New Graph Algorithms via Polyhedral Techniques

Thèse N° 9622

Présentée le 28 juin 2019

à la Faculté informatique et communications
Laboratoire de théorie du calcul 2
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Jakub TARNAWSKI

Acceptée sur proposition du jury

Prof. R. Urbanke, président du jury
Prof. O. N. A. Svensson, directeur de thèse
Prof. J. Håstad, rapporteur
Prof. A. Saberi, rapporteur
Prof. M. Kapralov, rapporteur

2019

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

to my family

Acknowledgments

The five years I spent at EPFL have been a crucial time for me, and I have many people to thank for making my Ph.D. both successful and enjoyable. The following is a certainly incomplete attempt at this.

First, I would like to express my sincere gratitude to my advisor Ola. He resoundingly qualifies as a role model of a researcher, with a great taste for problems and a winning strategy for solving them powered by perseverance, excitement, intuition and brilliance. The working environment Ola has created is top-notch, bringing out the best qualities in all of his students and maintaining high standards and a strong drive for excellence without exerting undue stress. I wish I could offer him some (unsolicited) constructive criticism, but nothing really comes to my mind. His contagious enthusiasm and curiosity go beyond science, making him a singularly fun person to be around.

I would like to thank the other members of my thesis committee: Johan Håstad, Michael Kapralov, Amin Saberi, and Rüdiger Urbanke, for their careful reading of this work and the engaging discussions we have had around the defense. I thank Simon Meinhard for his help with the German version of the abstract.

I am also grateful to people who have guided me at various points of my career (at the University of Wrocław, EPFL, and Microsoft Research), from whom I have learned a lot: Nikhil Devanur, Janardhan Kulkarni, Krzysztof Loryś, Aleksander Mądry, and Nisheeth Vishnoi. I thank all researchers and institutions who have invited me for visits for their kind hospitality.

It has been a true honor to meet, think, and write papers together with many amazing co-authors and valued friends: Ilija Bogunović, Volkan Cevher, Pascal Fua, Christos Kalaitzis, Silvio Lattanzi, Shi Li, Agata Mosińska-Domańska, Aida Mousavifar, Ashkan Norouzi-Fard, László Végh, Morteza Zadimoghaddam, and Amir Zandieh. Apart from our collaborations, special thanks go to Slobodan Mitrović also for being a natural (hike) group leader, and to Damian Straszak for having shared his passion with me for 11 years now, during which we have taken over 60 courses together and participated in countless programming competitions.

It has been an immense pleasure to be part of the Theory of Computation lab and to share an office with many wonderful people. They are directly responsible for most of the fun times I have had during my studies. I thank all my friends in Lausanne for their great company. Credit is also due to administrative staff, especially Chantal, for keeping things in order in the most cheerful way imaginable.

I fondly remember all the programming contests organized together with my friends in the PolyProg association. It is through no coincidence that they were all so much fun and went without any hiccups that I can recall.

I begrudgingly acknowledge the role of EPFL's many cafeterias in making me appreciate that good food is not always a given.

Finally, and most importantly, I want to express my boundless gratitude to my family, to whom this thesis is dedicated, for their endless love, devotion and support. I thank my parents

Teresa and Dariusz for all their selfless care and giving me better conditions to develop than anyone could ever ask for. I am grateful to my brother Krzysiek for being the best company to grow up with and always having my back. Finally, I thank my lovely newlywed wife Maja for coming with me on this journey, our many travels together, and all her love, understanding and support throughout what will hopefully end up constituting a short prefix of our time together.

Abstract

In this thesis we give new algorithms for two fundamental graph problems. We develop novel ways of using linear programming formulations, even exponential-sized ones, to extract structure from problem instances and to guide algorithms in making progress. Somewhat surprisingly, similar polyhedral techniques can be harnessed in the two seemingly disparate settings.

In the first part of the thesis we address a benchmark problem in combinatorial optimization: the *asymmetric traveling salesman problem* (ATSP). It consists in finding the shortest tour that visits all vertices of a given directed graph with weights on edges. Due to its NP-hardness, the theoretical study of algorithms for ATSP has focused on approximation algorithms: ones that are provably both efficient and give solutions competitive with the optimum. Specifically, a ρ-approximation algorithm for ATSP is one that runs in polynomial time and always outputs a tour that is at most ρ times longer than the shortest tour. Finding such an approximation algorithm with ρ bounded (i.e., a constant factor) had been a long-standing open problem.

In this thesis, we give such an algorithm. Our approximation guarantee is analyzed with respect to the standard linear programming relaxation, and thus our result also confirms the conjectured constant integrality gap of that relaxation. Our techniques build upon the constant-factor approximation algorithm for the special case of node-weighted metrics due to Svensson [Sve15]. In particular, we give a generic reduction to structured instances that resemble but are more general than those arising from node-weighted metrics. This reduction takes advantage of a laminar family of vertex sets that arises from the linear programming relaxation.

In the second part of the thesis we address the *perfect matching problem*. The first polynomial-time algorithm for it, given by Edmonds in 1965 [Edm65b], is historically associated with the introduction of the class P and our notion that "polynomial-time" means "efficient". That algorithm is sequential and deterministic. We have also known since the 1980s that the matching problem has efficient parallel algorithms if the use of randomness is allowed [Lov79, KUW86, MVV87]. Formally, it is in the class RNC, i.e., it has randomized algorithms that use polynomially many processors and run in polylogarithmic time. However, we do not know if randomness is necessary – that is, whether the matching problem is in the class NC.

In this thesis we show that the matching problem is in quasi-NC. That is, we give a deterministic parallel algorithm that runs in $O(\log^3 n)$ time on $n^{O(\log^2 n)}$ processors. The result is obtained by a derandomization of the Isolation Lemma for perfect matchings, which was introduced in the classic paper by Mulmuley, Vazirani and Vazirani [MVV87] to obtain an RNC algorithm. Our proof extends the framework of Fenner, Gurjar and Thierauf [FGT16], who proved the analogous result in the special case of bipartite graphs. Compared to that setting, several new ingredients are needed due to the significantly more complex structure of perfect matchings in general graphs. In particular, our proof heavily relies on the laminar structure of the faces of the perfect matching polytope.

Keywords: graph algorithms, linear programming, traveling salesman problem, perfect

matching, derandomization, parallel algorithms, approximation algorithms, combinatorial optimization, discrete optimization, theoretical computer science

Zusammenfassung

In dieser Arbeit werden neue Algorithmen für zwei grundlegende Graphenprobleme vorgestellt. Mit Hilfe von linearen Programmen – darunter auch Programme exponentieller Größe – werden Struktureigenschaften ermittelt die vom Algorithmus verwendet werden. Etwas überraschend können ähnliche polyedrische Methoden in beiden Graphenproblemen angewendet werden.

Der erste Teil der Dissertation widmet sich dem *asymmetrischen Handlungsreisendenproblem* (Asymmetric Traveling Salesman Problem – ATSP), einem Benchmark-Problem der kombinatorischen Optimierung. Bei diesem Problem geht es darum, die kürzeste Tour für einen gerichteten und kantengewichteten Graphen zu finden, die alle Knoten besucht. Aufgrund der NP-Schwere des ATSP befasst sich die theoretische Forschung hauptsächlich mit Approximationsalgorithmen, die nachweislich sowohl effizient sind und deren Lösung relativ nah an der optimalen Lösung liegt. Ein Approximationsalgorithmus der Güte ρ für ATSP ist ein Algorithmus in Polynomialzeit der eine Tour errechnet, die höchstens ρ mal länger ist als die kürzeste Tour. Seit langem galt es als offen, ob es einen Approximationsalgorithmus für dieses Problem mit beschränktem ρ (einer konstanten Güte) gibt.

Ein Ergebnis dieser Arbeit ist ein solcher Algorithmus. Die Güte des Approximationsalgorithmus wird mit Hilfe der üblichen LP-Relaxation ermittelt. Somit wird auch die Vermutung bestätigt, dass diese Relaxation eine konstante Integralitätslücke aufweist. Die Herangehensweise basiert auf einem Approximationsalgorithmus von Svensson [Sve15] mit konstanter Güte für den Fall knotengewichteter Metriken. Insbesondere wird eine Reduktion zu ähnlich strukturierten Instanzen vorgestellt. Die Instanzen der Reduktion sind allgemeiner als die die sich aus knotengewichteten Metriken ergeben. Dafür wird eine laminare knotenbasierte Mengenfamilie verwendet, die sich aus der LP-Relaxation ergibt.

Der zweite Teil der Dissertation widmet sich dem *perfekten Matching-Problem*. Der erste Algorithmus polynomieller Zeit wurde 1965 von Edmonds entdeckt [Edm65b]. Mit diesem Algorithmus wird historisch auch die Einführung der Komplexitätsklasse P und der Auffassung, dass "Polynomialzeit" "effizient" bedeutet, verbunden. Dieser Algorithmus ist sequentiell und deterministisch. Zudem wurde in den Achtzigerjahren gezeigt, dass es effiziente parallele Algorithmen für das Matching-Problem gibt, sofern die Verwendung von Zufälligkeit zulässig ist [Lov79, KUW86, MVV87]. Formal betrachtet liegt dieses Problem in der Komplexitätsklasse RNC, es gibt dafür also einen randomisierten Algorithmus, der polynomial viele Prozessoren verwendet und in polylogarithmischer Zeit läuft. Allerdings ist es noch offen ob das Matching-Problem auch in der Komplexitätsklasse NC liegt, also ob Zufälligkeit notwending ist.

Diese Arbeit zeigt, dass das Matching-Problem in quasi-NC liegt. Um dies zu beweisen wird ein deterministischer und paralleler Algorithmus vorgestellt, der auf $n^{O(\log^2 n)}$ Prozessoren in Zeit $O(\log^3 n)$ läuft. Dafür wird das von Mulmuley, Vazirani und Vazirani [MVV87] stammende Isolations-Lemma für perfekte Matchings derandomisiert. Das Lemma wurde damals verwendet um zu zeigen, dass das Matching-Problem in RNC liegt. Der Beweis erweitert die Arbeit von Fenner, Gurjar und Thierauf [FGT16], die dieses Ergebnis für bipartite Graphen bewiesen haben.

Aufgrund der wesentlich komplexeren Struktur perferkter Matchings in allgemeinen Graphen sind zusätzliche Herangehensweisen erforderlich. Der Beweis beruht unter anderem auf der laminaren Struktur der Flächen des Polytops für das perfekte Matching-Problem.

Schlüsselwörter: Graphenalgorithmen, lineare Programmierung, Handlungsreisendenproblem, perfektes Matching, Derandomisierung, parallele Algorithmen, Approximationsalgorithmen, kombinatorische Optimierung, diskrete Optimierung, theoretische Informatik

Contents

Chapter 1

Introduction

The design and analysis of algorithms has always been at the core of computer science. As data volumes grow, computational efficiency becomes more and more important. At the same time, increasingly high-consequence decisions are being made either by algorithms or using algorithmic outputs. This necessitates that the solutions produced by our algorithms be of high quality, and that we have a strong understanding of their power and limitations. A rigorous, theoretical study of efficient computation is thus becoming more relevant than ever. In this thesis we focus on the design of algorithms with provable, worst-case guarantees on their performance.

The tasks that we can perform using a computer are limited by the computational resources that we have at our disposal – the most important of them being time. In the classic computational complexity perspective, an algorithm is considered to be efficient if the number of operations it performs on any input can be bounded by a polynomial function of the size of the input. The fundamental class P consists of problems having algorithms that are efficient in the above sense (we call them polynomial-time) and always output the correct solution. These problems are considered to be tractable. One example might be the *shortest path* problem: given a graph with weights (distances) on its edges and two special vertices, what is the length of a shortest path connecting them?

Unfortunately, a large and significant class of fundamental problems, called *NP-hard problems*, are believed to not be tractable. A well-known example is the *traveling salesman problem* (TSP): given a graph with weights (distances) on its edges, what is the length of a shortest tour that visits every vertex of the graph?

Many approaches to dealing with NP-hardness have been developed. For instance, instead of requiring that the algorithm output the shortest tour, we might be satisfied with a tour that is not much longer than possible; however, we would like to have provable guarantees on how long it can be. This brings us to the concept of *approximation algorithms*: for a minimization problem such as the TSP, we say that an algorithm is an *α-approximation* algorithm if it runs in polynomial time and the length of the returned tour is always at most a factor α times longer than the optimal (shortest) tour. The quantity α is called an *approximation ratio* (or *approximation factor*). Given an optimization problem, we ask the question: what is the best approximation ratio possible?

A particular challenge in the analysis of approximation algorithms is to prove, for a given algorithm, that every solution it returns has cost within some factor α of the optimal cost – even though the latter is unknown, and possibly not efficiently computable or easy to reason about.

For this reason, we seek strong lower bounds on the cost of any solution, which can be used in lieu of the optimum values. To illustrate this, let us consider the traveling salesman problem on undirected graphs (called *symmetric TSP*), and the following approximation algorithm: compute the minimum spanning tree (MST) of the graph and take its every edge twice. It is easy to see that this gives rise to a tour. Moreover, as the optimal tour spans the entire graph, it too contains some spanning tree, which is at least as expensive as the MST – thus the MST cost is a lower bound on the cost of the optimal tour. As our algorithm incurs twice the MST cost, it is *a fortiori* a 2-approximation for the symmetric TSP.

A canonical way of obtaining powerful lower bounds is via the use of *linear programming relaxations*. One begins from finding an equivalent formulation of the problem as an integer program. The linear program (LP) is obtained by relaxing the integrality constraints on the variables (hence the name). Crucially, the LP can be solved in polynomial time. The LP solution can be used to guide the algorithm in making decisions, and its value constitutes a lower bound that can be used to analyze the approximation guarantee. The quality (tightness) of this lower bound is measured by a quantity called the *integrality gap*.

The utility of linear programming techniques is however not limited to approximation algorithms for NP-hard problems. Indeed, employing a polyhedral perspective on the considered problem allows an algorithm designer to tap into a deep vault of tools and results from the theory of polyhedra, duality, and polyhedral combinatorics. These methods can be used in diverse settings to better understand the structure of the problem at hand.

In this thesis we employ such techniques to make progress on two basic problems in optimization: the asymmetric version of the traveling salesman problem, and the perfect matching problem. These are both fundamental problems concerning graphs, which are perhaps the most pervasive and important structures in computer science. However, in many other regards, these two problems seem to be very different: while the asymmetric traveling salesman problem (ATSP) is NP-hard, the perfect matching problem is in P, and we are interested in exact algorithms (albeit in a *parallel* setting) rather than approximations. Somewhat surprisingly, we are able to employ similar techniques to obtain crucial structural insights into both problems. Indeed, in both cases our solution begins from considering an exponential-sized linear programming formulation that exhibits a *laminar* structure.

