance, unless ETH fails.*

Proof. Observe that the number of terminals k of the SCSS instance is $|B \cup F| = |V(G)| + 2|E_G| = O(|E_G|)$ since we had the assumption that G is connected. The number of vertices in the SCSS instance is $|V^*| = |V_G| + 2|V_H| + 4|E_H| + 2|E_G| = O(|E_H|)$. Therefore from Theorem 3.15 we can conclude under the ETH there is no $f(k) \cdot n^{o(k/\log k)}$ algorithm for SCSS where n is the number of vertices in the graph and k is the number of terminals. □

3.2.4 Tight Lower Bound for DSF on Planar DAGs

Even though Feldman and Ruhl [FR06] were able to generalize their $n^{O(k)}$ time algorithm from SCSS to DSF, we show that, surprisingly, such a generalization is not possible for our $2^{O(k\log k)} \cdot n^{O(\sqrt{k})}$ time algorithm for planar SCSS from Section 3.1.2.

Theorem 3.17. *The* DIRECTED STEINER FOREST *problem on planar directed acyclic graphs (DAGs) is W[1]-hard parameterized by the number k of terminal pairs and there is no $f(k) \cdot n^{o(k)}$ algorithm for any function f, unless the ETH fails.*

This implies that the $n^{O(k)}$ algorithm of Feldman-Ruhl for DSF is optimal, even on planar directed acyclic graphs. As in our lower bound for planar SCSS (see Section 3.2.2), the proof is by reduction from an instance of $k \times k$ GRID TILING problem. However, unlike in the reduction to SCSS where we needed $O(k^2)$ terminals, the reduction to DSF needs only $O(k)$ pairs of terminals (see Figure 3.6). Since the parameter blowup is linear, the $f(k) \cdot n^{o(k)}$ lower bound for GRID TILING from [Marc] transfers to DSF. We show hardness for the edge version with integer weights. By Section 3.2.1, our hardness results also carry over to unweighted vertex version of DSF.

Consider an instance of GRID TILING. We now build an instance of edge-weighted DSF as shown in Figure 3.6. We consider $2k$ pairs to be connected: (a_i, b_i) and (c_i, d_i) for each $i \in [k]$. We introduce k^2 red gadgets where each gadget is an $n \times n$ grid. Let the weight of each black edge be 2.

Definition 3.13. *An $a_i \rightsquigarrow b_i$* canonical *path is a path from a_i to b_i which starts with a blue edge coming out of a_i, then follows a horizontal path of black edges and finally ends with a blue edge going into b_i. Similarly an $c_j \rightsquigarrow d_j$* canonical *path is a path from c_j to d_j which starts with a blue edge coming out of c_j, then follows a vertically downward path of black edges and finally ends with a blue edge going into d_j.*

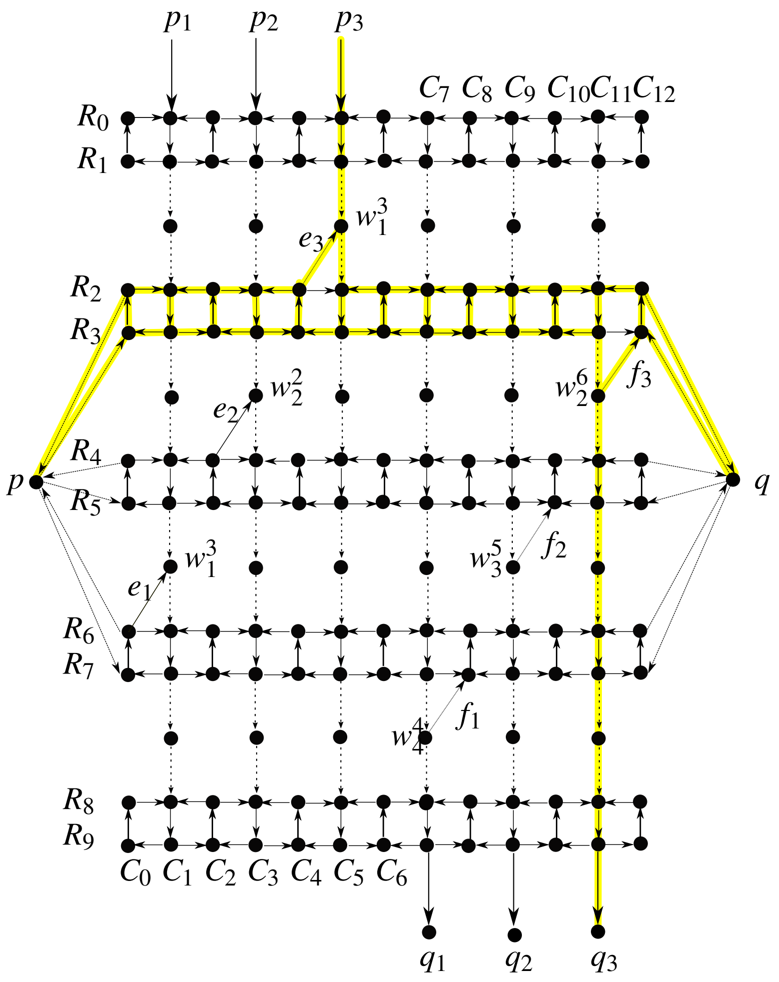


Figure 3.3: The connector gadget for $n = 3$. A set of edges representing 3 is highlighted in the figure.

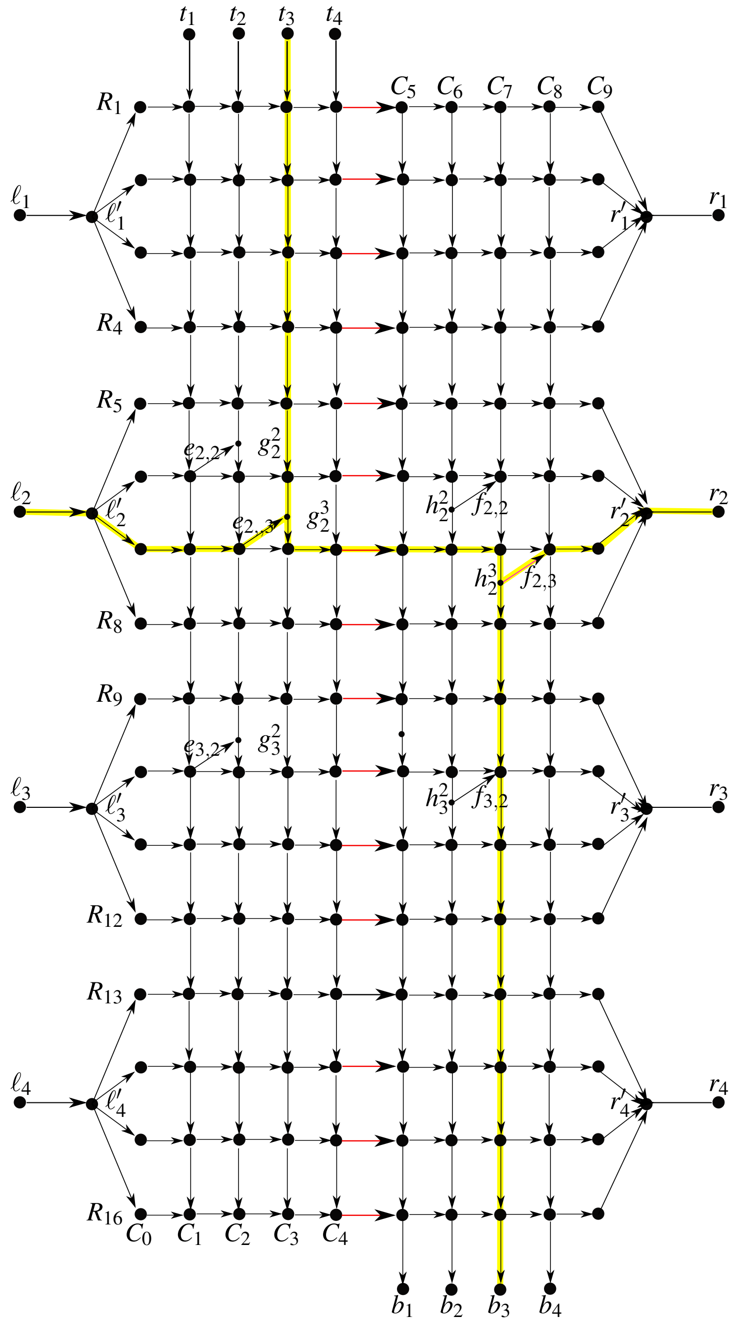


Figure 3.4: The main gadget representing the set $\{(2,2),(2,3),(3,2)\}$. The highlighted edges represent the pair $(2,3)$.