1.1 Our contributions

In Part I of this thesis, we study the asymmetric traveling salesman problem (ATSP) in the context of approximation algorithms and LP relaxations. Specifically, we give the first algorithm for this problem whose approximation ratio is provably bounded by a constant, rather than an unbounded function of the input size. The existence of such an algorithm has been the subject of a long-standing conjecture, which we thus affirmatively resolve. Since the cost of the solutions returned by our algorithm is analyzed with respect to the standard LP relaxation for ATSP, our result also confirms the conjectured constant integrality gap of that relaxation.

Part I (Chapters 3 to 6) is based on a joint work with Ola Svensson and László A. Végh that was published in STOC 2018 [STV18a]. We note that the contents of Chapter 4 are primarily based on a previous work by Svensson [Sve15], but included here for completeness, as well as integrated into the entire argument arc and tuned to optimize the approximation ratio.

Next, we turn to the perfect matching problem, which we study in the parallel setting. Specifically, we consider NC: the class of problems having algorithms that can be executed in

parallel on a polynomial number of processors and in polylogarithmic time. The perfect matching problem, which is to determine whether the vertices of a given graph can be paired up using edges, is not known to be in NC, and it is a major open question to give such an algorithm. In fact, it turns out to be an issue of derandomization, as such algorithms have been known for 40 years in the setting where randomization is allowed. The question of whether all efficient computation can be performed deterministically is a fundamental one in computer science, and computing matchings in parallel is one of several high-profile concrete problems for which we know a (simple) randomized algorithm but no deterministic one.

In Part II of this thesis we make substantial progress towards resolving this question: we give a deterministic algorithm for the perfect matching problem that uses quasi-polynomially many processors (as opposed to polynomially many) and runs in polylogarithmic time. This demonstrates that the matching problem is in the class quasi-NC.

Part II (Chapters 7 and 8) is based on a joint work with Ola Svensson that was published in FOCS 2017 [ST17].

1.2 Outline

This thesis is organized as follows. In Chapter 2 we introduce basic notions related to linear programming and the uncrossing technique. There we also formally define the two problems that we study in this thesis, and discuss their standard linear programming formulations and their properties. We highlight the common polyhedral techniques that will be utilized in both parts of the thesis.

Part I (Chapters 3 to 6) is devoted to ATSP. We begin in Chapter 3 by motivating the problem and discussing its background, as well as using the polyhedral properties of the standard LP relaxation to focus our attention on structured (*laminarly-weighted*) instances. In Chapter 4 we introduce a crucial auxiliary problem called Subtour Partition Cover, which is, roughly speaking, a relaxed version of ATSP, and show that solving this problem is enough to give a constant-factor approximation for ATSP. In Chapter 5 we show that, at the price of losing a constant factor in our approximation guarantee, we can make further strong structural assumptions on ATSP instances. Finally, in Chapter 6, we solve the Subtour Partition Cover problem on those highly structured instances, which, by the results of the previous chapters, yields a constant-factor approximation algorithm for general instances.

Part II (Chapters 7 and 8) is devoted to matching. In Chapter 7 we discuss the parallel setting for the perfect matching problem, as well as introduce and motivate several ingredients that will be needed throughout Part II. Next, in Chapter 8 we present our quasi-NC algorithm for matching.

We conclude with future work directions in Chapter 9.

Chapter 2

Preliminaries and Linear Programming

In this chapter we cover the preliminaries necessary for both parts of the thesis and the common techniques utilized by our approaches to the two (seemingly very different) problems. In particular, we discuss: our notation for graphs (Section 2.1), approximation algorithms (Section 2.2), the definitions of the two problems that are the focal points of this thesis (Sections 2.3 and 2.4), basics of linear programming and the standard formulations for the two problems (Section 2.5), and laminarity and the uncrossing technique (Section 2.6).

2.1 Notation

We let $\mathbb{R}_+$ denote the set of nonnegative real numbers. We work on graphs $G = (V, E)$, where V is a finite nonempty set of n vertices and E is a set of edges, i.e., pairs of vertices. For the matching problem we work with undirected graphs (the pairs are unordered), and for ATSP we work with directed edges (the pairs are ordered). Graphs will often be weighted: there will be an edge-weight function $w : E \to \mathbb{R}_+$ associated with G. We will use the terms "weight", "cost", "value", and "length" interchangeably.

We use the following notation. For a subset $S \subseteq V$ of the vertices, let $\delta(S) = \{(u, v) \in E : |\{u, v\} \cap S| = 1\}$ denote the edges crossing the cut $(S, V \setminus S)$ and $E(S) = \{(u, v) \in E : u, v \in S\}$ denote the edges inside S. For directed graphs, we use $\delta^+(S) = \{(u, v) \in E : u \in S, v \notin S\}$ for the outgoing edges and $\delta^-(S) = \{(u, v) \in E : u \notin S, v \in S\}$ for the incoming edges. We shorten $\delta(\{v\})$ to $\delta(v)$ for $v \in V$, and similarly for δ^+ and δ^-. We let $\delta(S, T) = \{(u, v) \in E : u \in S, v \in T\}$ for $S, T \subseteq V$.

For a vector $x = (x_e)_{e \in E} \in \mathbb{R}^E$ (usually the solution of a linear program or a weight function) and a subset $F \subseteq E$, we define $x(F) = \sum_{e \in F} x_e$, as well as $\text{supp}(x) = \{e \in E : x_e > 0\}$. We use x_e and $x(e)$ interchangeably. For a subset $F \subseteq E$ we define $\mathbb{1}_F$ to be the indicator vector with 1 on coordinates in F and 0 elsewhere. We shorten $\mathbb{1}_{\{e\}}$ to $\mathbb{1}_e$ for $e \in E$.

Some additional notation that is used only in Part I is introduced in Section 3.2.

2.2 Approximation algorithms

Let us consider minimization problems. We say that an algorithm is an α-approximation algorithm if it runs in polynomial time and, for every instance of the problem, the cost of the solution returned by the algorithm is at most α times the cost of an optimum solution. Here, the quantity α is called an approximation ratio or approximation factor. It could be either a constant or a function of the instance, usually of the size n of the instance.

For any minimization problem that is NP-hard (or at least not known to be in P), there arises the natural question: what is the smallest achievable constant approximation ratio α (or, failing that, the slowest-growing function $\alpha(n)$)?

Part I of this thesis is concerned with approximation algorithms for the asymmetric version of the traveling salesman problem, which we introduce in the following section.

2.3 The traveling salesman problem

The traveling salesman problem (TSP) is a classic NP-hard optimization problem. Informally, it is the task of finding the shortest tour visiting n given cities.

One popular formulation of the TSP is that we are given pairwise distances between the cities that are arbitrary (think of an arbitrary complete weighted graph), and wish to find the shortest tour that visits each city *exactly* once. Unfortunately, this formulation, apart from being arguably unnatural, is also NP-hard to approximate within any reasonable ratio, as doing so would essentially require solving the Hamiltonian cycle problem.

Fact 2.1

It is NP-hard to approximate the visit-exactly-once, arbitrary-weights version of the TSP within a factor 2^{n^5}.

Proof.
Let us think of undirected graphs. We will show a reduction from the Hamiltonian cycle problem, which is NP-hard. Consider an instance $G = (V, E)$; we should output YES if G contains a Hamiltonian cycle, and NO otherwise. We build an instance $G' = (V, E', w)$ of TSP as follows: let E' be such that (V, E') forms a complete graph and set weights $w(e) = 1$ for $e \in E$ and $w(e) = n \cdot 2^{n^5}$ for $e \in E' \setminus E$. In the YES-case there is a tour of length n, and thus a 2^{n^5}-approximation algorithm would return a tour of length at most $n \cdot 2^{n^5}$, but in the NO-case every tour has length at least $n \cdot 2^{n^5} + n - 1$. Therefore such an algorithm could differentiate between the two cases. Moreover, it would run in polynomial time, as the size of G' is bounded by a polynomial in the size of G. □

For this reason, we understand TSP to be the problem where every vertex should be visited *at least* once. (We also need the assumption that the input graph is (strongly) connected.) Equivalently, we can assume that the graph is complete and the weight function satisfies the triangle inequality ($w(v_1, v_3) \leq w(v_1, v_2) + w(v_2, v_3)$ for all $v_1, v_2, v_3 \in V$). In other words, w gives rise to a metric (this setting is sometimes called *metric TSP*).

The above two problem formulations are equivalent. This is because, given an instance of the at-least-once version, we can take the metric completion (that is, induce a complete graph where pairwise distances correspond to shortest-path distances in the original graph). Every tour in the original instance can be mapped to a tour of no larger cost in the metric completion

by *shortcutting*: we retrace the tour, skipping any vertex that would be revisited, and instead fast-forwarding to the first yet-unvisited vertex. Conversely, any tour in the metric completion can be mapped to a tour of the same cost in the original instance by replacing every edge (u, v) by a shortest path from u to v. Finally, if an instance already satisfies the triangle inequality, then an at-least-once tour in it can be mapped to an exactly-once tour of no larger cost by shortcutting.

We prefer the at-least-once formulation, as we find it more natural (for instance, we can talk about the special case of unweighted graphs rather than shortest-path metrics arising from unweighted graphs) and also because one of our reductions (Theorem 3.4) will yield a graph where not all edges are retained.

Moreover, we think of a tour as a multiset of edges: that is, we disregard their (cyclic) order in the tour. To be associated with an ordered tour, it is necessary and sufficient that an edge multiset F be Eulerian and connected. In the symmetric setting, being Eulerian means that the degree of every vertex (with respect to F) is even; in the asymmetric setting, the indegree (number of incoming edges) should be equal to the outdegree (number of outgoing edges). Here, by "connected" we mean that (V, F) is a connected graph (with no isolated vertices). In general, given an Eulerian multiset F that is not necessarily connected, we will call the edge sets of connected components of (V, F) *subtours* (see Definition 3.2). The set F itself will often be referred to as a collection of subtours. Further, once we have a connected Eulerian edge multiset F, we can turn it into an ordered tour using a simple linear-time algorithm.

As in this thesis we are interested in the asymmetric version of the problem, below we give the definition in this setting.

Definition 2.2 (*Asymmetric Traveling Salesman Problem – ATSP*)

Given: *an edge-weighted directed graph $G = (V, E, w)$ that is strongly connected.*
Output: *a minimum-weight connected and Eulerian edge multiset F.*

2.4 The perfect matching problem

Another classic task that this thesis is concerned with is the matching problem. Here, we are given an undirected unweighted graph $G = (V, E)$; a *matching* is a set $M \subseteq E$ of edges such that no two edges share an endpoint. Clearly we must have $|M| \leq n/2$; if $|M| = n/2$, i.e., every vertex is an endpoint of an edge in M, we say that M is a perfect matching.

Definition 2.3 (*Perfect Matching Problem*)

Given: *an undirected graph $G = (V, E)$.*
Output: *YES if G has a perfect matching, NO otherwise.*

The perfect matching problem is a decision problem. It has an associated *search version*: given a graph, *find* a perfect matching if one exists.

There are also several classic optimization versions of the problem: given a graph, what is the largest matching it contains (maximum-cardinality matching)? If edges are assigned weights, what is the minimum-weight perfect matching? For brevity, we do not address the optimization versions in this thesis; however, our results in Part II extend to those settings rather easily (as long as weights are polynomially bounded). Also, the approach of Part II will in fact heavily rely on finding (unique) minimum-weight perfect matchings with respect to a carefully[1] chosen

[1] Or, in fact, completely randomly!

edge-weight function.

The perfect matching problem is well-known to have efficient sequential algorithms. In this context, the case of *bipartite* graphs is much easier. In fact, as we have learned in 2006, the first polynomial-time algorithm for matching on bipartite graphs was given already in the 19th century by Jacobi (and published posthumously in Latin [Jac90]). Indeed, in his works on differential equations, he even gave an efficient algorithm for the weighted version of the problem (now known as the Hungarian method). For general graphs, the first polynomial-time algorithm was given by Edmonds [Edm65b]. This development is historically related to the idea that "efficient" means "polynomial-time", and the definition of the class P.

2.5 Linear programming

Linear programming is a mathematical optimization task where we seek to optimize a linear objective function over a polyhedron – i.e., subject to linear inequality constraints. A general form for a linear program can be given as

$$\min\{w^\top x : Ax \leq b, x \geq 0\}\,.$$

Here $x = (x_1, ..., x_n) \in \mathbb{R}^n$ is a vector of variables, $w \in \mathbb{R}^n$ is an objective function, and $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ describe the constraints, which are of the form $a_{j1}x_1 + ... + a_{jn}x_n \leq b_j$ for $j = 1, ..., m$. Throughout this section we focus on minimization problems.

The subset

$$P = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$$

is called the *feasible region* of the LP. It is a *polyhedron*; if it is also bounded as a subset of $\mathbb{R}^E$, we call it a *polytope*. We say that the LP is *feasible* if the polyhedron is nonempty; otherwise it is *infeasible*. If the objective function attains arbitrarily small values over the polyhedron, then we call it *unbounded*. If the LP is feasible but not unbounded, then it attains its minimum value.

Given a constraint of the form $A_i x \leq b_i$, where A_i is the i-th row of A, if a point $x \in P$ satisfies $A_i x = b_i$, then we say that this constraint is *tight* for x.

Solving LPs One reason why LPs are so useful in the design of algorithms is that they can be efficiently solved. From a theoretical perspective, the most helpful guarantees are given by the *ellipsoid method* [Kha79], which yields a (weakly) polynomial-time algorithm for linear programming. In fact, this algorithm can even be used to solve linear programs with an exponential number of constraints (which could not be written down in polynomial time). It only requires access to a *separation oracle*: a black-box routine that, given a point $x \in \mathbb{R}^n$, answers YES if x is in the feasible region of the LP, and otherwise returns a constraint that is violated by x.

Relaxations As mentioned in the introduction, the design of approximation algorithms requires a good lower bound on the value of an optimal solution to the considered problem – usually one that can be computed efficiently (such as a minimum spanning tree in the case of the 2-approximation algorithm for symmetric TSP). The efficient solvability of LPs means that, since 1979, they have become a crucial source of such lower bounds. Namely, one formulates the problem as an equivalent integer program (that is still NP-hard to solve), and then relaxes the integrality constraints to obtain a linear program. This linear program is called a *relaxation*, and

the integer points in the associated polyhedron correspond exactly to feasible solutions of the original problem.

Integrality gaps Of course, such a relaxation is not very useful if the lower bound that it yields is too loose. This gives rise to the concept of *integrality gap*, which is defined as the ratio between the values of the true integral optimum and of the relaxation, maximized over all problem instances. The smaller the integrality gap, the tighter and more useful the lower bound that it provides. The study of integrality gaps is worthwhile for two main reasons. First, if we know that the integrality gap for an efficiently solvable (e.g., polynomial-size) relaxation is at most ρ, then an algorithm that just solves this LP will return a value that is at most ρ times off from the true optimum. Such an algorithm is sometimes called a *ρ-estimation* algorithm, as it is non-constructive: it does not necessarily return an integral solution of value at most ρ times the returned LP lower bound. Second, the knowledge that a relaxation has an integrality gap of *more than* ρ rules out the possibility of obtaining a ρ-approximation algorithm whose guarantee is provable by comparing the value of the returned solution to ρ times the LP lower-bound, as there will exist an instance for which there is indeed no integral solution of such value.

The Held-Karp relaxation In this thesis we will prove a constant upper bound (Theorem 6.9) on the integrality gap of the standard relaxation associated with the Asymmetric Traveling Salesman Problem, which is called the Held-Karp relaxation. We discuss it below.

Recall from Section 2.3 that in ATSP we wish to find a minimum-weight connected and Eulerian edge multiset F. It is therefore natural that the relaxation should use a variable x_e for the number of times an edge $e \in E$ appears in this multiset (i.e., the tour). The Eulerian constraint is easy to express: we should have $x(\delta^-(v)) = x(\delta^+(v))$ for every $v \in V$ (recall that $\delta^-(v)$ and $\delta^+(v)$ are the sets of incoming and outgoing edges of v). We will call a nonnegative vector x satisfying these constraints a *circulation*.

Now, it would also make sense to require that $x(\delta^+(v)) \geq 1$ for every vertex $v \in V$, but this in itself would not be enough yet: namely, the resulting solution might consist of multiple connected components, which we call subtours. If S were the vertex set of such a subtour, then we would have $x(\delta^+(S)) = 0$. Therefore, to prevent the incidence of subtours, we introduce *subtour elimination constraints*: $x(\delta^+(S)) \geq 1$ for all proper subsets $S \subsetneq V$. We prefer to write these constraints in the equivalent, more symmetric form $x(\delta(S)) \geq 2$ (recall that $\delta(S) = \delta^+(S) \cup \delta^-(S)$). The linear programming relaxation $\mathrm{LP}(G, w)$ is therefore defined as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} w(e)x(e) \\ \text{subject to} \quad & x(\delta^+(v)) = x(\delta^-(v)) \qquad \text{for } v \in V, \\ & x(\delta(S)) \geq 2 \qquad \text{for } \emptyset \neq S \subsetneq V, \\ & x \geq 0. \end{aligned} \tag{$\mathrm{LP}(G, w)$}$$

The optimum value of this LP is called the *Held-Karp lower bound*.

We remark that the Held-Karp relaxation has exponentially many constraints. Nevertheless, we can solve it in polynomial time using the ellipsoid method with a separation oracle, which needs to solve the following task: given a vector $x \in \mathbb{R}^E$, either state that x is feasible, or return a constraint that x violates. The non-negativity constraints and Eulerian constraints are easy to efficiently verify. For the subtour elimination constraints, the problem amounts to

finding a minimum-weight cut in the graph (where edge weights are given by x), which can be done in polynomial time; cuts of weight less than 2 correspond to violated subtour elimination constraints.

The perfect matching polytope In terms of describing solutions using LPs, an ideal scenario is when the integrality gap is 1. In this case we say that the polyhedron is *integral*, and we usually do not call it a relaxation, as it is indeed an exact formulation for the original problem. If such an efficiently-solvable LP exists, then the problem is in P.

In this thesis we study the perfect matching problem, for which such an LP exists. Given a graph, let us associate an indicator vector $\mathbb{1}_M \in \{0,1\}^E \subseteq \mathbb{R}^E$ with any perfect matching M. (Sometimes for ease of notation we identify matchings M with their indicator vectors $\mathbb{1}_M$.) Then, the *perfect matching polytope* is defined as the convex hull of indicator vectors of all perfect matchings in the graph.

Now let us try to describe this polytope using linear inequalities, i.e., as an intersection of hyperplanes. Surely, a point $x \in \mathbb{R}^E$ in the polytope must satisfy that the degree of every vertex in the matching is 1: $x(\delta(v)) = 1$ for all $v \in V$. It turns out that this is already enough to describe the perfect matching polytope in the case of bipartite graphs. However, in general graphs, odd cycles pose an issue. Indeed, consider a triangle graph; it clearly does not have a perfect matching (since the number of vertices is odd), so its perfect matching polytope is empty, yet the above formulation would allow a point x with $x_e = 1/2$ for every $e \in E$. Therefore it is natural to add the following constraints: for any *odd* set S of vertices we should have $x(\delta(S)) \geq 1$. These constraints are clearly satisfied by the indicator vector of any perfect matching.

$$\begin{aligned} x(\delta(v)) &= 1 \quad \text{for } v \in V, \\ x(\delta(S)) &\geq 1 \quad \text{for } S \subseteq V \text{ with } |S| \text{ odd}, \\ x_e &\geq 0 \quad \text{for } e \in E. \end{aligned} \tag{PM(G)}$$

It turns out that this is enough:

Theorem 2.4 (*Edmonds [Edm65a]*)

The set of constraints PM(G) *describes the perfect matching polytope.*

Note that the constraints imply that $x_e \leq 1$ for any $e \in E$. We refer to the perfect matching polytope of the graph G by PM(G) or simply by PM when the graph is clear from the context.

To obtain a linear program, we minimize a linear objective function $\sum_{e \in E} w(e)x_e$ over this polytope. We denote this LP by PM(G, w). Like the Held-Karp relaxation, it has an exponential number of constraints. However, one can optimize over PM in polynomial time. The first reason for this is simply the existence of a polynomial-time algorithm for minimum-weight perfect matching [Edm65a]. Alternatively, one can also solve the LP using the ellipsoid method. Then, the task of the separation oracle can be reduced to finding a minimum-weight *odd* cut in a graph. This can be accomplished using Gomory-Hu trees [PR82].

Interestingly, there is in fact *no* polynomial-sized formulation for the perfect matching polytope, even in the *extension complexity* model, where extra variables are allowed in the formulation [Rot17].

Duality Duality is a foundational concept in the theory of linear programming. For us, it will be instrumental in extracting structure from the LP solution of the Held-Karp relaxation for ATSP.

Consider a linear program of the form

$$\min\{w^\top x : x \in \mathbb{R}^E, x \geq 0, Ax \geq b\}$$

with $A \in \mathbb{R}^{m \times E}$ and $b \in \mathbb{R}^m$. Suppose we wanted to provide a lower bound on the optimum value. What we could do is add several constraints $A_i x \geq b_i$ (where A_i is the i-th row of A), perhaps multiplied by coefficients $y_i \geq 0$, in such a way that we obtain a left-hand side $\sum_i y_i A_i x$ where every variable x_e appears with coefficient at most w_e, i.e., $\sum_i y_i A_{ie} \leq w_e$. Then the right-hand side $\sum_i y_i b_i$ would constitute a lower bound on the optimum value, since

$$\sum_e w_e x_e \geq \sum_e \sum_i y_i A_{ie} x_e = \sum_i y_i A_i x \geq \sum_i y_i b_i \,. \tag{2.1}$$

The task of finding the best such lower bound is called the *dual linear program*, and can be expressed as

$$\max\{b^\top y : y \in \mathbb{R}^m, y \geq 0, A^\top y \leq w\} \,.$$

Crucially, it turns out that adding constraints multiplied by coefficients y_i in this way is the only necessary way of obtaining such lower bounds, in the sense that some setting of the y's will always yield the best possible lower bound. This statement is known as *strong LP duality*:

Theorem 2.5 (*see e.g. [Sch03, Theorem 5.4]*)

Either both the primal and the dual program are infeasible, or one is feasible and unbounded and the other is infeasible, or both are feasible. In the last case, their values are equal.

Note that, given optimal solutions x and y to the primal and the dual, there must be equalities everywhere in (2.1). This implies that whenever $x_e > 0$, the corresponding dual constraint is tight: $\sum_i y_i A_{ie} = w_e$, and likewise whenever $y_i > 0$, the corresponding primal constraint is tight: $A_i x = b_i$. The converse is also true. More precisely, we have the following useful fact:

Fact 2.6 (*complementarity slackness*)

Let x and y be feasible solutions to the primal and the dual. Then x and y are both optimal if and only if, whenever $x_e > 0$ or $y_i > 0$, the corresponding constraints (as above) are tight.

Let us now write the dual program $\mathrm{DUAL}(G, w)$ and the complementarity slackness conditions for the Held-Karp relaxation $\mathrm{LP}(G, w)$ for ATSP. We associate variables $(\alpha_v)_{v \in V}$ and $(y_S)_{\emptyset \neq S \subsetneq V}$ with the first and second set of constraints of $\mathrm{LP}(G, w)$, respectively:

$$\begin{aligned} \text{maximize} \quad & \sum_{\emptyset \neq S \subsetneq V} 2 \cdot y_S \\ \text{subject to} \quad & \sum_{S:\ (u,v) \in \delta(S)} y_S + \alpha_u - \alpha_v \leq w(u,v) \qquad \text{for } (u,v) \in E, \\ & y \geq 0. \end{aligned} \tag{DUAL(G,w)}$$

By Fact 2.6 we get the following:

Fact 2.7

If x and (α, y) are feasible solutions to $\mathrm{LP}(G, w)$ *and* $\mathrm{DUAL}(G, w)$ *respectively, then they are both optimal if and only if we have*

$$\sum_{S:\,(u,v)\in\delta(S)} y_S + \alpha_u - \alpha_v = w(u,v) \quad \text{whenever } x_{(u,v)} > 0$$

and

$$x(\delta(S)) = 2 \quad \text{whenever } y_S > 0\,.$$

Definition 2.8 (*tight sets, ATSP*)

We say that a set $S \subseteq V$ is tight *if $x(\delta(S)) = 2$ (that is, $x(\delta^+(S)) = x(\delta^-(S)) = 1$).*

For singleton sets $\{u\}$, we will use the notation $y_u = y_{\{u\}}$.

Finally, let us remark that $\mathrm{DUAL}(G, w)$ can also be solved in polynomial time, despite having an exponential number of variables. To see this, note that if we solve $\mathrm{LP}(G, w)$ using the ellipsoid method, then the separation oracle is going to return a polynomial number of violated constraints (cutting planes) during its execution. Thus, $\mathrm{LP}(G, w)$ with only those constraints has the same optimum value, and in $\mathrm{DUAL}(G, w)$ we only need to use the dual variables corresponding to those constraints; more precisely, we obtain a polynomial-size family $\mathcal{S} \subseteq 2^V$ such that $\mathrm{DUAL}(G, w)$ has an optimal solution (α, y) with support $\{S \subseteq V : y_S > 0\}$ contained in $\mathcal{S}$. In the subsequent Section 2.6 we will show that we can efficiently obtain a dual optimal solution with a very structured support. Later, in Theorem 3.4 we show that we can also dispose of the variables α_v.

Faces For our result on matchings, we will crucially use the structure of *faces* of the perfect matching polytope. Let us define this notion.

Definition 2.9

A subset F of a polyhedron $P = \{x \in \mathbb{R}^E : Ax \leq b\}$ is called a face *of P if it is the set of optimum solutions of $\min\{w^\top x : x \in P\}$ for some $w \in \mathbb{R}^E$.*

We will also need the following equivalent characterization:

Fact 2.10 (*see [KV12, Proposition 3.3]*)

A nonempty subset F of a polyhedron $P = \{x \in \mathbb{R}^E : Ax \leq b\}$ is a face of P if $F = \{x \in P : A'x = b'\}$ for some subsystem $A'x \leq b'$ of $Ax \leq b$.

That is, a face can be obtained either by minimizing an objective function over the polyhedron, or by setting some of the inequality constraints in the polyhedral description to equality. Note that a face is always nonempty.

We say that a constraint is tight for a face F if it is tight (i.e., holds with equality) for every point $x \in F$. If a constraint $x(\delta(S)) = 1$ is tight, then we say that S is tight (it is a tight odd set). Notice that if a set is tight for a face, then it is also tight for any of its subfaces.

Recall PM – the formulation of the perfect matching polytope. In fact, all faces considered in this thesis will be faces of PM. By Fact 2.10, any face $F \subseteq \mathrm{PM}$ can be described using a set of inequality constraints that are tight for F. In the case of PM, such constraints correspond to edges or to odd sets of vertices. We use the following notation:

Definition 2.11

For a face $F \subseteq \mathrm{PM}$ we define

$$E(F) = \{e \in E : (\exists x \in F)\ x_e > 0\}$$

and

$$\mathcal{S}(F) = \{S \subseteq V : |S| \text{ odd and } (\forall x \in F)\ x(\delta(S)) = 1\}\,.$$

In other words, $E(F)$ contains the edges that appear in a perfect matching in F and $\mathcal{S}(F)$ contains the tight cut constraints of F.

For a face $F \subseteq \mathrm{PM}$, by Fact 2.10 we have

$$F = \{x \in \mathrm{PM} : x_e = 0 \text{ for all } e \in E \setminus E(F) \text{ and } x(\delta(S)) = 1 \text{ for all } S \in \mathcal{S}(F)\}\,.$$

See also Lemma 2.16.
We remark that any face of an integral polytope such as PM is again an integral polytope.

2.6 Laminarity and uncrossing

In both of our results we crucially utilize a laminar structure that arises from the LP formulations. In this section we explain the notion of laminarity and discuss how to obtain this structure.

Definition 2.12

Consider subsets A, B of a universe V. We say that A and B cross *if all of $A \setminus B$, $B \setminus A$, $A \cap B$ are nonempty.*

Note that for crossing A, B it might be the case that $A \cup B = V$.

Definition 2.13

A family $\mathcal{L} \subseteq 2^V$ of subsets of V is called laminar *if no two sets $A, B \in \mathcal{L}$ cross.*

In other words, for any $A, B \in \mathcal{L}$ we have either $A \subseteq B$, $B \subseteq A$, or $A \cap B = \emptyset$. Thus a laminar family is a tree-like structure of sets. See Figure 8.4 on page 95 for an example.

Uncrossing is a general technique of obtaining laminar families from arbitrary set families in certain scenarios. This is done by repeatedly taking crossing sets A, B and replacing one of them[2] with two new sets: either $A \cup B$, $A \cap B$ or $A \setminus B$, $B \setminus A$. Note that the obtained three sets are not crossing.