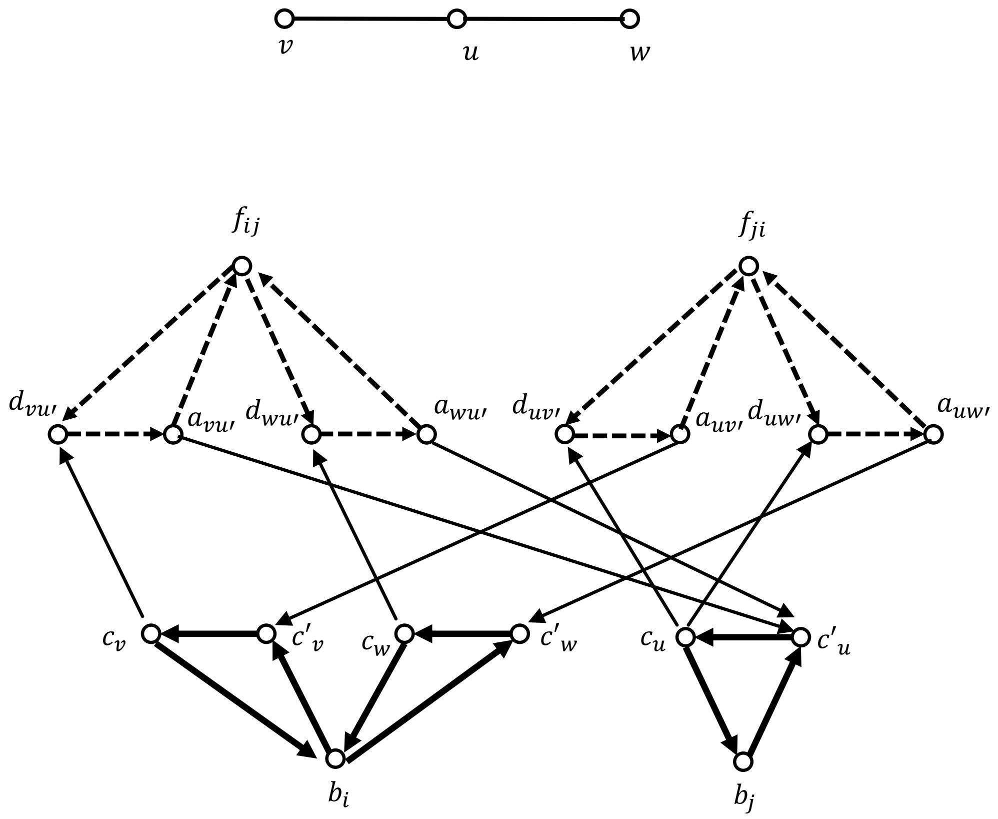


Figure 3.5: Part of the construction from Theorem 3.16 with three vertices: v, w of color i and u of color j and two edges $\{v, u\}$ and $\{u, w\}$.

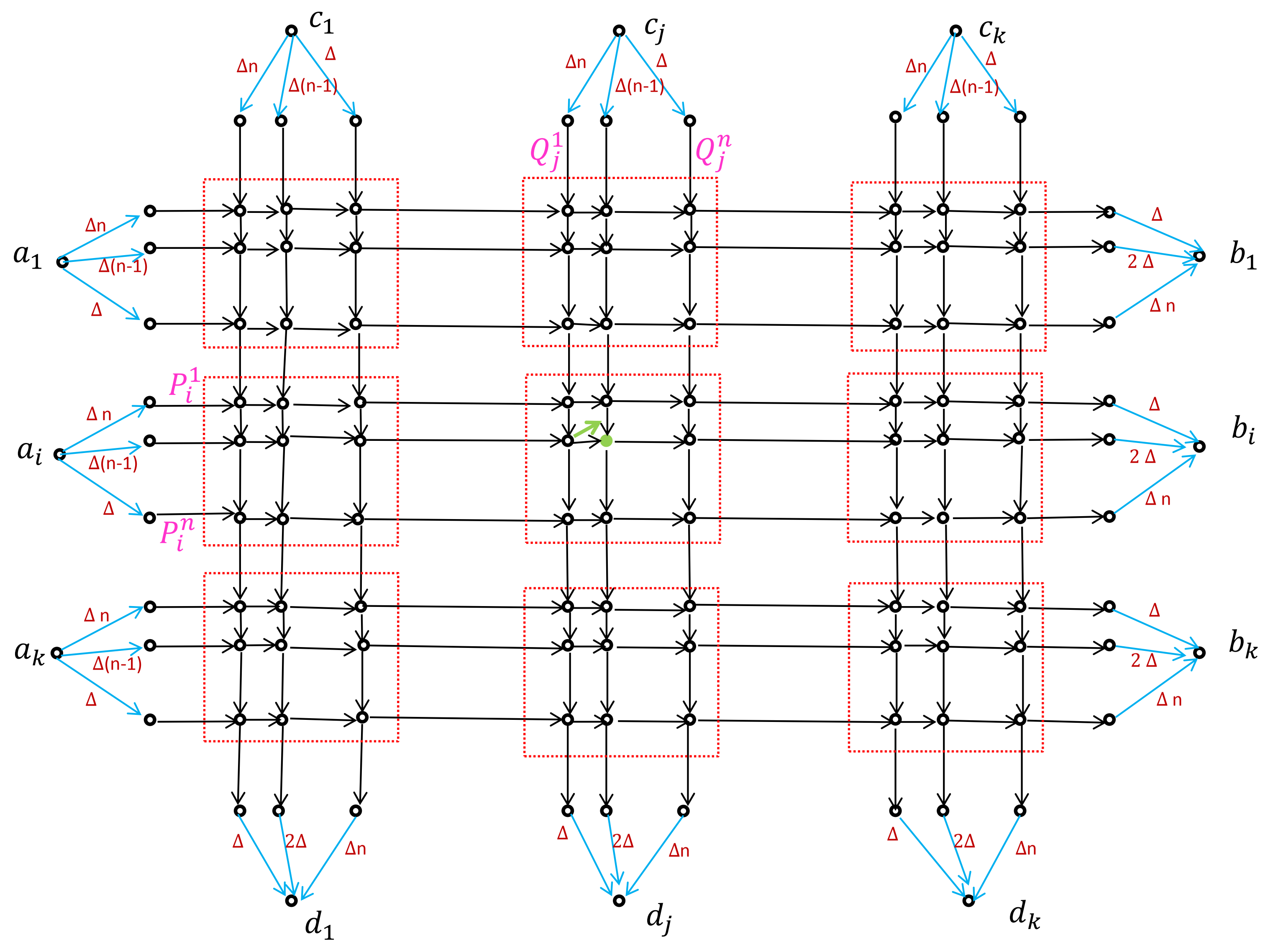


Figure 3.6: The instance of DSF created from an instance of Grid Tiling.

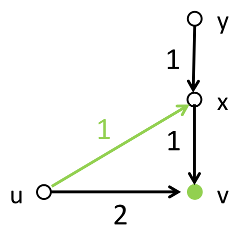


Figure 3.7: Let u, v be two consecutive vertices on the canonical path say P_i^{ℓ}. Let v be on the canonical path $Q_j^{\ell'}$ and let y be the vertex preceding it on this path. If v is a green vertex then we subdivide the edge (y, v) by introducing a new vertex x and adding two edges (y,x) and (x, v) of weight 1. We also add an edge (u,x) of weight 1. The idea is if both the edges (y, v) and (u, v) were being used initially then now we can save a weight of 1 by making the horizontal path choose (u,x) and then we get (x, v) for free, as it is already being used by the vertical canonical path.

There are n edge-disjoint $a_i \rightsquigarrow b_i$ canonical paths: let us call them $P_i^1, P_i^2, \ldots, P_i^n$ as viewed from top to bottom. They are named using magenta color in Figure 3.6. Similarly we call the canonical paths from c_j to d_j as $Q_j^1, Q_j^2, \ldots, Q_j^n$ when viewed from left to right. For each $i \in [k]$ and $\ell \in [n]$ we assign a weight of $\Delta(n+1-\ell), \Delta\ell$ to the first, last blue edges of P_i^{ℓ} respectively. Similarly for each $j \in [k]$ and $\ell \in [n]$ we assign a weight of $\Delta(n+1-\ell), \Delta\ell$ to the first, last blue edges of Q_j^{ℓ} respectively. Thus the total weight of first and last blue edges on any canonical path is exactly $\Delta(n+1)$. The idea is to choose Δ large enough such that in any optimum solution the paths between the terminals will be exactly the canonical paths. We will see $\Delta = 4n^2$ will suffice for our purposes. Any canonical path uses two blue edges (which sum up to $\Delta(n+1)$), $(k+1)$ black edges not inside the gadgets and $(n-1)$ black edges inside each gadget. Since the number of gadgets each canonical path visits is k and the weight of each black edge is 2, we have the total weight of any canonical path is $\Delta(n+1) + 2(k+1) + 2k(n-1)$.

Intuitively the k^2 gadgets correspond to the k^2 sets in the GRID TILING instance.

Let us denote the gadget which is the intersection of the $a_i \rightsquigarrow b_i$ path and $c_j \rightsquigarrow d_j$ path by $G^{i,j}$. If $(x,y) = s_{i,j} \in S_{i,j}$ then we color green the vertex in the gadget $G_{i,j}$ which is the unique intersection of the canonical paths P_i^x and Q_j^y. Then we add a shortcut as shown in Figure 3.7. The idea is if both the $a_i \rightsquigarrow b_i$ path and $c_j \rightsquigarrow d_j$ path pass through the green vertex then the $a_i \rightsquigarrow b_i$ path can save a weight of 1 by using the green edge and a vertical edge to reach the green vertex, instead of paying a weight of 2 to use the horizontal edge reaching the green vertex. It is easy to see there is an easy solution (without using green edges) for the DSF instance of weight $\beta = 2k\Big(\Delta(n+1) + 2(k+1) + 2k(n-1)\Big)$: each terminal pair just uses a canonical path and these canonical paths are pairwise edge-disjoint.

We need a small technical modification: we add one dummy row and column to the GRID TILING instance. Essentially we now have a dummy index 1. So neither the first row nor the first column of any $S_{i,j}$ has any elements in the GRID TILING instance. That is, no green vertex can be in the first row or first column of any gadget. Combining this fact with the orientation of the edges we get the only gadgets which can intersect any $a_i \rightsquigarrow b_i$ path are $G_{i,1}, G_{i,2}, \ldots, G_{i,k}$. Similarly the only gadgets which can intersect any $c_j \rightsquigarrow d_j$ path are $G_{1,j}, G_{2,j}, \ldots, G_{k,j}$.

We now prove two theorems which together give a reduction from GRID TILING to DSF.

Theorem 3.18. GRID TILING *has a solution implies OPT for DSF is at most* $\beta - k^2$.

Proof. For each $1 \leq i, j \leq k$ let $s_{i,j} \in S_{i,j}$ be the vertex in the solution of the GRID TILING instance. Therefore for every $i \in k$ we know that each of the k vertices $s_{i,1}, s_{i,2}, \ldots, s_{i,k}$ have the same x-coordinate, say α_i. Similarly for every $j \in [k]$ each of the k vertices $s_{1,j}, s_{2,j}, \ldots, s_{k,j}$ has the same x-coordinate, say γ_j. We now use the canonical path $P_i^{\alpha_i}$ for (a_i, b_i) and the canonical path $Q_j^{\gamma_j}$ for (c_j, d_j). Each of the $c_j \rightsquigarrow d_j$ paths will pay the full weight of a canonical path, which is $\Delta(n+1) + 2(k+1) + 2k(n-1)$. However each $a_i \to b_i$ path will encounter a green vertex in each of the k gadgets along its way, and hence will save one in each gadget (as shown in Figure 3.7) for a total saving of k. Hence over all the terminals we save a weight of k^2, and this is a solution for DSF instance of weight $\beta - k^2$. □

Now we show the other direction which is more involved. First we show some preliminary lemmas:

Lemma 3.23. *In any optimum solution for DSF there is a $c_j \rightsquigarrow d_j$ canonical path for some $j \in [k]$.*

Proof. Suppose to the contrary that for some $j \in [k]$ there is no canonical $c_j \rightsquigarrow d_j$ path in the optimum solution. From the orientation of the edges, we know that there is a $c_j \rightsquigarrow d_j$ path in the optimum solution that starts with the blue edge from Q_j^{ℓ} and ends with a blue edge from $Q_j^{\ell'}$ for some $\ell' > \ell$. We add to the solution those edges of Q_j^{ℓ} which were not present: in particular we at least add the last edge of Q_j^{ℓ}. Delete from the solution the last edge of $Q_j^{\ell'}$. Both these operations do not change the fact that the current set of edges forms a solution for DSF. This is because the last (blue) edge of any $c_j \rightsquigarrow d_j$ canonical path cannot be on any $a_i \rightsquigarrow b_i$ path. Changing the last edge saves us $(\ell' - \ell)\Delta \leq \Delta = 4n^2$.

However we have to be careful since we added some edges to the solution. But these edges are the internal (black) edges of Q_j^ℓ, and their weight is $2(k+1)+2k(n-1) < 4n^2$ since $1 \leq k \leq n$. Therefore we are able to create a new solution whose weight is less than that of an optimum solution, which is a contradiction. $\square$

Note the shortcut described in Figure 3.7 again brings the $a_i \rightsquigarrow b_i$ path back to the same horizontal canonical path.