The following basic fact helps explain why such a replacement might be desirable:

Fact 2.14 (*submodularity of the cut function*)

Let $A, B \subseteq V$. We have

$$\begin{aligned}\mathbb{1}_{\delta(A)} + \mathbb{1}_{\delta(B)} &= \mathbb{1}_{\delta(A\cap B)} + \mathbb{1}_{\delta(A\cup B)} + 2 \cdot \mathbb{1}_{\delta(A\setminus B, B\setminus A)} \\ &= \mathbb{1}_{\delta(A\setminus B)} + \mathbb{1}_{\delta(B\setminus A)} + 2 \cdot \mathbb{1}_{\delta(A\cap B, V\setminus(A\cup B))}\,.\end{aligned}$$

[2] We generally do not have control over which one is replaced.

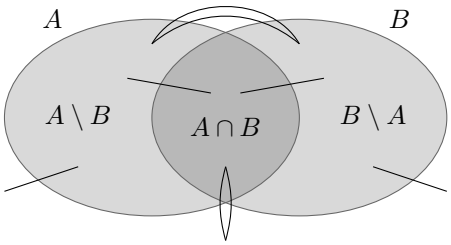


Figure 2.1: Illustration of the proof of Fact 2.14.

Proof.
Consider the following six disjoint sets of edges: $\delta(A \cap B, V \setminus (A \cup B))$, $\delta(A \setminus B, V \setminus (A \cup B))$, $\delta(B \setminus A, V \setminus (A \cup B))$, $\delta(A \setminus B, A \cap B)$, $\delta(B \setminus A, A \cap B)$, $\delta(A \setminus B, B \setminus A)$. Entries corresponding to edges that are not in any of these sets are zero on all three sides of the equation. One can verify that entries corresponding to $\delta(A \cap B, V \setminus (A \cup B))$ and $\delta(A \setminus B, B \setminus A)$ are 2 on all sides, and those corresponding to any of the other four sets are 1. See Figure 2.1 for an illustration. □

We demonstrate the power and utility of uncrossing on two examples, which pertain to the two parts of this thesis.

Matching For the perfect matching problem, we consider any face F of the primal polytope PM. As remarked above (see Definition 2.11), F can be described using a subset of edges and a family of tight odd sets. We show that this family can be chosen to be laminar. More precisely, we want to prove the following lemma.

Lemma 2.15

Consider a face F. For any maximal laminar subset $\mathcal{L}$ of $\mathcal{S}(F)$ we have

$$\operatorname{span}(\mathcal{L}) = \operatorname{span}(\mathcal{S}(F)),$$

where for a subset $\mathcal{T} \subseteq \mathcal{S}(F)$, $\operatorname{span}(\mathcal{T})$ denotes the linear subspace of $\mathbb{R}^E$ spanned by the boundaries of sets in $\mathcal{T}$, i.e., $\operatorname{span}(\mathcal{T}) = \operatorname{span}\{\mathbb{1}_{\delta(S)} : S \in \mathcal{T}\}$.

We say that $\mathcal{L}$ is a *maximal laminar subset* of a family $\mathcal{S}$ if no set in $\mathcal{S} \setminus \mathcal{L}$ can be added to $\mathcal{L}$ while maintaining laminarity. Lemma 2.15 implies that a maximal laminar subfamily $\mathcal{L} \subseteq \mathcal{S}(F)$ is enough to describe a face F (together with the edge set $E(F)$):

Lemma 2.16

Let F be a face of PM *and $\mathcal{L}$ be a maximal laminar subset of $\mathcal{S}(F)$. Then*

$$F = \{x \in \mathrm{PM} : x_e = 0 \text{ for all } e \in E \setminus E(F) \text{ and } x(\delta(S)) = 1 \text{ for all } S \in \mathcal{L}\}.$$

Proof.
The direction $\subseteq$ is clear since $\mathcal{L} \subseteq \mathcal{S}(F)$. Now consider any $S \in \mathcal{S}(F)$. By Lemma 2.15, we can write $\mathbb{1}_{\delta(S)} = \sum_{L \in \mathcal{L}} \mu_L \mathbb{1}_{\delta(L)}$ for some coefficients $\mu \in \mathbb{R}^{\mathcal{L}}$. Moreover, we have $\sum_{L \in \mathcal{L}} \mu_L = 1$,

since, taking an arbitrary $y \in F$, we can write

$$1 = \langle y, \mathbb{1}_{\delta(S)} \rangle = \left\langle y, \sum_{L \in \mathcal{L}} \mu_L \delta_{\mathbb{1}(L)} \right\rangle = \sum_{L \in \mathcal{L}} \mu_L \,.$$

For the direction $\supseteq$, we have that for any x from the right-hand side and for any $S \in \mathcal{S}(F)$,

$$\langle x, \mathbb{1}_{\delta(S)} \rangle = \left\langle x, \sum_{L \in \mathcal{L}} \mu_L \mathbb{1}_{\delta(L)} \right\rangle = \sum_{L \in \mathcal{L}} \mu_L \cdot x(\delta(L)) = 1 \,.$$

□

Note that any single-vertex set is tight for any face (due to the constraints $x(\delta(v)) = 1$ for all $v \in V$), and therefore a maximal laminar family contains all singletons. In Part II, all considered laminar families will always contain all singletons. Furthermore, given a subface $F' \subseteq F$, we can extend $\mathcal{L}$ to a larger laminar family $\mathcal{L}' \supseteq \mathcal{L}$ that describes F'.

The following proof of Lemma 2.15 is adapted from [LRS11]. Assume without loss of generality that $E = E(F)$. We can do this since including the constraint $x_e = 0$ yields the same face as removing the edge e from G.

Since we rely on the structure of *odd* sets, the new sets that we add must also be odd. Thus the choice of $A \cup B$, $A \cap B$ or $A \setminus B$, $B \setminus A$ simply depends on the parity of $|A \cap B|$. The following fact shows that the new sets are still tight for the considered face.

Lemma 2.17

Let $A, B \in \mathcal{S}(F)$ be two tight sets that are crossing. Then:

- *if $|A \cap B|$ is odd: then $A \cap B, A \cup B \in \mathcal{S}(F)$ and $\mathbb{1}_{\delta(A)} + \mathbb{1}_{\delta(B)} = \mathbb{1}_{\delta(A \cap B)} + \mathbb{1}_{\delta(A \cup B)}$,*
- *otherwise: $A \setminus B, B \setminus A \in \mathcal{S}(F)$ and $\mathbb{1}_{\delta(A)} + \mathbb{1}_{\delta(B)} = \mathbb{1}_{\delta(A \setminus B)} + \mathbb{1}_{\delta(B \setminus A)}$.*

Proof.
Case $|A \cap B|$ odd: Take any $x \in F$. Consider the first equality in Fact 2.14, and take the dot product of both sides with x. Since $A, B \in \mathcal{S}(F)$ and because $A \cap B$, $A \cup B$ are nonempty (being odd), we have

$$1 + 1 = x(\delta(A)) + x(\delta(B)) = x(\delta(A \cap B)) + x(\delta(A \cup B)) + 2 \cdot x(\delta(A \setminus B, B \setminus A)) \geq 1 + 1 + 2 \cdot 0$$

where the inequality must be an equality, and thus $x(\delta(A \cap B)) = 1$, $x(\delta(A \cup B)) = 1$ (implying $A \cap B, A \cup B \in \mathcal{S}(F)$) and $x(\delta(A \setminus B, B \setminus A)) = 0$ for all $x \in F$ (which, given that $E = E(F)$, implies that $\delta(A \setminus B, B \setminus A) = \emptyset$ and thus $\mathbb{1}_{\delta(A \setminus B, B \setminus A)} = 0$).
Case $|A \cap B|$ even: analogous. □

Proof of Lemma 2.15.
We wish to show that $\operatorname{span}(\mathcal{L}) = \operatorname{span}(\mathcal{S}(F))$, i.e., that for all $A \in \mathcal{S}(F)$ we have $\mathbb{1}_{\delta(A)} \in \operatorname{span}(\mathcal{L})$. We prove this by induction on the number of sets in $\mathcal{L}$ that cross A. Let us call this quantity $\operatorname{cross}_{\mathcal{L}}(A)$.

The case $\text{cross}_{\mathcal{L}}(A) = 0$ is easy: we must have $A \in \mathcal{L}$ (then clearly $\mathbb{1}_{\delta(A)} \in \text{span}(\mathcal{L})$), or else $\mathcal{L} \cup \{A\}$ would be a larger laminar family, contradicting the maximality of $\mathcal{L}$. So let $\text{cross}_{\mathcal{L}}(A) \geq 1$, $A \notin \mathcal{L}$, and let $B \in \mathcal{L}$ be some set crossing A.

Claim 1

All four numbers $\text{cross}_{\mathcal{L}}(A \cap B)$, $\text{cross}_{\mathcal{L}}(A \cup B)$, $\text{cross}_{\mathcal{L}}(A \setminus B)$ *and* $\text{cross}_{\mathcal{L}}(B \setminus A)$ *are smaller than* $\text{cross}_{\mathcal{L}}(A)$.

Proof.
Let $X \in \{A \cap B, A \cup B, A \setminus B, B \setminus A\}$. It is easy to check that if $S \in \mathcal{L}$ is a set that crosses X, then S also crosses A. For the strict inequality, note that $B \in \mathcal{L}$ itself is a set that crosses A, but does not cross X. □

Assume that $|A \cap B|$ is odd; the other case is analogous. Then by Lemma 2.17, $A \cap B, A \cup B \in \mathcal{S}(F)$ and

$$\mathbb{1}_{\delta(A)} + \mathbb{1}_{\delta(B)} = \mathbb{1}_{\delta(A \cap B)} + \mathbb{1}_{\delta(A \cup B)}. \tag{2.2}$$

By Claim 1 and the inductive hypothesis we have $\mathbb{1}_{\delta(A \cap B)}, \mathbb{1}_{\delta(A \cup B)} \in \text{span}(\mathcal{L})$, and of course also $\mathbb{1}_{\delta(B)} \in \text{span}(\mathcal{L})$. This and (2.2) implies that $\mathbb{1}_{\delta(A)} \in \text{span}(\mathcal{L})$. □

Note that here, uncrossing was performed only as an argument inside the proof (indeed, our algorithm in Part II is quite simple and the difficulty lies in its analysis).

Traveling salesman problem For ATSP, we consider optimal dual solutions. We show that we can obtain such a solution y whose support $\{S \subseteq V : y_S > 0\}$ is laminar. Apart from uncrossing the dual rather than the primal program, here the operation is also being performed algorithmically.

Lemma 2.18

For every edge-weighted directed graph (G, w) *there exists an optimal solution* (α, y) *to* $\text{DUAL}(G, w)$ *such that the support of* y *is a laminar family of vertex sets. Moreover, such a solution can be computed in polynomial time.*

Proof.
We start by showing the existence of a laminar optimal solution. Select (α, y) to be an optimal solution to $\text{DUAL}(G, w)$ minimizing $\sum_S |S| y_S$. That is, among all dual solutions that maximize the dual objective $2 \sum_S y_S$, we select one that minimizes $\sum_S |S| y_S$. We claim that the support $\mathcal{L} = \{S : y_S > 0\}$ is a laminar family. Suppose not, i.e., that there are crossing sets $A, B \in \mathcal{L}$. Then we can obtain a new dual solution $(\alpha, \hat{y})$, where $\hat{y}$ is defined, for $\varepsilon = \min(y_A, y_B) > 0$, as

$$\hat{y}_S = \begin{cases} y_S - \varepsilon & \text{if } S = A \text{ or } S = B, \\ y_S + \varepsilon & \text{if } S = A \setminus B \text{ or } S = B \setminus A, \\ y_S & \text{otherwise.} \end{cases}$$

Note that this has the effect of removing A or B from the support, while adding both $A \setminus B$ and $B \setminus A$ (if they were not already present).

Let us verify that $(\alpha, \hat{y})$ remains a feasible solution. Clearly $\hat{y}$ remains nonnegative by the selection of ε. Now consider any constraint $\sum_{S\colon e\in\delta(S)} y_S \le w(e) - \alpha_u + \alpha_v$ for $e = (u, v) \in E$. We have

$$\sum_{S\colon\, e\in\delta(S)} \hat{y}_S = \sum_{S\colon\, e\in\delta(S)} y_S - \varepsilon \cdot \underbrace{\left(\mathbb{1}_{\delta(A)}(e) + \mathbb{1}_{\delta(B)}(e) - \mathbb{1}_{\delta(A\setminus B)}(e) - \mathbb{1}_{\delta(B\setminus A)}(e)\right)}_{\ge 0 \text{ by Fact 2.14}}$$

and thus the constraint remains satisfied for $\hat{y}$. Further, we clearly have $2\sum_S \hat{y}_S = 2\sum_S y_S$. In other words, $(\alpha, \hat{y})$ is an optimal dual solution. However,

$$\sum_S |S|(y_S - \hat{y}_S) = \varepsilon \cdot (|A| + |B| - |A \setminus B| - |B \setminus A|) > 0\,,$$

which contradicts the choice of (α, y) as an optimal dual solution minimizing $\sum_S |S| y_S$. Therefore, there can be no such sets A and B in $\mathcal{L}$, and so it is a laminar family.

To find a laminar optimal solution in polynomial time, we start with an arbitrary dual optimal solution. As noted above, one can be computed in polynomial time. Now we repeatedly apply the above uncrossing operation to obtain a laminar optimal solution. A result by Karzanov [Kar96, Theorem 2] shows that if we carefully select the sequence of pairs A, B to uncross, this can be performed in polynomial time (although for an arbitrary sequence, the number of uncrossing steps may not be polynomially bounded). Alternatively, one could add the constraint $2\sum_S y_S = \mathrm{OPT}$ to the dual, where OPT is the primal-dual optimum value, and replace the objective function by minimizing $\sum_S |S| y_S$ and solve the obtained dual program. □

We remark that for TSP, uncrossing was first used for a variant of the symmetric setting by Cornuejols et al. [CFN85]. Further, Vempala and Yannakakis [VY99] prove that any so-called basic solution to the Held-Karp relaxation for ATSP is sparse: it uses at most $3n-2$ edges. They use uncrossing to argue about the span of tight constraints in a manner similar to Lemma 2.15.

Part I

A Constant-Factor Approximation Algorithm for the Asymmetric Traveling Salesman Problem

Chapter 3

Asymmetric Traveling Salesman Problem

The traveling salesman problem (TSP) is one of the best-known NP-hard optimization problems. Recall from Section 2.3 that we are given an edge-weighted graph and wish to find the shortest tour, which we define as an Eulerian multiset of edges that connects the graph by visiting every vertex at least once.

For the symmetric TSP, where the graph is undirected, recall that in Chapter 1 we gave a 2-approximation algorithm: simply compute a minimum-cost spanning tree and take every edge two times. It is possible to do better: Christofides' classic algorithm [Chr76], also discovered independently by Serdyukov [Ser78], transforms the spanning tree into a tour more judiciously (by adding a minimum-cost matching) and yields a $3/2$-approximation. Improving this guarantee is a notorious open question in combinatorial optimization. There has been recent progress in the *graph-TSP* case, where the input graph is unweighted [GSS11, MS16, Muc12], and the current best approximation is 1.4 [SV14]. The integrality gap of the symmetric Held-Karp relaxation is known to be between $4/3$ and $3/2$ (the latter due to Wolsey's analysis [Wol80] of the Christofides-Serdyukov algorithm) and conjectured to be $4/3$.

Our understanding is much more limited in the asymmetric case (ATSP), where the graph is directed. No constant-factor approximation algorithm was known prior to the work described in this part of thesis. This is despite the integrality gap of the asymmetric Held-Karp relaxation (see Section 2.5) being conjectured to be close to the best known lower bound, which is 2 [CGK06]. In terms of hardness of approximation, it is NP-hard to do better than 75/74 [KLS13].

The first approximation algorithm for ATSP was due to Frieze, Galbiati and Maffioli [FGM82] and achieves a factor of $\log_2 n$ using a repeated cycle cover approach. Their approach was further refined over the years [Blä08, KLSS05, FS07]; all these algorithms have approximation ratios of the form $c \log_2 n$ for some $0 < c < 1$. The first asymptotic improvement was given by Asadpour et al. [AGM+10], who introduced a novel connection between ATSP and the concept of thin spanning trees. This connection was subsequently used to obtain a constant-factor approximation algorithm for planar (and more generally bounded-genus) graphs [GS11] and to prove a $\operatorname{poly} \log \log n$ upper bound on the integrality gap of the Held-Karp relaxation for ATSP [AG15].

In 2015, Svensson [Sve15] gave a constant-factor approximation algorithm for the case of unweighted graphs. More generally, this result holds for *node-weighted graphs*, i.e., graphs whose

weight function can be written as $w(u, v) = f(u) + f(v)$ for some $f : E \to \mathbb{R}_+$.[3]

In 2016, in joint work with Svensson and Végh, we generalized this result to graphs that have two different edge weights [STV18b]. In this part of the thesis, we build upon and generalize both of these results to give a constant-factor approximation algorithm for arbitrary weight functions.

Theorem 3.1

There is a polynomial-time algorithm for ATSP that returns a tour of value at most 506 times the Held-Karp lower bound.

This result is also joint work with Svensson and Végh and was first published in STOC 2018 with an approximation ratio of 5500 [STV18a]. In this thesis we present a subsequent, more optimized version.

We remark that we can obtain a tighter upper bound of 319 for the integrality gap of the Held-Karp relaxation, and that our results also imply a constant-factor approximation algorithm for the Asymmetric Traveling Salesman *Path* Problem via black-box reductions [FS07, KTV18] – see Section 6.2.

3.1 Our approach and outline of Part I

Svensson's result for node-weighted graphs [Sve15] was obtained via a new, relaxed variant of ATSP called Local-Connectivity ATSP. As all components of that result are useful for this thesis, we reprise and adapt them in Chapter 4. In particular, the Local-Connectivity ATSP problem has been adjusted to our setting to optimize the approximation guarantee and renamed *Subtour Partition Cover* – see Section 4.1. This relaxed problem is easily seen to be no harder than ATSP; the crux of Svensson's result is a reduction which shows that if one can solve Local-Connectivity ATSP on some class of graphs, then one can obtain a constant-factor approximation for ATSP on that class of graphs (cf. Theorem 4.3). Further, it turns out that it is rather straightforward to solve Local-Connectivity ATSP on node-weighted graphs (cf. Theorem 4.2). Via the aforementioned reduction, this yields a constant-factor approximation algorithm for ATSP on node-weighted graphs. With much additional work, one can also solve Local-Connectivity ATSP on graphs with two different edge weights [STV18b].

Still, it does not seem easy to solve Local-Connectivity ATSP / Subtour Partition Cover on graphs with general weights. Therefore we take a different approach. Before applying Svensson's reduction (Theorem 4.3), we remain in the realm of ATSP and use a series of natural reductions to gradually simplify the structure of instances that we are dealing with. The first of these reductions (in Section 3.3) crucially uses the laminar structure arising from the Held-Karp relaxation and its dual linear program (see Lemma 2.18 and Theorem 3.4). All further reductions are described in Chapter 5. The most structured instances, for which we apply the reduction to Subtour Partition Cover and on which we then solve Subtour Partition Cover, are called *vertebrate pairs*.

An outline of this part of the thesis is as follows. In Section 3.2 we introduce new notation relevant to Part I. Section 3.3 is devoted to our first reduction. There, we show that we can focus on *laminarly-weighted ATSP instances*: there is a laminar family $\mathcal{L}$ of vertex sets and a nonnegative vector $(y_S)_{S\in\mathcal{L}}$ such that any edge e has $w(e) = \sum_{S\in\mathcal{L}:\, e\in\delta(S)} y_S$. See the left part

[3] In [Sve15], the definition is slightly different: $w(u, v) = f(u)$ for every $(u, v) \in E$ for a function $f : V \to \mathbb{R}_+$. The two definitions are equivalent: by assigning $f(u) = 2h(u)$, the weight of any tour is equal for the weights $w(u, v) = f(u)$ and for the weights $w(u, v) = h(u) + h(v)$.

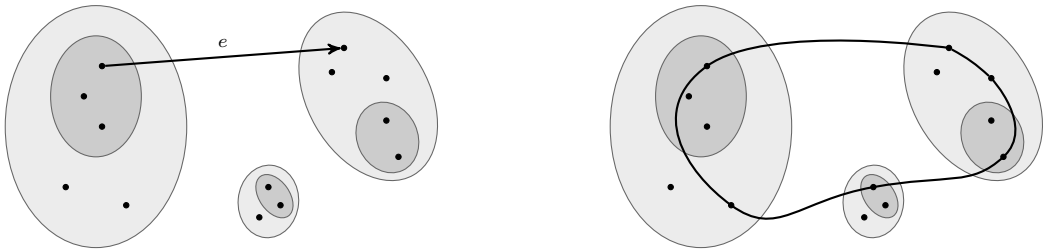


Figure 3.1: On the left we give an example of a laminarly-weighted ATSP instance. The sets of the laminar family are shown in gray. We depict a single edge e that crosses three sets in the laminar family, say S_1, S_2, S_3, and so $w(e) = y_{S_1} + y_{S_2} + y_{S_3}$. On the right, we give an example of a vertebrate pair. Notice that the backbone (depicted as the cycle) crosses all non-singleton sets of the laminar family, though it may not visit all the vertices.

of Figure 3.1 for an example. Note that the special case when the laminar family consists only of singletons roughly corresponds to node-weighted instances. We call laminarly-weighted ATSP instances where $\mathcal{L} \subseteq \{\{v\} : v \in V\}$ *singleton instances*.[4]

Next, in Chapter 4 we define the Subtour Partition Cover problem and reduce the task of solving ATSP (with a constant-factor approximation) to that problem. We also solve Subtour Partition Cover on singleton instances (Theorem 4.2), thus illustrating the power of the reduction as well as developing a tool necessary later in Chapter 5.

In Chapter 5 we turn our attention back to ATSP and, starting from laminarly-weighted instances, show that we can obtain very structured ATSP instances called vertebrate pairs by only increasing the approximation guarantee by a constant factor. Let us now discuss the main ideas of this chapter.

We begin by exploring the structure of sets in the laminar family $\mathcal{L}$: in Section 5.1 we study paths inside sets $S \in \mathcal{L}$ and, in Section 5.2, we introduce analogues of the classic graph-theoretic operations of contracting and inducing on such a set. These operations naturally give rise to a recursive algorithm that, intuitively, works as long as the contraction of some set $S \in \mathcal{L}$ results in a significant decrease in the value of the LP relaxation. In Section 5.3 we formally analyze this recursive algorithm and reduce the task of approximating ATSP to that of approximating ATSP on *irreducible* instances: those where no set $S \in \mathcal{L}$ brings about a significant decrease of the LP value if contracted.

Informally, every set $S \in \mathcal{L}$ in an irreducible instance has two vertices $u, v \in S$ such that the shortest path from u to v crosses a large (weighted) fraction of the sets $R \in \mathcal{L} : R \subsetneq S$ (otherwise contracting S into a single vertex, endowed with a node-weight equal to the weight of the shortest path, would lead to a decrease in the LP value). This insight, together with the approximation algorithm for singleton instances in Chapter 4 (Corollary 4.4), allows us to construct a low-weight subtour B that does not necessarily visit every vertex, but crosses every non-singleton set of $\mathcal{L}$. See the right part of Figure 3.1 for an example. We refer to B as a *backbone*, and to the ATSP instance and the backbone together as a *vertebrate pair*. This reduction allows us to further assume that our input is such a vertebrate pair; it is presented in Section 5.4.

In each of the above stages, we prove a theorem of the form: if there is a constant-factor approximation for ATSP on more structured instances, then there is a constant-factor approxi-

[4] For an explanation of the difference between node-weighted and singleton instances see the discussion following Definition 3.5.

mation for ATSP on less structured instances. For example, an algorithm for irreducible instances implies an algorithm for laminarly-weighted instances. One can also think of making a stronger and stronger assumption on the instance without loss of generality, making it increasingly resemble a singleton instance. The reduction to vertebrate pairs concludes Chapter 5.

Finally, in Chapter 6 we give an algorithm for Subtour Partition Cover on such instances. The main technical concept in the argument is that of *witness flows*. On a high level, we want every subtour T in our solution to Subtour Partition Cover to be forced to intersect the backbone B if T crosses a non-singleton set in the laminar family $\mathcal{L}$. Every subtour that does not cross any such set behaves exactly as in a singleton instance with regard to its cost, and it is easy to account for those subtours. On the other hand, we are able to take care of the cost of all subtours that do cross some such set (and thus also intersect B), together with B, using a global cost argument. The witness flow is a tool that allows us to enforce this crucial property in our solution to Subtour Partition Cover. It is inspired by a general method of ensuring connectivity in integer/linear programming formulations for graph problems, which requires the existence of a flow (supported on the LP solution) between the pairs of vertices that should be connected. The same role was played in our previous work on ATSP on graphs with two different edge weights [STV18b] and in our conference paper on the general ATSP result [STV18a] by a concept called the *split graph*.

By the reductions in Chapters 4 and 5, solving Subtour Partition Cover for vertebrate pairs, which we do in Chapter 6, is sufficient for obtaining a constant-factor approximation algorithm for general ATSP. We combine all the ingredients and calculate the obtained ratio in Section 6.2.

See the conclusions (Chapter 9) for a discussion of future research directions.

3.2 Notation

Recall the notation introduced in Section 2.1. In this section we introduce further notions that are needed for ATSP.

Throughout Part I, the subsets $F \subseteq E$ of edges we refer to will be multisets – that is, they may contain multiple copies of the same edge. Standard set operations are defined appropriately (for instance, union $\cup$ sums the multiplicities of every edge). The indicator vector $\mathbb{1}_F$ may take values larger than one.

For a vertex set $U \subsetneq V$, we let $G[U]$ denote the subgraph induced by U. That is, $G[U] = (U, E(U))$. We also let G/U denote the graph obtained by contracting the vertex set U, i.e., by replacing the vertices in U by a single new vertex u and redirecting every edge with one endpoint in U to the new vertex u. This may create parallel edges in G/U. We keep all parallel copies; thus, every edge in G/U will have a unique preimage in G.

For a set $S \subsetneq V$ we let S_{in} and S_{out} be those vertices of S that have an incoming edge from outside of S and those that have an outgoing edge to outside of S, respectively. That is,

$$S_{\text{in}} = \{v \in S : \delta^-(S) \cap \delta^-(v) \neq \emptyset\} \quad \text{and} \quad S_{\text{out}} = \{v \in S : \delta^+(S) \cap \delta^+(v) \neq \emptyset\}.$$

For $S, T \subseteq V$ and $F \subseteq E$, we use the shorthand notation $\delta_F^+(S) = F \cap \delta^+(S)$, $\delta_F^-(S) = F \cap \delta^-(S)$, and $\delta_F(S,T) = F \cap \delta(S,T)$.

For $F \subseteq E$, we let $V(F)$ denote the set of vertices incident to at least one edge in F.

Finally, a closed walk will be called a subtour:

Definition 3.2

We call $F \subseteq E$ a subtour if F is Eulerian (we have $|\delta_F^+(v)| = |\delta_F^-(v)|$ for every $v \in V$) and the graph $(V(F), F)$ is connected. By convention, $F = \emptyset$ is a subtour.

Therefore a subtour is an Eulerian multiset of edges that form a single connected component (or an empty set). A subtour F is a tour if it visits every vertex at least once, i.e., $V(F) = V$. In other words, a tour is an Eulerian multiset of edges that connects the graph.

For any Eulerian multiset F of edges, we refer to the connected components of $(V(F), F)$ as *subtours in* F. We often refer to F as a collection of subtours. Note that if T is a subtour in F, then $T \neq \emptyset$.

We say that a subtour T intersects another subtour T' if we have $V(T) \cap V(T') \neq \emptyset$.

3.3 Laminarly-weighted ATSP and singleton instances

In this section we show that without loss of generality (i.e., without any loss in the approximation factor) we can focus on instances whose weights come from a sparse and highly structured family of sets. Below we define the crucial notion of laminarly-weighted instances that we will work with throughout Part I.

Definition 3.3

A tuple $\mathcal{I} = (G, \mathcal{L}, x, y)$ is called a laminarly-weighted ATSP instance *if G is a strongly connected directed graph, $\mathcal{L}$ is a laminar family of vertex subsets, x is a feasible solution to the Held-Karp relaxation for G, and $y : \mathcal{L} \to \mathbb{R}_+$. We further require that $x_e > 0$ for every $e \in E$ and that every set $S \in \mathcal{L}$ be tight with respect to x, i.e., that $x(\delta^+(S)) = x(\delta^-(S)) = 1$. We define the* induced weight function $w_{\mathcal{I}} : E \to \mathbb{R}_+$ *as*

$$w_{\mathcal{I}}(e) = \sum_{S \in \mathcal{L}:\ e \in \delta(S)} y_S \qquad \text{for every } e \in E. \tag{3.1}$$

Given an instance $\mathcal{I}$ as in the definition, the vectors x and y have the following important property. Define a dual solution $(\bar{\alpha}, \bar{y})$ by setting $\bar{\alpha}_u = 0$ for all $u \in V$, and $\bar{y}_S = y_S$ if $S \in \mathcal{L}$ and $\bar{y}_S = 0$ otherwise. Then complementarity slackness (see Fact 2.7) implies that for the induced weight function $w_{\mathcal{I}}$, the vector x is an optimal solution to $\mathrm{LP}(G, w_{\mathcal{I}})$ and $(\bar{\alpha}, \bar{y})$ is an optimal solution to $\mathrm{DUAL}(G, w_{\mathcal{I}})$.