Definition 3.14. *We call an $a_i \rightsquigarrow b_i$ path as an* almost canonical *path if it is basically a canonical path, but can additionally take the small detour given by the green edge in Figure 3.7. An almost canonical path must however end on the same horizontal level on which it began.*

Lemma 3.24. *In any optimum solution for DSF there is an $a_i \rightsquigarrow b_i$ almost canonical path for every $i \in [k]$.*

Proof. Suppose to the contrary that for some $i \in [k]$ there is no almost canonical $a_i \rightsquigarrow b_i$ path in the optimum solution. From the orientation of the edges, we know that there is a $a_i \rightsquigarrow b_i$ path in the optimum solution that starts with the blue edge from P_i^ℓ and ends with a blue edge from $P_i^{\ell'}$ for some $\ell' > \ell$ (note that the construction in Figure 3.7 does not allow any $a_i \rightsquigarrow b_i$ path to climb onto an upper canonical path). We add to the solution those edges of P_i^ℓ which were not present: in particular we at least add the last edge of P_i^ℓ. Delete from the solution the last edge of $P_i^{\ell'}$. Both these operations do not change the fact that the current set of edges forms a solution for DSF: this is because the last (blue) edge of any $a_i \rightsquigarrow b_i$ canonical path cannot be on any $c_j \rightsquigarrow d_j$ path. Changing the last edge saves us $(\ell' - \ell)\Delta \leq \Delta = 4n^2$. But we have to careful: we added some edges to the solution. But these edges are the internal (black) edges of P_i^{ℓ}, and their weight is $2(k+1) + 2k(n-1) < 4n^2$ since $1 \leq k \leq n$. So we are able to create a new solution whose weight is less than that of an optimum solution, which is a contradiction. □

Theorem 3.19. *OPT for DSF is at most $\beta - k^2$ implies the* Grid Tiling *instance has a solution.*

Proof. Consider any optimum solution say $\mathcal{X}$. By Lemma 3.23 and Lemma 3.24 we know that $\mathcal{X}$ has a $a_i \rightsquigarrow b_i$ almost canonical path and a $c_j \rightsquigarrow d_j$ canonical path for every $1 \leq i, j \leq k$. Moreover these set of $2k$ paths form a solution for DSF. Since any optimum solution is minimal $\mathcal{X}$ is the union of these $2k$ paths: one for each terminal pair. For the moment let us forget the modifications we did in Figure 3.7. So the $a_i \rightsquigarrow b_i$ path and $c_j \rightsquigarrow d_j$ path in $\mathcal{X}$ intersect in a unique point (in the gadget $G_{i,j}$). The weight of $\mathcal{X}$ is exactly β. However we know that there is a solution of weight at most $\beta - k^2$. It is easy to see any $a_i \rightsquigarrow b_i$ almost canonical path and a $c_j \rightsquigarrow d_j$ canonical path can have at most one edge in common: the edge which comes vertically downwards into the green vertex (see Figure 3.7). There are k^2 gadgets, and there is at most one edge per gadget which is double counted in $\mathcal{X}$. Hence for each gadget $G_{i,j}$ there is exactly one edge which is used by both the $a_i \rightsquigarrow b_i$ almost canonical path and the $c_j \rightsquigarrow d_j$ canonical path in $\mathcal{X}$. So the endpoint of each of these common edges must be green vertices, and at each such point we save a weight of one as described in Figure 3.7. Since each $a_i \rightsquigarrow b_i$ path is an almost canonical path and each $c_j \rightsquigarrow d_j$ path is a canonical path, the green vertices form a solution for the GRID TILING instance. □

We are now ready the prove Theorem 3.17 (restated below) which essentially says the $n^{O(k)}$ algorithm of Feldman-Ruhl [FR06] for DSF is optimal.

Theorem 3.17 . *The* DIRECTED STEINER FOREST *problem on planar directed acyclic graphs (DAGs) is W[1]-hard parameterized by the number k of terminal pairs and there is no $f(k) \cdot n^{o(k)}$ algorithm for any function f, unless the ETH fails.*

Proof. Theorem 3.18 and Theorem 3.19 together imply the W[1]-hardness. It is not hard to see the graph we constructed in Figure 3.6 is a planar DAG.

Chen et al. [CHKX06] showed for any function f an $f(k) \cdot n^{o(k)}$ algorithm for CLIQUE implies ETH fails. Theorem 3.17 gives a reduction which transforms the problem of $k \times k$ GRID TILING into an instance of DSF with $2k$ terminal pairs. Composing this reduction with the reduction of Marx [Marc] from $k \times k$ GRID TILING to k-CLIQUE, we obtain under ETH there is no $f(k) \cdot n^{o(k)}$ algorithm for DSF (even on planar DAGs) for any function f. □

CHAPTER 4

Conclusions and Open Problems

Shadowless Solutions: As a first step towards developing general techniques for establishing the fixed-parameter tractability of problems on directed graphs, we developed the framework of shadowless solutions. We gave a general family of problems, called as *"Finding an $\mathcal{F}$-transversal for some T-connected $\mathcal{F}$"*, for which we can do random sampling of important separators and obtain a set which is disjoint from a minimum solution and covers its shadow. After taking the `torso` operation (which is problem-specific, but usually straightforward) the general problem reduces to finding a shadowless solution. For the DIRECTED MULTIWAY CUT problem and the SUBSET-DFVSproblem, we were able to analyze the structure of shadowless solutions and find them in FPT time thus giving the first FPT algorithms for these problems parameterized by the size of the solution. Our algorithms used various tools from the FPT world such as iterative compression, bounded-depth search trees, random sampling of important separators, splitters, etc. We believe that this general approach will be useful for deciding the fixed-parameter tractability status of other problems in directed graphs. We now describe the two main open problems for directed graphs. Both these problem fit within our template of *"Finding an $\mathcal{F}$-transversal for some T-connected $\mathcal{F}$"*. The respective `torso` operations are also clear and hence it is enough to find shadowless solutions. Unfortunately, we are not yet able to understand the structure of shadowless solutions for these problems well enough to find them in FPT time.

DIRECTED MULTICUT
Input : A directed graph $G = (V,E)$, an integer k and a set of terminal pairs $T = \{(s_1,t_1),(s_2,t_2),\ldots,(s_p,t_p)\}$.
Parameter: $k+p$
Output : A set $S \subseteq V(G)$ of size at most k such that $G \setminus S$ has no $s_i \rightsquigarrow t_i$ path for any $i \in [p]$, or "NO" if such a set does not exist.

Open Problem 1: Is DIRECTED MULTICUT FPT parameterized by $p+k$? We know that the problem is FPT on directed acyclic graphs [KPPW12b]. Is DIRECTED MULTICUT FPT parameterized by k for $p = 3$? In Section 2.2.3, we gave an FPT algorithm parameterized by k for $p = 2$. For general p, it is known that the problem is W[1]-hard parameterized by k [MR14b].

DIRECTED ODD CYCLE TRANSVERSAL
Input : A directed graph $G = (V,E)$ and an integer k.
Parameter: k
Output : A set $S \subseteq V(G)$ of size at most k such that $G \setminus S$ has no directed cycles of odd length, or "NO" if such a set does not exist.

Open Problem 2: Is DIRECTED ODD CYCLE TRANSVERSAL FPT parameterized by k? The undirected version is known to be FPT parameterized by k [RSV04].

Connectivity Problems: We considered two generalizations of the DIRECTED STEINER TREE (DST) problem, namely the STRONGLY CONNECTED STEINER SUBGRAPH (SCSS) problem (by requiring two-way connectivity) and the DIRECTED STEINER FOREST (DSF) problem (by requiring connectivity between terminal pairs). Feldman and Ruhl [FR06] gave $n^{O(k)}$ algorithms for both these problems where k is the number of terminals/terminal pairs. In the second part of this thesis, we obtained the following results:

- An $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$ algorithm for SCSS on planar graphs (and more generally H-minor-free graphs for any fixed graph H). We modify the Feldman-Ruhl game in a highly non-trivial way to obtain a planar structure and then use the Excluded Grid Minor theorem for planar graphs.
- A matching lower bound for our algorithm for SCSS on planar graphs: there is no $f(k) \cdot n^{o(\sqrt{k})}$ algorithm under ETH, for any function f.
- An almost tight lower bound for the Feldman-Ruhl algorithm for SCSS on general graphs: there is no $f(k) \cdot n^{o(k/\log k)}$ algorithm under ETH, for any function f.
- A tight lower bound for the Feldman-Ruhl algorithm for DSF: there is no $f(k) \cdot n^{o(k)}$ algorithm under ETH, for any function f.