Now we prove the main insight of this section: that ATSP with arbitrary weights can be reduced to the laminarly-weighted ATSP problem. In fact, we have already done most of the work in Section 2.6 (see Lemma 2.18): the support of the dual solution can be assumed to be laminar. If we disregard edges e with $x_e = 0$, then complementarity slackness (Fact 2.7) almost implies the crucial property (3.1) of the weights that we want – it would do so if we had $\alpha = 0$. The extra step we need here (in the proof of Theorem 3.4) is to dispose of the vertex potential variables α_v from the dual. This is done by replacing the weight function with an equivalent one.

Theorem 3.4

Assume we have a polynomial-time algorithm that finds a solution of weight at most α times the Held-Karp lower bound for every laminarly-weighted ATSP instance. Then there is a polynomial-time algorithm for the general ATSP problem that finds a solution of weight at most α times the Held-Karp lower bound.

Proof.
Consider an arbitrary edge-weighted strongly connected directed graph (G, w). Let x be an optimal solution to $\mathrm{LP}(G, w)$ and let (α, y) be an optimal solution to $\mathrm{DUAL}(G, w)$ as guaranteed by Lemma 2.18, that is, y has laminar support $\mathcal{L}$. We now define a pair (G', w') as

$$V(G') = V(G)\,, \quad E(G') = \{e \in E(G) : x(e) > 0\}\,, \quad \text{and} \quad w'(u,v) = w(u,v) - \alpha_u + \alpha_v\,.$$

We claim that $\mathcal{I} = (G', \mathcal{L}, x, y)$ is a laminarly-weighted ATSP instance whose induced weight function $w_{\mathcal{I}}$ equals w'. To see this, recall that x is a primal optimal solution and that (α, y) is a dual optimal solution (for (G, w)). Therefore complementarity slackness implies that every set in $\mathcal{L}$ is tight with respect to x and that for every edge $(u, v) \in E(G')$, the weight $w'(u, v) = w(u, v) - \alpha_u + \alpha_v$ equals the sum of y_S-values for the sets S crossed by (u, v). Finally, we have $x_e > 0$ for every $e \in E(G')$ by definition. So $\mathcal{I}$ satisfies all the properties of Definition 3.3, i.e., it is a laminarly-weighted instance.

We now argue that an α-approximate solution for $\mathcal{I}$ with respect to the Held-Karp relaxation $\mathrm{LP}(G', w')$ implies an α-approximate solution for the original instance (G, w) with respect to $\mathrm{LP}(G, w)$. To this end, we make the following observation:

Claim. *For any circulation $x \in \mathbb{R}_+^{E(G')}$, we have $\sum_{e \in E(G')} w(e)x(e) = \sum_{e \in E(G')} w'(e)x(e)$.*

Therefore the Held-Karp lower bound is the same for (G, w) and for (G', w'), and any solution (integral or fractional) for (G', w') is a solution of the same weight for (G, w). □

In the rest of Part I we work exclusively with laminarly-weighted ATSP instances $\mathcal{I} = (G, \mathcal{L}, x, y)$. We will refer to them as simply *instances*.

Definition 3.5

We say that an instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ is a singleton instance *if all sets in $\mathcal{L}$ are singletons.*

Such instances will play an important role in our algorithm. In particular, note that for singleton instances, the weight function $w_{\mathcal{I}}$ is induced by nodes ($w(u, v) = y_u + y_v$ for all $(u, v) \in E$) just like in a node-weighted instance. The difference between singleton and node-weighted instances is that singleton instances are those *laminarly-weighted* instances whose weight function is induced by nodes after having performed the reduction of Theorem 3.4. A node-weighted instance does not necessarily give rise to a singleton instance.

Recall that $w_{\mathcal{I}}(F)$ is the induced weight of an edge multiset $F \subseteq E$ in the instance $\mathcal{I}$. We will omit the subscript and use simply $w(F)$ whenever $\mathcal{I}$ is clear from the context.

Definition 3.6

For an instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ and a set $S \subseteq V$ we define

$$\mathrm{value}_{\mathcal{I}}(S) = 2 \cdot \sum_{R \in \mathcal{L}:\ R \subsetneq S} y_R$$

to be the fractional dual value associated with the sets strictly inside S.

Again, we will omit the subscript whenever clear from the context. We also use $\mathrm{value}(\mathcal{I}) = \mathrm{value}_{\mathcal{I}}(V)$; note that this equals the Held-Karp lower bound of the instance. Indeed, as noted above, y can be extended to an optimal dual solution to $\mathrm{DUAL}(G, w)$, and hence the optimum value for $\mathrm{DUAL}(G, w)$ equals $2 \cdot \sum_{S \in \mathcal{L}} y_S$, which is equal to the primal optimum value $\sum_{e \in E} w(e)x(e)$ for $\mathrm{LP}(G, w)$ by strong duality (Theorem 2.5).

Chapter 4

Reducing ATSP to Subtour Partition Cover

In this chapter we define the Subtour Partition Cover problem and reduce the task of solving ATSP to that problem. This reduction will be used to solve general instances in Chapter 6. Here, we illustrate its power by giving a constant-factor approximation algorithm for singleton instances.

4.1 Subtour Partition Cover

In this section we define the *Subtour Partition Cover* problem that is obtained from ATSP by relaxing the connectivity requirements. A variant of this problem, called *Local-Connectivity ATSP*, was the key tool in [Sve15] for obtaining a constant-factor approximation for node-weighted instances. We now introduce a different variant that is more adapted to our approach for the general case.

Consider a laminarly-weighted instance $\mathcal{I} = (G, \mathcal{L}, x, y)$. For notational convenience we extend the vector y to all singletons so that $y_v = 0$ if $\{v\} \notin \mathcal{L}$. Let $\text{lb}_{\mathcal{I}} : V \to \mathbb{R}$ be the *lower bound function* defined by $\text{lb}_{\mathcal{I}}(v) = 2y_v$. We simplify notation and write lb instead of $\text{lb}_{\mathcal{I}}$ if $\mathcal{I}$ is clear from the context. Note that $\text{lb}(V)$ is at most the Held-Karp lower bound value($\mathcal{I}$), with equality only for singleton instances. For an edge set F, we use the simplified notation $\text{lb}(F) = \text{lb}(V(F))$ to denote the total lower bound of the vertices incident to F.

Perhaps the main difficulty of ATSP is to satisfy the connectivity requirement, i.e., to select a Eulerian subset F of edges that satisfies all subtour elimination constraints. In *Subtour Partition Cover*, this condition is relaxed and we only require that the subtour elimination constraints be satisfied for some disjoint sets.

Subtour Partition Cover

Given: An instance $\mathcal{I} = (G, \mathcal{L}, x, y)$, a subtour B in G, and a partition $(V_1, V_2, \ldots, V_k)$ of $V \setminus V(B)$ such that the graph induced by V_i is strongly connected for $i = 1, \ldots, k$.

Find: A collection F of subtours of E such that $|\delta_F^+(V_i)| \geq 1$ for $i = 1, 2, \ldots, k$.

Definition 4.1

We say that an algorithm for Subtour Partition Cover is (α, β)-light *for an instance* $\mathcal{I}$ *and subtour* B *if, for any input partition of strongly connected subsets, the collection* F *of subtours satisfies*

- $w_{\mathcal{I}}(T) \leq \alpha \operatorname{lb}(T)$ *for every subtour* T *in* F *with* $V(T) \cap V(B) = \emptyset$, *and*
- $w_{\mathcal{I}}(F_B) \leq \beta$, *where* $F_B \subseteq F$ *is the collection of subtours in* F *that intersect* B.[5]

We use the (α, β)-light terminology to avoid any ambiguities with the concept of approximation algorithms.

If we let c be the scaling factor such that $c \operatorname{lb}(V) = \text{value}(\mathcal{I})$, then an α-approximation algorithm for ATSP with respect to the Held-Karp relaxation is trivially an $(\alpha \cdot c, 0)$-light algorithm for Subtour Partition Cover with $B = \emptyset$: output the same tour F as the algorithm for ATSP. However, Subtour Partition Cover seems like a significantly easier problem than ATSP, as the set of subtours F only needs to cross k cuts formed by a partitioning of the vertices $V \setminus V(B)$. We substantiate this intuition by proving, in Section 4.2, that there exists a simple $(2, 0)$-light algorithm for Subtour Partition Cover on singleton instances with $B = \emptyset$. Perhaps more surprisingly, in Section 4.3 we show that an (α, β)-light algorithm for Subtour Partition Cover for an instance $\mathcal{I}$ with subtour B can be turned into an approximation algorithm for ATSP with an approximation guarantee of $(9 + \varepsilon)\alpha \operatorname{lb}(V \setminus V(B)) + \beta + w(B)$ for any $\varepsilon > 0$.

The main difference between the Subtour Partition Cover problem and the Local-Connectivity ATSP problem introduced in [Sve15] is the introduction of the subtour B and the more general definition of lightness. While this flexibility is unnecessary for singleton instances with $B = \emptyset$ (which are closely related to the node-weighted instances considered in that paper), it will be useful in the general case: in Section 6.1 we give an algorithm for Subtour Partition Cover that, in turn, implies the constant-factor approximation algorithm for general instances.

We remark that our generic reduction from ATSP to Subtour Partition Cover (Theorem 4.3) is robust with respect to the definition of lb and there are many possibilities to define such a lower bound. Another natural example is $\operatorname{lb}(v) = \sum_{e \in \delta^+(v)} x^*_e w(e)$. In [Sve15], Local-Connectivity ATSP was defined with this lb function, and with $B = \emptyset$; in this case we can set $\beta = 0$. In fact, in order to get a constant bound on the integrality gap of the Held-Karp relaxation, our results say that it is enough to find an $(O(1), 0)$-light algorithm for Subtour Partition Cover with respect to some nonnegative lb that only needs to satisfy that $\operatorname{lb}(V)$ is at most the value $\text{value}(\mathcal{I})$ of the optimal solution to the LP. Even more generally, if $\operatorname{lb}(V)$ is at most the value of an optimal tour (rather than the LP value) then our methods would give a similar approximation guarantee (but not with respect to the Held-Karp relaxation).

A variant of Subtour Partition Cover was used in [STV18b] to obtain a constant factor approximation guarantee for ATSP with two different edge weights; a key idea of that paper is the careful choice of the lb function.

[5] Recall that we say that a subtour T intersects another subtour B if they visit a common vertex, i.e., $V(T) \cap V(B) \neq \emptyset$. Hence, $F_B = \{T \text{ subtour in F} : V(T) \cap V(B) \neq \emptyset\}$.

4.2 Subtour Partition Cover for singleton instances

We give a simple $(2,0)$-light algorithm for Subtour Partition Cover for the special case of singleton instances, that is, when $\mathcal{L}$ is a singleton family, and for $B = \emptyset$.

The proof is based on finding an integral circulation that sends flow across the cuts $\{(V_i, \bar{V}_i) : i = 1, 2, \ldots, k\}$ and, in addition, satisfies that the outgoing flow of each vertex $v \in V$ with $y_v > 0$ is at most 2, which in turn, by the assumptions on the metric, implies a $(2,0)$-light algorithm.

Theorem 4.2

There exists a polynomial-time algorithm for Subtour Partition Cover that is $(2,0)$-light for singleton instances with $B = \emptyset$.

Proof.
Let $\mathcal{I} = (G, \mathcal{L}, x, y), B, (V_1, V_2, \ldots, V_k)$ be an instance of Subtour Partition Cover where $\mathcal{I}$ is a singleton instance and $B = \emptyset$. Let also $w = w_{\mathcal{I}}$ denote the induced weight function. We prove the theorem by giving a polynomial-time algorithm that finds a collection F of subtours satisfying

$$|\delta_F^+(V_i)| \geq 1 \text{ for } i = 1, \ldots, k \quad \text{and} \quad |\delta_F^+(v)| \leq 2 \text{ for } v \in V \text{ with } y_v > 0. \tag{4.1}$$

The first condition means that F crosses every cut $(V_i, \bar{V}_i)$, thus the algorithm indeed solves the Subtour Partition Cover problem. We show that the second condition implies $(2,0)$-lightness. Since $B = \emptyset$, we need to show that $w(T) \leq 2\,\mathrm{lb}(T)$ for every subtour T in F. Since $\mathcal{I}$ is a singleton instance, $w(u,v) = y_u + y_v$ for all $(u,v) \in E$ (recall the convention $y_u = 0$ if $\{u\} \notin \mathcal{L}$). Consider now any subtour T in F. We have

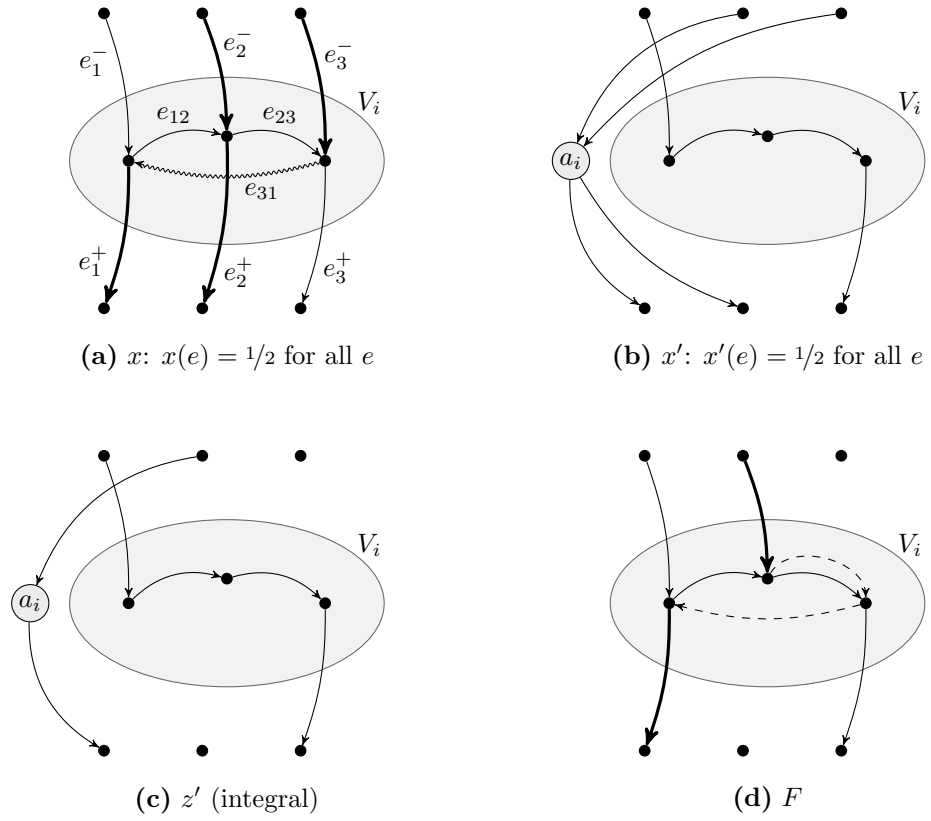
$$w(T) = \sum_{e \in T} w(e) = \sum_{v \in V(T)} |\delta_F(v)| y_v = 2 \sum_{v \in V(T)} |\delta_F^+(v)| y_v \leq 4 \sum_{v \in V(T)} y_v = 2\,\mathrm{lb}(T).$$

We proceed by describing a polynomial-time algorithm for finding an Eulerian set F satisfying (4.1). The set F will be obtained by rounding the circulation x to integrality while maintaining that it crosses each cut $(V_i, \bar{V}_i)$. For each cut $(V_i, \bar{V}_i)$ we introduce a new auxiliary vertex a_i to represent it. In lieu of requiring a flow of at least 1 through V_i, we will require such a flow through a_i. To show that such a (fractional) circulation exists, we modify x by redirecting an arbitrary flow of value 1 that passes through V_i to instead pass through a_i.