Our results leave open the following two open problems:

Open Problem 3: The best algorithm for SCSS on general graphs is $n^{O(k)}$ [FR06] and the best known hardness is $f(k) \cdot n^{o(k/\log k)}$. What is the correct time complexity?

A more general problem than SCSS and DSF is the DIRECTED STEINER NETWORK (DSN) problem:

DIRECTED STEINER NETWORK
Input : A directed graph $G = (V,E)$, a set of terminal pairs $T = \{(s_1,t_1),(s_2,t_2),\dots,(s_k,t_k)\}$, and a set of demands $d_1,d_2,\dots,d_k$.
Output : A set $S \subseteq V(G)$ of minimum size $G[S]$ has d_i disjoint $s_i \rightsquigarrow t_i$ paths for each $i \in [k]$.

Open Problem 4: Can we design an algorithm for DSN which runs in time $n^{O(d)}$ where $d = \sum_{i=1}^{k} d_i$? This would generalize the $n^{O(k)}$ algorithm of Feldman and Ruhl for DSF.

Bibliography

[AAC07] Amit Agarwal, Noga Alon, and Moses Charikar. Improved Approximation for Directed Cut Problems. In *STOC*, pages 671–680, 2007. 23

[AGM$^+$10] Arash Asadpour, Michel X. Goemans, Aleksander Madry, Shayan Oveis Gharan, and Amin Saberi. An O(log n/ log log n)-Approximation Algorithm for the Asymmetric Traveling Salesman Problem. In *SODA*, pages 379–389, 2010. 23

[AKK$^+$11] Isolde Adler, Stavros G. Kolliopoulos, Philipp Klaus Krause, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Tight Bounds for Linkages in Planar Graphs. In *ICALP (1)*, pages 110–121, 2011. 18

[Aro98] Sanjeev Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems. *J. ACM*, 45(5):753–782, 1998. 8

[BBF99] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-Approximation Algorithm for the Undirected Feedback Vertex Set Problem. *SIAM J. Discrete Math.*, 12(3):289–297, 1999. 23

[BBM$^+$13] Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Approximation Algorithms for Spanner Problems and Directed Steiner Forest. *Inf. Comput.*, 222:93–107, 2013. 98

[BBYG00] Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized Algorithms for the Loop Cutset Problem. *J. Artif. Intell. Res. (JAIR)*, 12:219–234, 2000. 25, 74

[BCE$^+$11] MohammadHossein Bateni, Chandra Chekuri, Alina Ene, Mohammad Taghi Hajiaghayi, Nitish Korula, and Dániel Marx. Prize-Collecting Steiner Problems on Planar Graphs. In *SODA*, pages 1028–1049, 2011. 100

[BDD$^+$13] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. An $O(c^k \cdot n)$ 5-Approximation Algorithm for Treewidth. In *FOCS*, pages 499–508, 2013. 125

[BDFH09] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On Problems Without Polynomial Kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. 16

[BDHK06] Dietmar Berwanger, Anuj Dawar, Paul Hunter, and Stephan Kreutzer. DAG-Width and Parity Games. In *STACS*, pages 524–536, 2006. 24

[BDT11] Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. Multicut is FPT. In *STOC*, pages 459–468, 2011. 72

[BEKP13] Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th. Paschos. On Subexponential and FPT-Time Inapproximability. In *IPEC*, pages 54–65, 2013. 9

[Bel62] Richard Bellman. Dynamic Programming Treatment of the Travelling Salesman Problem. *Journal of the ACM (JACM)*, 9(1):61–63, 1962. 10

[BEP09] Nicolas Bourgeois, Bruno Escoffier, and Vangelis Th. Paschos. Efficient Approximation of Min Set Cover by Moderately Exponential Algorithms. *Theor. Comput. Sci.*, 410(21-23):2184–2195, 2009. 9

[BFL$^+$09] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. In *FOCS*, pages 629–638, 2009. 100

[BFPV13] Ivan Bliznets, Fedor V. Fomin, Michal Pilipczuk, and Yngve Villanger. Largest Chordal and Interval Subgraphs Faster Than 2^n. In *ESA*, pages 193–204, 2013. 10

[BG04] Dietmar Berwanger and Erich Grädel. Entanglement - A Measure for the Complexity of Directed Graphs with Applications to Logic and Games. In *LPAR*, pages 209–223, 2004. 24

[BH06] Andreas Björklund and Thore Husfeldt. Inclusion–Exclusion Algorithms for Counting Set Partitions. In *FOCS*, pages 575–582, 2006. 9

[BHKK07] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier Meets Möbius: Fast Subset Convolution. In *STOC*, pages 67–74, 2007. 99

[BHM11] MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Dániel Marx. Approximation Schemes for Steiner Forest on Planar Graphs and Graphs of Bounded Treewidth. *J. ACM*, 58(5):21, 2011. 100

[Bjö14] Andreas Björklund. Determinant Sums for Undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. 10

[BKM09] Glencora Borradaile, Philip N. Klein, and Claire Mathieu. An $O(n \log n)$ Approximation Scheme for Steiner Tree in Planar Graphs. *ACM Transactions on Algorithms*, 5(3), 2009. 100

[BL11] Paul Bonsma and Daniel Lokshtanov. Feedback Vertex Set in Mixed Graphs. In *WADS*, pages 122–133, 2011. 74

[BLP09] Hans L. Bodlaender, Daniel Lokshtanov, and Eelko Penninkx. Planar Capacitated Dominating Set Is $W[1]$-Hard. In *IWPEC*, pages 50–60, 2009. 168

[Bod91] Hans L. Bodlaender. On Disjoint Cycles. In *WG*, pages 230–238, 1991. 25, 74

[CC14] Chandra Chekuri and Julia Chuzhoy. Polynomial Bounds for the Grid-Minor Theorem. In *STOC*, pages 60–69, 2014. 19

[CCC$^+$99] Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation Algorithms for Directed Steiner Problems. *J. Algorithms*, 33(1):73–91, 1999. 98

[CCH$^+$12] Rajesh Hemant Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT Algorithms for Cut Problems Using Randomized Contractions. In *FOCS*, pages 460–469, 2012. 18

[CCHM12] Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. Directed Subset Feedback Vertex Set Is Fixed-Parameter Tractable. In *ICALP (1)*, pages 230–241, 2012. 20, 26, 29

[CCL10] Yixin Cao, Jianer Chen, and Yang Liu. On Feedback Vertex Set: New Measure and New Structures. In *SWAT*, pages 93–104, 2010. 25, 74

[CFG13] Rajesh Hemant Chitnis, Fedor V. Fomin, and Petr A. Golovach. Parameterized Complexity of the Anchored k-Core Problem for Directed Graphs. In *FSTTCS*, pages 79–90, 2013. 20

[CFJR07] Liming Cai, Michael R. Fellows, David W. Juedes, and Frances A. Rosamond. The Complexity of Polynomial-Time Approximation. *Theory Comput. Syst.*, 41(3):459–477, 2007. 168

[CFL$^+$08] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved Algorithms for Feedback Vertex Set Problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008. 16, 25, 74, 77

[CFL$^+$13] Rajesh Hemant Chitnis, Fedor V. Fomin, Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, and Saket Saurabh. Faster Exact Algorithms for Some Terminal Set Problems. In *IPEC*, pages 150–162, 2013. 10

[CHK13] Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Guy Kortsarz. Fixed-Parameter and Approximation Algorithms: A New Look. In *IPEC*, pages 110–122, 2013. 9

[CHKX06] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong Computational Lower Bounds via Parameterized Complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. 15, 168, 184, 212, 228