First, we transform G into a new graph G' and x into a new circulation x' by performing the following for each $i = 1, \ldots, k$ (see also Figure 4.1):

- Select a subset of incoming edges $X_i^- \subseteq \delta^-(V_i)$ with $x(X_i^-) = 1$. This is possible since $x(\delta^-(V_i)) \geq 1$.[6]
- Consider a cycle decomposition of x and follow the incoming edges in X_i^- in the decomposition. Select a subset of outgoing edges $X_i^+ \subseteq \delta^+(V_i)$ to be the set of edges on which these cycles first leave V_i after entering on an edge in X_i^-. We define a flow x_i to be the x-flow on these cycle segments connecting the heads of edges in X_i^- and the tails of edges in X_i^+.

[6] To obtain exactly 1, we might need to break an edge up into two copies, dividing its x-value between them appropriately, and include one copy in X_i^- but not the other; we omit this for simplicity of notation, and assume there is such an edge set with exactly $x(X_i^-) = 1$.



(a) x: $x(e) = 1/2$ for all e

(b) x': $x'(e) = 1/2$ for all e

(c) z' (integral)

(d) F

Figure 4.1: A depiction of the proof of Theorem 4.2. The neighborhood of a component V_i is shown.
(a) shows x, with $x(e) = 1/2$ on every shown edge. We select $X_i^- = \{e_2^-, e_3^-\}$ (thick incoming edges). Suppose that the cycle decomposition of x has a cycle containing e_1^-, e_{12}, e_{23}, e_3^+, a cycle containing e_2^-, e_2^+, and a cycle containing e_3^-, e_{31}, e_1^+. Thus we have $X_i^+ = \{e_1^+, e_2^+\}$ (thick outgoing edges), and the flow x_i (wiggly) puts value $1/2$ on e_{31}.
(b) shows x', with $x'(e) = 1/2$ on every shown edge. We redirect e_2^-, e_3^- to point to a_i and e_1^+, e_2^+ to point from a_i. (Note that the non-V_i endpoints of all the boundary edges of V_i might now also become a_j for some $j \neq i$.) We also subtract x_i, removing e_{31}.
(c) shows z', which is integral. Note that $z'(\delta^-(a_i)) = z'(\delta^+(a_i)) = 1$.
(d) shows the final solution F. The thick edges, which are redirected from the edges incident to a_i in z', guarantee that F crosses V_i. The path P_i is dashed.

- We introduce a new auxiliary vertex a_i and redirect all edges in X_i^- to point to a_i, and those in X_i^+ to point from a_i. We subtract the flow x_i from x.

Note that x' is a circulation, and that it satisfies the following conditions:

- $x'(\delta^+(v)) \leq 1$ for all $v \in V$ with $y_v > 0$,
- $x'(\delta^+(a_i)) = 1$ for all $i = 1, \ldots, k$.

Here, the first condition holds since all sets in $\mathcal{L}$ are tight and thus $x(\delta^+(v)) = 1$ whenever $y_v > 0$; the rest is by construction. As the vertex-degree bounds are integral, we can also,

in polynomial time, find an *integral* circulation z' that satisfies these two conditions (see e.g. Chapter 11 in [Sch03]).

Next, we map z' from G' to a flow z in G in the natural way: by reversing the redirection of the edges incident to the auxiliary vertices a_i while retaining their flow. Now, the flow z so obtained may not be a circulation. Specifically, since the in- and out-degree of a_i were exactly 1 in z', in each component V_i there is a pair of vertices u_i, v_i that are the head and tail, respectively, of the mapped-back edges adjacent to a_i. These are the only vertices whose in-degree in z may differ from their out-degree. (They differ unless $u_i = v_i$.) To repair this, for each $i = 1, \dots, k$ we route a walk P_i from u_i to v_i in V_i; this is always possible as we assumed that V_i is strongly connected (by the definition of Subtour Partition Cover).

We obtain our final solution F from z by taking every edge $e \in E$ with multiplicity z_e and adding the paths P_i. Note that F is Eulerian, i.e, a collection of subtours. To see that F satisfies (4.1), note that F crosses every cut $(V_i, \bar{V}_i)$ (since the edges redirected from a_i are boundary edges of V_i). Moreover, by the first property above and the fact that paths P_i are vertex-disjoint (being inside disjoint subsets V_i), we have $|\delta_F^+(v)| \leq 2$ for each $v \in V$ with $y_v > 0$. This concludes the proof of Theorem 4.2.

□

4.3 From local to global connectivity

In this section, we reduce the task of approximating ATSP to that of solving Subtour Partition Cover. To simplify the notation, for a subtour B, we let

$$\mathrm{lb}_{\mathcal{I}}(\bar{B}) = \mathrm{lb}_{\mathcal{I}}(V \setminus V(B)) = 2 \sum_{v \in V \setminus V(B)} y_v. \tag{4.2}$$

The main theorem can be stated as follows.

Theorem 4.3

Let $\mathcal{A}$ be an algorithm for Subtour Partition Cover. For any instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ and subtour B, if $\mathcal{A}$ is (α, β)-light for $\mathcal{I}$ and B, then there exists a tour of G of weight at most $5\alpha\,\mathrm{lb}_{\mathcal{I}}(\bar{B}) + \beta + w_{\mathcal{I}}(B)$. Moreover, for any $\varepsilon > 0$, a tour of weight at most $9(1+\varepsilon)\alpha\,\mathrm{lb}_{\mathcal{I}}(\bar{B}) + \beta + w_{\mathcal{I}}(B)$ can be found in time polynomial in the number $n = |V|$ of vertices, in $1/\varepsilon$, and in the running time of $\mathcal{A}$.

Using Theorem 4.2, we immediately obtain a constant-factor approximation for ATSP on singleton instances.

Corollary 4.4

For any $\varepsilon > 0$, there exists a polynomial-time $(18 + \varepsilon)$-approximation algorithm for ATSP on singleton instances.

In the sequel, we will use $\alpha_{\mathrm{S}} = 18 + \varepsilon$ for the approximation ratio for ATSP on singleton instances to make the dependence on this factor transparent.

Throughout this section we let $\mathcal{I} = (G, \mathcal{L}, x, y)$, B and $\mathcal{A}$ be fixed as in the statement of the theorem; we let $w = w_{\mathcal{I}}$ throughout. The proof of the theorem is by giving an algorithm that uses $\mathcal{A}$ as a subroutine. We first give the non-polynomial-time algorithm (with the better guarantee) in Section 4.3.1, followed by Section 4.3.2 where we modify the arguments so that we also efficiently find a tour (with a slightly worse guarantee).

4.3.1 Existence of a good tour

The idea of the algorithm is to start with a collection of subtours and then iteratively merge/connect them into a single tour that visits all vertices by adding additional (cheap) subtours. We remark that since we will only add Eulerian subsets of edges, the algorithm always maintains a collection of subtours. So the state of the algorithm is described by a collection of subtours T^*.

Initialization For the rest of the section, we assume that $V(B) \neq V$. Otherwise, B itself is a tour and the algorithm simply returns B. The algorithm starts by selecting nonempty subtours $T_1^*, T_2^*, \ldots, T_k^*$ such that

I1: $B, T_1^*, T_2^*, \ldots, T_k^*$ are disjoint subtours;

I2: $w(T_i^*) \leq 2\alpha \operatorname{lb}(T_i^*)$ for $i = 1, 2, \ldots, k$;

I3: the lexicographic order of $\langle \operatorname{lb}(T_1^*), \operatorname{lb}(T_2^*), \ldots, \operatorname{lb}(T_k^*) \rangle$ is maximized.

As the lexicographic order is maximized, the subtours are ordered so that $\operatorname{lb}(T_1^*) \geq \operatorname{lb}(T_2^*) \geq \cdots \geq \operatorname{lb}(T_k^*)$. The set T^* is initialized as $T^* = T_0^* \cup T_1^* \cup T_2^* \cup \cdots \cup T_k^*$, where we let $T_0^* = B$.

During the execution of the algorithm we will also use the following concept. For a subtour T of G, let $\operatorname{ind}(T)$ be the smallest index of a subtour in $T_0^*, \ldots, T_k^*$ that it intersects (or ∞ if it intersects none). That is,

$$\operatorname{ind}(T) = \min\{i : V(T_i^*) \cap V(T) \neq \emptyset\}.$$

Moreover, an important quantity will be $\operatorname{lb}(T^*_{\operatorname{ind}(T)})$, with the convention that $\operatorname{lb}(T^*_\infty) = 0$.

Remark 4.5

The main difference in the polynomial-time algorithm is the initialization, as we do not know how to find an initialization satisfying I1-I3 in polynomial time. Indeed, it is consistent with our knowledge that 2α (even 2) is an upper bound on the integrality gap and, in that case, such an algorithm would always find a tour for singleton instances with $B = \emptyset$.

Remark 4.6

For intuition, let us mention that the reason to maximize the lexicographic order (subject to I1-I2) is that we will use the following properties to bound the weight of the final tour:

1. *Let T be a subtour with $\operatorname{ind}(T) = i > 0$ and $w(T) \leq 2\alpha \operatorname{lb}(T)$. Then $\operatorname{lb}(T) \leq \operatorname{lb}(T_i^*)$.*
2. *For any disjoint subtours $T_1, T_2, \ldots, T_\ell$ with $\operatorname{ind}(T_j) = i > 0$ and $w(T_j) \leq \alpha \operatorname{lb}(T_j)$ for $j = 1, \ldots, \ell$, we have*

$$\sum_{j=1}^{\ell} \operatorname{lb}(T_j) \leq 2 \operatorname{lb}(T_i^*).$$

These claims will be used to bound the weight of the subtours added in the merge procedure (see below). Their proofs are easy and can be found in the analysis (see the proofs of Claim 3 and Claim 4).

Merge procedure After the initialization, T^* contains a collection of subtours that do not necessarily form a tour. The goal of the "merge procedure" is to form a tour of the entire graph, connecting these subtours by adding additional (cheap) subtours. We will do so while maintaining the invariant that $T_0^* = B$ is a disjoint subtour in T^* until the very last step when a tour is formed.

Specifically, the procedure repeats the following until T^* contains a tour that visits all the vertices. Let $T_0^*, T_1, \dots, T_\ell$ be the collection of subtours in T^* (recall that $T_0^* = B$). As they are disjoint, $T_1, T_2, \dots, T_\ell$ naturally partition the vertex set $V \setminus V(B)$ into $V(T_1), V(T_2), \dots, V(T_\ell)$ plus singleton sets for the remaining vertices in $V \setminus (V(B) \cup V(T_1) \cup \dots \cup V(T_\ell))$. Let $\mathcal{C}$ be this partitioning of $V \setminus V(B)$. By construction, each nonsingleton set in $\mathcal{C}$ corresponds to a subtour and thus, for each $V' \in \mathcal{C}$, the subgraph induced by V' is strongly connected. We can therefore use $\mathcal{A}$ to find a collection F of subtours such that

(i) $|\delta_F^+(V')| \geq 1$ for all $V' \in \mathcal{C}$,

(ii) $w_{\mathcal{I}}(T) \leq \alpha \operatorname{lb}(T)$ for every subtour T in F disjoint from B, and

(iii) $w(F_B) \leq \beta$, where $F_B \subseteq F$ is the collection of subtours in F that intersect B.

Note that $\mathcal{A}$ is guaranteed to find such a collection F since it is assumed to be an (α, β)-light algorithm for Subtour Partition Cover on $(\mathcal{I}, B)$. Furthermore, we may assume that a subtour T in F does not only visit a subset $V(T)$ of the vertices $V(T')$ visited by a subtour T' in T^*. Indeed, such a subtour can safely be removed from F, yielding a new (smaller) collection of subtours that satisfies the above conditions. Having selected F, we now proceed to explain the "update phase".

U1: Let $X = \emptyset$.

U2: Select a subtour T in $T^* \cup F \cup X$ that *maximizes* $\operatorname{ind}(T)$. Let $j = \operatorname{ind}(T)$.

U3: If there exists a cycle C of weight $w(C) \leq \alpha \operatorname{lb}(T_j^*)$ that connects T to other vertices, i.e., $V(T) \cap V(C) \neq \emptyset$ and $V(C) \not\subseteq V(T)$, then add C to X and repeat from Step U2.

U4: Otherwise, update T^* by adding the "new" edges in T, i.e., $T^* \leftarrow T^* \cup (T \cap F) \cup (T \cap X)$.

Some comments about the update of T^* are in order. We emphasize that we do *not* add all edges of $F \cup X$ to T^*. Instead, we only add those new edges that belong to the component T selected in the final iteration of the update phase. Among other things, this ensures the invariant that B is a subtour in T^* until the very end. Indeed, the iteration where we add a subtour intersecting $B = T_0^*$ must be the last iteration. This is because, in that case, the selected T that maximizes $\operatorname{ind}(T)$ must satisfy $\operatorname{ind}(T) = 0$, which in turn implies that $T^* \cup F \cup X$ is a single tour T that visits all vertices.

Finally, let us remark that the update maintains that T^* is a collection of subtours (i.e., an Eulerian multiset of edges). As T is a subtour in $T^* \cup F \cup X$, and F and X themselves are collections of subtours, we have that T^* remains a collection of subtours after the update. This finishes the description of the merging procedure and the algorithm (see also the example below).