[CHM14] Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Tight Bounds for Planar Strongly Connected Steiner Subgraph with Fixed Number of Terminals (and Extensions). In *SODA*, pages 1782–1801, 2014. 26

[CHM12] R. Chitnis, M. Hajiaghayi, and D. Marx. Fixed-Parameter Tractability of Directed Multiway Cut Parameterized by the Size of the Cutset. *SIAM Journal on Computing*, 42(4):1674–1696, 2013. A preliminary version appeared in SODA '12. 20, 25, 26, 28, 29, 38

[Chr76] Nicos Christofides. Worst-case Analysis of a New Heuristic for the Travelling Salesman Problem. Technical report, Technical Report 388, Graduate School of Industrial Administration, CMU, 1976. 23

[CK09] Julia Chuzhoy and Sanjeev Khanna. Polynomial Flow-Cut Gaps and Hardness of Directed Cut Problems. *J. ACM*, 56(2), 2009. 23

[CKK$^+$06] Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. On the Hardness of Approximating Multicut and Sparsest-Cut. *Computational Complexity*, 15(2):94–114, 2006. 23

[CKW09] Marek Cygan, Lukasz Kowalik, and Mateusz Wykurz. Exponential-time Approximation of Weighted Set Cover. *Inf. Process. Lett.*, 109(16):957–961, 2009. 9

[CKX10] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved Upper Bounds for Vertex Cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. 15

[CLL$^+$08] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A Fixed-Parameter Algorithm for the Directed Feedback Vertex Set Problem. *J. ACM*, 55(5), 2008. 16, 20, 26, 74, 76, 77

[CLL09] Jianer Chen, Yang Liu, and Songjian Lu. An Improved Parameterized Algorithm for the Minimum Node Multiway Cut Problem. *Algorithmica*, 55(1):1–13, 2009. 20, 21, 29, 34

[CMPP13] Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. The Planar Directed K-Vertex-Disjoint Paths Problem Is Fixed-Parameter Tractable. In *FOCS*, pages 197–206, 2013. 18

[CNP$^+$11] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *FOCS*, pages 150–159, 2011. 25, 74

[Coo71] Stephen A. Cook. The Complexity of Theorem-Proving Procedures. In *STOC*, pages 151–158, 1971. 6

[CPPW13a] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. On Multiway Cut Parameterized Above Lower Bounds. *TOCT*, 5(1):3, 2013. 29, 61

[CPPW13b] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Subset Feedback Vertex Set Is Fixed-Parameter Tractable. *SIAM J. Discrete Math.*, 27(1):290–309, 2013. 75

[DF95] Rodney G. Downey and Michael R. Fellows. Fixed-Parameter Tractability and Completeness II: On Completeness for W[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995. 13

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999. 530 pp. 11, 25, 74

[DF13] Rodney G Downey and Michael R Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. 11, 12

[DFHT04] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Bidimensional Parameters and Local Treewidth. *SIAM J. Discrete Math.*, 18(3):501–511, 2004. 20

[DFHT05a] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-Parameter Algorithms for (k, r)-center in Planar Graphs and Map Graphs. *ACM Transactions on Algorithms*, 1(1):33–47, 2005. 20

[DFHT05b] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential Parameterized Algorithms on Bounded-Genus Graphs and H-minor-free Graphs. *J. ACM*, 52(6):866–893, 2005. 19, 20

[DFL$^+$07] Frank Dehne, Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Kim Stevens. An $O(2^{O(k)}n^3)$ FPT Algorithm for the Undirected Feedback Vertex Set Problem. *Theory Comput. Syst.*, 41(3):479–492, 2007. 16, 25, 74, 77

[DFMR08] Rodney G. Downey, Michael R. Fellows, Catherine McCartin, and Frances A. Rosamond. Parameterized Approximation of Dominating Set Problems. *Inf. Process. Lett.*, 109(1):68–70, 2008. 9

[DFRS04] Frank Dehne, Michael R. Fellows, Frances A. Rosamond, and Peter Shaw. Greedy Localization, Iterative Compression, Modeled Crown Reductions: New FPT Techniques, an Improved Algorithm for Set Splitting, and a Novel $2k$ Kernelization for Vertex Cover. In *IWPEC*, pages 271–280, 2004. 17

[DFT08] Frederic Dorn, Fedor V. Fomin, and Dimitrios M. Thilikos. Subexponential Parameterized Algorithms. *Computer Science Review*, 2(1):29–39, 2008. 20

[DGHK01] Evgeny Dantsin, Michael Gavrilovich, Edward A. Hirsch, and Boris Konev. MAX SAT Approximation Beyond the Limits of Polynomial-Time Approximation. *Ann. Pure Appl. Logic*, 113(1-3):81–94, 2001. 9

[DH04a] Erik D. Demaine and Mohammad Taghi Hajiaghayi. Diameter and Treewidth in Minor-Closed Graph Families, Revisited. *Algorithmica*, 40(3):211–215, 2004. 20

[DH04b] Erik D. Demaine and Mohammad Taghi Hajiaghayi. Equivalence of Local Treewidth and Linear Local Treewidth and its Algorithmic Applications. In *SODA*, pages 840–849, 2004. 20

[DH05a] Erik D. Demaine and Mohammad Taghi Hajiaghayi. Bidimensionality: New Connections between FPT Algorithms and PTASs. In *SODA*, pages 590–601, 2005. 20

[DH05b] Erik D. Demaine and Mohammad Taghi Hajiaghayi. Graphs Excluding a Fixed Minor Have Grids as Large as Treewidth, with Combinatorial and Algorithmic Applications Through bBidimensionality. In *SODA*, pages 682–689, 2005. 19, 20, 109, 124

[DH08] Erik D. Demaine and MohammadTaghi Hajiaghayi. The Bidimensionality Theory and Its Algorithmic Applications. *Comput. J.*, 51(3):292–302, 2008. 20, 100

[DHK09] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Philip N. Klein. Node-Weighted Steiner Tree and Group Steiner Tree in Planar Graphs. In *ICALP (1)*, pages 328–340, 2009. 100

[DHN$^+$04] Erik D. Demaine, Mohammad Taghi Hajiaghayi, Naomi Nishimura, Prabhakar Ragde, and Dimitrios M. Thilikos. Approximation Algorithms for Classes of Graphs Excluding Single-Crossing Graphs as Minors. *J. Comput. Syst. Sci.*, 69(2):166–195, 2004. 20

[DHT05] Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Exponential Speedup of Fixed-Parameter Algorithms for Classes of Graphs Excluding Single-Crossing Graphs as Minors. *Algorithmica*, 41(4):245–267, 2005. 20

[DHT06] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Dimitrios M. Thilikos. The Bidimensional Theory of Bounded-Genus Graphs. *SIAM J. Discrete Math.*, 20(2):357–371, 2006. 20

[DJP$^+$94] Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The Complexity of Multiterminal Cuts. *SIAM J. Comput.*, 23(4):864–894, 1994. 60

[Dru12] Andrew Drucker. New Limits to Classical and Quantum Instance Compression. In *FOCS*, pages 609–618, 2012. 16

[DS05] Irit Dinur and Samuel Safra. On the Hardness of Approximating Minimum Vertex Cover. *Annals of Mathematics*, pages 439–485, 2005. 8

[DvM10] Holger Dell and Dieter van Melkebeek. Satisfiability Allows no Nontrivial Sparsification Unless the Polynomial-Time Hierarchy Collapses. In *STOC*, pages 251–260, 2010. 16

[DW71] S. E. Dreyfus and R. A. Wagner. The Steiner Problem in Graphs. *Networks*, 1(3):195–207, 1971. 99

[Edm65] Jack Edmonds. Paths, Trees, and Flowers. *Canad. J. Math.*, 17:449–467, 1965. 5

[EFG$^+$09] Rosa Enciso, Michael R. Fellows, Jiong Guo, Iyad A. Kanj, Frances A. Rosamond, and Ondrej Suchý. What Makes Equitable Connected Partition Easy. In *IWPEC*, pages 122–133, 2009. 168

[EKM12] David Eisenstat, Philip N. Klein, and Claire Mathieu. An Efficient Polynomial-time Approximation Scheme for Steiner Forest in Planar Graphs. In *SODA*, pages 626–638, 2012. 100

[ENSS95] Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating Minimum Feedback Sets and Multi-Cuts in Directed Graphs. In *IPCO*, pages 14–28, 1995. 23

[Fei98] U. Feige. A Threshold of $\ln n$ for Approximating Set Cover. *Journal of the ACM*, 45(4):634–652, 1998. 8

[FF56] L.R. Ford and D.R. Fulkerson. Maximal Flow Through A Network. *Canad. J. Math.*, 8:399–404, 1956. 60

[FF62] L. R. Ford, Jr. and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, N.J., 1962. 59

[FG01] Markus Frick and Martin Grohe. Deciding First-Order Properties of Locally Tree-decomposable Structures. *J. ACM*, 48(6):1184–1206, 2001. 100, 101

[FG06] Jorge Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006. 493 pp. 11

[FGK$^+$10] Fedor V. Fomin, Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Saket Saurabh. Iterative Compression and Exact Algorithms. *Theor. Comput. Sci.*, 411(7-9):1045–1053, 2010. 17, 77

[FGPR08] Fedor V. Fomin, Serge Gaspers, Artem V. Pyatkin, and Igor Razgon. On the Minimum Feedback Vertex Set Problem: Exact and Enumeration Algorithms. *Algorithmica*, 52(2):293–307, 2008. 10

[FGT09] Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. Contraction Bidimensionality: The Accurate Picture. In *ESA*, pages 706–717, 2009. 20

[FK10] Fedor V Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer, 2010. 10

[FLRS11] Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In *SODA*, pages 748–759, 2011. 20

[FLS12] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Bidimensionality and Geometric Graphs. In *SODA*, pages 1563–1575, 2012. 20

[FLST10] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and Kernels. In *SODA*, pages 503–510, 2010. 20

[FR06] Jon Feldman and Matthias Ruhl. The Directed Steiner Network Problem is Tractable for a Constant Number of Terminals. *SIAM J. Comput.*, 36(2):543–561, 2006. 99, 101, 103, 104, 106, 107, 108, 126, 217, 228, 231

[FS11] Lance Fortnow and Rahul Santhanam. Infeasibility of Instance Compression and Succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. 16

[GG07] Martin Grohe and Magdalena Grüber. Parameterized Approximability of the Disjoint Cycle Problem. In *ICALP*, pages 363–374, 2007. 9

[GGH$^+$06] Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-Based Fixed-Parameter Algorithms for Feedback Vertex Set and Edge Bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006. 16, 25, 74, 77

[GHK$^+$10] Robert Ganian, Petr Hlinený, Joachim Kneis, Daniel Meister, Jan Obdrzálek, Peter Rossmanith, and Somnath Sikdar. Are There Any Good Digraph Width Measures? In *IPEC*, pages 135–146, 2010. 24

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Co., San Francisco, Calif., 1979. 6, 7

[GNS11] Jiong Guo, Rolf Niedermeier, and Ondrej Suchý. Parameterized Complexity of Arc-Weighted Directed Steiner Problems. *SIAM J. Discrete Math.*, 25(2):583–599, 2011. 99, 100, 167, 211, 212

[Gui11] Sylvain Guillemot. FPT Algorithms for Path-Transversal and Cycle-Transversal Problems. *Discrete Optimization*, 8(1):61–71, 2011. 67, 71

[GVY96] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate Max-Flow Min-(Multi)Cut Theorems and Their Applications. *SIAM J. Comput.*, 25(2):235–251, 1996. 23

[GVY04] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Multiway Cuts in Node Weighted Graphs. *J. Algorithms*, 50(1):49–61, 2004. 22, 60, 61

[Hak71] S L Hakimi. Steiner's Problem in Graphs and Its Implications. *Networks*, 1:113–133, 1971. 96

[Hås96] Johan Håstad. Clique is Hard to Approximate Within $n^{1-\varepsilon}$. In *FOCS*, 1996. 8

[HK] Eran Halperin and Robert Krauthgamer. Polylogarithmic Inapproximability. *STOC '03*, pages 585–594. 98

[HK62] Michael Held and Richard M Karp. A Dynamic Programming Approach to Sequencing Problems. *Journal of the Society for Industrial & Applied Mathematics*, 10(1):196–210, 1962. 10

[HK08] Paul Hunter and Stephan Kreutzer. Digraph Measures: Kelly Decompositions, Games, and Orderings. *Theor. Comput. Sci.*, 399(3):206–219, 2008. 24

[HKMN08] Falk Hüffner, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Fixed-Parameter Algorithms for Cluster Vertex Deletion. In *LATIN*, pages 711–722, 2008. 16, 77

[Hoc97] Ed. Dorit Hochbaum. *Approximation Algorithms for NP-hard problems*. PWS Publishing Co., Boston, MA, USA, 1997. 8

[iKK14] Ken ichi Kawarabayashi and Stephan Kreutzer. An Excluded Grid Theorem for Digraphs with Forbidden Minors. In *SODA*, pages 72–81, 2014. 25

[iKT11] Ken ichi Kawarabayashi and Mikkel Thorup. The Minimum k-way Cut of Bounded Size is Fixed-Parameter Tractable. In *FOCS*, pages 160–169, 2011. 18

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2), 2001. 15, 167

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001. 15, 167

[JLR$^+$13] Mark Jones, Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Ondrej Suchý. Parameterized Complexity of Directed Steiner Tree on Sparse Graphs. In *ESA*, pages 671–682, 2013. 101

[JRST01a] Thor Johnson, Neil Robertson, Paul D. Seymour, and Robin Thomas. Directed Tree-Width. *J. Comb. Theory, Ser. B*, 82(1):138–154, 2001. 24, 25

[JRST01b] Thor Johnson, Neil Robertson, Paul D. Seymour, and Robin Thomas. Excluding a Grid Minor in Directed Graphs. *Unplublished manuscript*, 2001. 25

[Kar72] Richard M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103, 1972. 6, 74, 96

[Kho02] Subhash Khot. On the Power of Unique 2-prover 1-round Games. In *STOC*, pages 767–775, 2002. 8, 23

[KKK12] Naonori Kakimura, Kenichi Kawarabayashi, and Yusuke Kobayashi. Erdös-Pósa Property and Its Algorithmic Applications: Parity Constraints, Subset Feedback Set, and Subset Packing. In *SODA*, pages 1726–1736, 2012. 75

[KM] Philip N. Klein and Dániel Marx. Solving Planar k-Terminal Cut in $O(n^{c\sqrt{k}})$ time. *ICALP '12*, pages 569–580. 109, 125

[KPPW12a] Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Magnus Wahlström. Fixed-Parameter Tractability of Multicut in Directed Acyclic Graphs. In *ICALP (1)*, pages 581–593, 2012. 20, 25

[KPPW12b] Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Magnus Wahlström. Fixed-Parameter Tractability of Multicut in Directed Acyclic Graphs. In *ICALP (1)*, pages 581–593, 2012. 230

[KPS04] Iyad A. Kanj, Michael J. Pelsmajer, and Marcus Schaefer. Parameterized Algorithms for Feedback Vertex Set. In *IWPEC*, pages 235–247, 2004. 25, 74

[KR08] Subhash Khot and Oded Regev. Vertex Cover Might be Hard to Approximate to Within $2-\varepsilon$. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. 8

[Lev71] A Levin. Algorithm for the Shortest Connection of a Group of Graph Vertices. *Soviet Math. Dokl.*, 12:1477–1481, 1971. 96

[LM13] Daniel Lokshtanov and Dániel Marx. Clustering With Local Restrictions. *Inf. Comput.*, 222:278–292, 2013. 20

[LMS12] Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - Preprocessing with a Guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161, 2012. 16

[LMSL92] Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. The point-to-point delivery and connection problems: Complexity and algorithms. *Discrete Applied Mathematics*, 36(3):267–292, 1992. 99

[LNR$^+$12] Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster Parameterized Algorithms using Linear Programming. *CoRR*, abs/1203.0833, 2012. 29

[LR12] Daniel Lokshtanov and M. S. Ramanujan. Parameterized Tractability of Multiway Cut with Parity Constraints. In *ICALP (1)*, pages 750–761, 2012. 20

[Mara] Dániel Marx. http://www.cs.bme.hu/ dmarx/papers/bertinoro-spider-talk.pdf. 21

[Marb] Dániel Marx. http://www.cs.bme.hu/ dmarx/papers/marx-warsaw-fpt3. 13

[Marc] Dániel Marx. On the Optimality of Planar and Geometric Approximation Schemes. *FOCS '07*, pages 338–348. 168, 183, 184, 218, 228

[Mar06] Dániel Marx. Parameterized Graph Separation Problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. 20, 21, 25, 29, 34, 61

[Mar08] Dániel Marx. Parameterized Complexity and Approximation Algorithms. *Comput. J.*, 51(1):60–78, 2008. 9

[Mar10] Dániel Marx. Can You Beat Treewidth? *Theory of Computing*, 6(1):85–112, 2010. 212

[Mar12a] Dániel Marx. A Tight Lower Bound for Planar Multiway Cut with Fixed Number of Terminals. In *ICALP (1)*, pages 677–688, 2012. 168

[Mar12b] Dániel Marx. Randomized Techniques for Parameterized Algorithms. In *IPEC*, page 2, 2012. 9

[Mar13] Dániel Marx. Completely Inapproximable Monotone and Antimonotone Parameterized Problems. *J. Comput. Syst. Sci.*, 79(1):144–151, 2013. 9

[Meh84] Kurt Mehlhorn. *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*. Springer, 1984. 25, 74

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. 9

[MR09] Dániel Marx and Igor Razgon. Constant Ratio Fixed-parameter Approximation of the Edge Multicut problem. *Inf. Process. Lett.*, 109(20):1161–1166, 2009. 17

[MR11] Dániel Marx and Igor Razgon. Fixed-parameter Tractability of Multicut Parameterized by the Size of the Cutset. In *STOC*, pages 469–478, 2011. 16, 20, 27, 28, 29, 33, 38, 39, 61, 72, 73, 77

[MR14a] Dániel Marx and Igor Razgon. Fixed-Parameter Tractability of Multicut Parameterized by the Size of the Cutset. *SIAM J. Comput.*, 43(2):355–388, 2014. 25, 27

[MR14b] Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM J. Comput.*, 43(2):355–388, 2014. 230

[MRS11] Neeldhara Misra, Venkatesh Raman, and Saket Saurabh. Lower Bounds on Kernelization. *Discrete Optimization*, 8(1):110–128, 2011. 16

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. 9

[NF97] Madan Natu and Shu-Cherng Fang. The Point-to-Point Connection Problem - Analysis and Algorithms. *Discrete Applied Mathematics*, 78(1-3):207–226, 1997. 99

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. 312 pp. 11

[NSS95] Joseph Naor, Leonard Schulman, and Aravind Srinivasan. Splitters and Near-Optimal Derandomization. In *FOCS*, 1995. pages 182-191. 44

[Obd06] Jan Obdrzálek. DAG-width: Connectivity Measure for Directed Graphs. In *SODA*, pages 814–821, 2006. 24

[PPSvL13] Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Subexponential-Time Parameterized Algorithm for Steiner Tree on Planar Graphs. In *STACS*, pages 353–364, 2013. 101

[Pul12] William R. Pulleyblank. Edmonds, Matching and the Birth of Polyhedral Combinatorics. *Documenta Mathematica*, pages 181–197, 2012. 5

[Ram96] S Ramanathan. Multicast Tree Generation in Networks with Asymmetric Links. *IEEE/ACM Transactions on Networking (TON)*, 4(4):558–568, 1996. 97

[Raz07] Igor Razgon. Computing Minimum Directed Feedback Vertex Set in $O(1.9977^n)$. In *ICTCS*, pages 70–81, 2007. 10

[RO09] Igor Razgon and Barry O'Sullivan. Almost 2-SAT is Fixed-Parameter Tractable. *J. Comput. Syst. Sci.*, 75(8):435–450, 2009. 16, 20, 29, 77

[Rob86] J. M. Robson. Algorithms for Maximum Independent Sets. *J. Algorithms*, 7(3):425–440, 1986. 14

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a Planar Graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986. 18, 19

[RS04] Neil Robertson and Paul D. Seymour. Graph Minors. XX. Wagner's Conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004. 18

[RSS06] Venkatesh Raman, Saket Saurabh, and C. R. Subramanian. Faster Fixed Parameter Tractable Algorithms for Finding Feedback Vertex Sets. *ACM Transactions on Algorithms*, 2(3):403–415, 2006. 25, 74

[RST94] Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly Excluding a Planar Graph. *J. Comb. Theory, Ser. B*, 62(2):323–348, 1994. 19

[RSV04] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding Odd Cycle Transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. 16, 18, 26, 77, 230

[Saf05] Mohammad Ali Safari. D-Width: A More Natural Measure for Directed Tree Width. In *MFCS*, pages 745–756, 2005. 24

[Sey95] Paul D. Seymour. Packing Directed Circuits Fractionally. *Combinatorica*, 15(2):281–288, 1995. 23

[SRV97] Hussein F. Salama, Douglas S. Reeves, and Yannis Viniotis. Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks. *Selected Areas in Communications, IEEE Journal on*, 15(3):332–345, 1997. 97

[Sta] CS Theory StackExchange. Directed problems that are easier than their undirected variant. 22

[Tur36] Alan Mathison Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *J. of Math*, 58:345–363, 1936. 5

[Vaz02] V. Vazirani. Approximation Algorithms. *Springer Verlag*, 2002. 8

[VWW06] Virginia Vassilevska, Ryan Williams, and Shan Leung Maverick Woo. Confronting Hardness Using a Hybrid Approach. In *SODA*, pages 1–10, 2006. 9

[Wig07] Avi Wigderson. P, NP and Mathematics: A Computational Complexity Perspective. *International Congress of Mathematicians (ICM 06), EMS Publishing House, Zurich*, 1:665–712, 2007. 10

[Win87] Pawel Winter. Steiner Problem in Networks: A Survey. *Networks*, 17(2):129–167, 1987. 97

[WS11] David P Williamson and David B Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. 8

[Xia10] Mingyu Xiao. Simple and Improved Parameterized Algorithms for Multiterminal Cuts. *Theory Comput. Syst.*, 46(4):723–736, 2010. 61

[YKCP83] Mihalis Yannakakis, Paris C. Kanellakis, Stavros S. Cosmadakis, and Christos H. Papadimitriou. Cutting and Partitioning a Graph aifter a Fixed Pattern (Extended Abstract). In *ICALP*, pages 712–722, 1983. 22

[Zuc06] David Zuckerman. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. In *STOC*, pages 681–690, 2006. 8