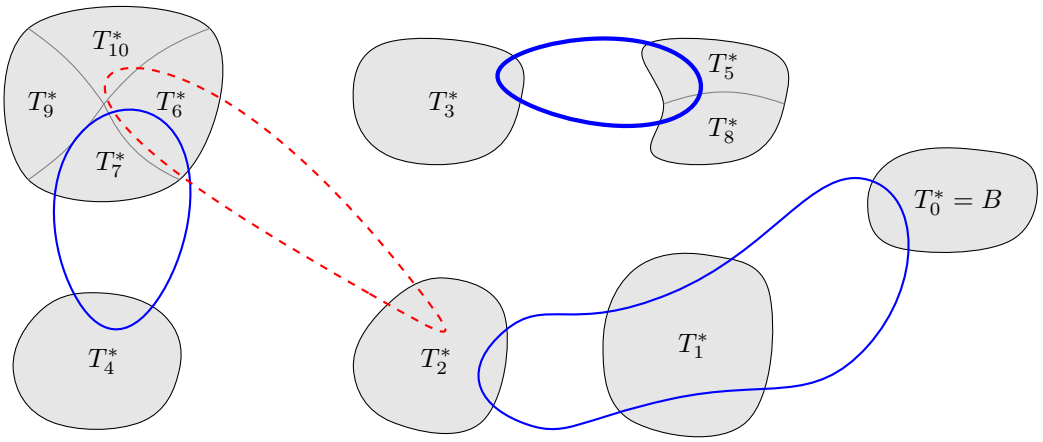


Figure 4.2: An illustration of the merge procedure. The gray areas depict the subtours in T^*. Blue (solid) cycles depict F and the red (dashed) cycle depicts X after one iteration of the update phase. The thick cycle represents the edges that this merge procedure would add to T^*.

Example 4.7

In Figure 4.2, we have that, at the start of a merging step, T^* consists of 7 subtours containing $\{T^*_6, T^*_7, T^*_9, T^*_{10}\}, \{T^*_3\}, \{T^*_5, T^*_8\}, \{T^*_4\}, \{T^*_2\}$, and $\{T^*_1\}$. The blue (solid) cycles depict the subtours of F. First, we set $X = \emptyset$ and the algorithm selects the subtour T in $T^* \cup F \cup X$ that maximizes $\mathrm{ind}(T)$. In this example, it would be the leftmost of the three subtours in $T^* \cup F$, with $\mathrm{ind}(T) = 4$. The algorithm now tries to connect this component to another component by adding a cycle with weight at most $\alpha\,\mathrm{lb}(T^*_4)$. The red (dashed) cycle corresponds to such a cycle and its edge set is added to X. In the next iteration, the algorithm considers the two subtours in $T^* \cup F \cup X$. The one that maximizes $\mathrm{ind}(T)$ contains T^*_3, T^*_5, and T^*_8. Now suppose that there is no cycle of weight at most $\alpha\,\mathrm{lb}(T^*_3)$ that connects this component to another component. Then the set T^* is updated by adding those subtours (edges) of $F \cup X$ that belong to this component (depicted by the thick cycle).

Analysis

Termination We show that the algorithm terminates by arguing that the update phase decreases the number of connected components and the merge procedure is therefore repeated at most $k \leq n$ times.

Lemma 4.8

The update phase terminates in polynomial time and decreases the number of connected components in (V, T^*).

Proof.
First, observe that each single step of the update phase can be implemented in polynomial time. The only nontrivial part is Step U3, which can be implemented as follows: for each edge $(u, v) \in \delta^+(V(T))$ consider the cycle consisting of (u, v) and a shortest path from v to u. Moreover, the entire update phase terminates in polynomial time, because each time the

if-condition of Step U3 is satisfied, we add a cycle to X that decreases the number of connected components in $(V, T^* \cup F \cup X)$. The if-condition of Step U3 can therefore be satisfied at most $k \leq n$ times.

We proceed by proving that, at termination, the update phase decreases the number of connected components in (V, T^*). Recall that, once the algorithm reaches Step U4, it has selected a subtour T in $T^* \cup F \cup X$. We claim that T visits vertices in at least two components of (V, T^*). This is because the subtours in F satisfy $|\delta_F^+(V')| \geq 1$ for all $V' \in \mathcal{C}$, where $\mathcal{C}$ denotes the connected components of $(V \setminus V(B), T^* \setminus B)$. Moreover, as already noted, T intersects B only in the last iteration, when T forms a tour. Therefore, when the algorithm updates T^* by adding the edges $(F \cup X) \cap T$, it decreases the number of components in (V, T^*) by at least one. □

Performance Guarantee We split our analysis of the performance guarantee into two parts. Note that when one execution of the merge procedure terminates (Step U4), we add the edge set $(F \cap T) \cup (X \cap T)$ to our solution. We will analyze the contribution of these two sets ($F \cap T$ and $X \cap T$) separately. More formally, suppose that the algorithm performs R repetitions of the merge procedure. Let $T_1, T_2, \ldots, T_R$, $F_1, F_2, \ldots, F_R$, and $X_1, X_2, \ldots, X_R$ denote the selected subtour T, the edge set F, and the edge set X, respectively, at the end of each repetition. To simplify notation, we denote the edges added to T^* in the r-th repetition by $\tilde{F}_r = F_r \cap T_r$ and $\tilde{X}_r = X_r \cap T_r$.

With this notation, we proceed to bound the total weight of the solution by

$$\underbrace{w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right)}_{\leq 2\alpha \operatorname{lb}(\bar{B}) + \beta \text{ by Lemma 4.10}} + \underbrace{w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right)}_{\leq \alpha \operatorname{lb}(\bar{B}) \text{ by Lemma 4.9}} + w(B) + \sum_{i=1}^{k} w(T_i^*)$$
$$\leq 5\alpha \operatorname{lb}(\bar{B}) + \beta + w(B),$$

as claimed in Theorem 4.3. Here we used that $\sum_{i=1}^{k} w(T_i^*) \leq 2\alpha \operatorname{lb}(\bar{B})$, which is implied by the selection of $T_1^*, T_2^*, \ldots, T_k^*$ (I1-I2). It remains to prove Lemmas 4.9 and 4.10.

Lemma 4.9

We have $w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right) \leq \alpha \operatorname{lb}(\bar{B})$.

Proof.
Note that $\tilde{X}_r$ consists of a subset of the cycles added to X_r in Step U3 of the update phase: specifically, of those cycles that were contained in the subtour T_r selected in Step U2 in the last iteration of the update phase during the r-th repetition of the merge procedure. We can therefore decompose $\bigcup_{r=1}^{R} \tilde{X}_r$ into cycles $C_1, C_2, \ldots, C_c$, indexed in the order they were added by the algorithm. We assume that all these cycles have strictly positive weight, as 0-weight cycles do not affect $w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right)$. At the time a cycle C_i (with $w(C_i) > 0$) was selected in Step U3 of the update phase, it satisfied the following two properties:

(i) it connected the subtour T with $\operatorname{ind}(T) = j > 0$ selected in Step U2 with at least one other subtour T' such that $\operatorname{ind}(T') < \operatorname{ind}(T)$; and

(ii) it had weight $w(C_i) \leq \alpha \operatorname{lb}(T_j^*)$.

In this case, we say that C_i is marked by j. Note that $1 \leq j \leq k$, since $\alpha \operatorname{lb}(T_j^*) \geq w(C_i) > 0$ and by convention $\operatorname{lb}(T_\infty^*) = 0$.

We claim that at most one cycle in $C_1, C_2, \ldots, C_c$ is marked by each of the numbers $\{1, 2, \ldots, k\}$. To see this, consider the first cycle C_i marked by j (if any). By *(i)* above, when C_i was added, it connected two subtours T and T' such that $\operatorname{ind}(T') < \operatorname{ind}(T) = j$. As the algorithm only adds edges, T and T' will remain connected throughout the execution of the algorithm. Therefore, by the definition of ind and by the fact that $\operatorname{ind}(T') < \operatorname{ind}(T)$, we have that a subtour T'' selected in Step U2 later in the algorithm always has $\operatorname{ind}(T'') \neq j$. Hence, no other cycle will be marked by j.

The bound now follows since at most one cycle with positive weight is marked by j, and such a cycle has weight at most $\alpha \operatorname{lb}(T_j^*)$. Moreover, we have $\sum_{j=1}^{k} \alpha \operatorname{lb}(T_j^*) \leq \alpha \operatorname{lb}(\bar{B})$, which is again implied by the selection of $T_1^*, T_2^*, \ldots, T_k^*$ (I1-I2). □

We complete the analysis of the performance guarantee with the following lemma. We remark that this is the only part of the proof that relies on the initial subtours $T_1^*, \ldots, T_k^*$ maximizing the lexicographic order (I3).

Lemma 4.10

We have $w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right) \leq 2\alpha \operatorname{lb}(\bar{B}) + \beta$.

Proof.
Consider the r-th repetition of the merge procedure. Partition the collection of subtours $\tilde{F}_r$ into

$$\tilde{F}_r^i = \{T \text{ subtour in } \tilde{F}_r : \operatorname{ind}(T) = i\} \quad \text{for } i \in \{0, 1 \ldots, k, \infty\}.$$

That is, $\tilde{F}_r^i$ contains those subtours in $\tilde{F}_r$ that intersect T_i^* and do not intersect any of the subtours $T_0^* = B, T_1^*, T_2^*, \ldots, T_{i-1}^*$ (or intersect none if $i = \infty$). The total weight $w(\tilde{F}_r)$ of $\tilde{F}_r$ thus equals

$$w(\tilde{F}_r^0) + w(\tilde{F}_r^\infty) + \sum_{i=1}^{k} w(\tilde{F}_r^i)\,.$$

We bound the weight of $\tilde{F}_r$ by considering these terms separately. Let us start with $\tilde{F}_r^0$.

Claim 2

The set $\tilde{F}_r^0$ *can be nonempty only for* $r = R$, *and* $w(\tilde{F}_R^0) \leq \beta$.

Proof.
The first claim follows by the invariant that B is a subtour in T^* until the very last iteration of the merge procedure. Indeed, if $\tilde{F}_r^0 \neq \emptyset$, then the algorithm must terminate after the r-th merge procedure: the subtour T selected in Step U2 must visit all vertices. For the second part, we have that every $T \in \tilde{F}_R^0$ must intersect $T_0^* = B$. Therefore, property (iii) of the edge set $\tilde{F}_R$ returned by $\mathcal{A}$ asserts that $w(\tilde{F}_R^0) \leq \beta$. □

For $i > 0$, we start by two simple claims that follow since each subtour T in $\tilde{F}_r$ satisfies $w(T) \leq \alpha \operatorname{lb}(T)$ (by property (ii) of $\mathcal{A}$) and the choice of $T_1^*, \ldots, T_k^*$ to maximize the lexicographic order I3 subject to I1-I2.

Claim 3

For $i > 0$ and $T \in \tilde{F}_r^i$ we have $\mathrm{lb}(T) \leq \mathrm{lb}(T_i^*)$.

Proof.
The inequality $\mathrm{lb}(T) > \mathrm{lb}(T_i^*)$ together with the fact that $w(T) \leq \alpha\,\mathrm{lb}(T) \leq 2\alpha\,\mathrm{lb}(T)$ would contradict that $T_1^*, \ldots, T_k^*$ was chosen to maximize the lexicographic order I3. Indeed, in that case, a an initialization satisfying I1-I2 of higher lexicographic order would be $T_1^*, \ldots, T_{i-1}^*, T$. □

The above claim already implies that $w(\tilde{F}_r^\infty) \leq \alpha\,\mathrm{lb}(\tilde{F}_r^\infty) \leq \alpha\,\mathrm{lb}(T_\infty^*) = 0$. We now present a more general claim that also applies to $\tilde{F}_r^i$ with $1 \leq i \leq k$.

Claim 4

If $i > 0$, then we have $\mathrm{lb}(\tilde{F}_r^i) \leq 2\,\mathrm{lb}(T_i^*)$.

Proof.
Suppose towards a contradiction that $\mathrm{lb}(\tilde{F}_r^i) > 2\,\mathrm{lb}(T_i^*)$. Let $T_1, T_2, \ldots, T_\ell$ be the subtours in $\tilde{F}_r^i$ and define T to be the subtour obtained by taking the union of the subtours T_i^* and $T_1, \ldots, T_\ell$. Consider the initialization $T_1^*, \ldots, T_{i-1}^*, T$. By construction these subtours are disjoint and disjoint from B since $i > 0$. Thus I1 is satisfied. Moreover, we have $\mathrm{lb}(T) > \mathrm{lb}(T_i^*)$ and therefore the lexicographic value of this initialization is larger than the lexicographic value of $T_1^*, \ldots, T_k^*$. This is a contradiction if T also satisfies I2, i.e., if $w(T) \leq 2\alpha\,\mathrm{lb}(T)$.

Therefore, we must have $w(T) > 2\alpha\,\mathrm{lb}(T)$. By the facts that $w(T_j) \leq \alpha\,\mathrm{lb}(T_j)$ (by property (ii) of $\mathcal{A}$) and that $w(T_i^*) \leq 2\alpha\,\mathrm{lb}(T_i^*)$ (by I2),

$$w(T) = w(T_i^*) + \sum_{j=1}^{\ell} w(T_j) \leq 2\alpha\,\mathrm{lb}(T_i^*) + \sum_{j=1}^{\ell} \alpha\,\mathrm{lb}(T_j) \quad \text{and} \quad \mathrm{lb}(T) \geq \sum_{j=1}^{\ell} \mathrm{lb}(T_j).$$

These inequalities together with $w(T) > 2\alpha\,\mathrm{lb}(T)$ imply $\mathrm{lb}(\tilde{F}_r^i) = \sum_{j=1}^{\ell} \mathrm{lb}(T_j) \leq 2\,\mathrm{lb}(T_i^*)$. □

Using the above claims, we can write $w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right)$ as

$$\begin{aligned}
\sum_{r=1}^{R} \left(w(\tilde{F}_r^0) + w(\tilde{F}_r^\infty) + \sum_{i=1}^{k} w(\tilde{F}_r^i) \right) &\leq \beta + \sum_{r=1}^{R} \sum_{i=1}^{k} w(\tilde{F}_r^i) \\
&\leq \beta + \alpha \sum_{r=1}^{R} \sum_{i=1}^{k} \mathrm{lb}(\tilde{F}_r^i) \\
&= \beta + \alpha \sum_{i=1}^{k} \sum_{r:\ \tilde{F}_r^i \neq \emptyset} \mathrm{lb}(\tilde{F}_r^i) \\
&\leq \beta + 2\alpha \sum_{i=1}^{k} \sum_{r:\ \tilde{F}_r^i \neq \emptyset} \mathrm{lb}(T_i^*).
\end{aligned}$$

We complete the proof of the lemma by using Claim 3 to prove that $\tilde{F}_r^i$ is nonempty for at most one repetition r of the merge procedure. Suppose towards a contradiction that there exist $1 \leq r_0 < r_1 \leq R$ such that both $\tilde{F}_{r_0}^i \neq \emptyset$ and $\tilde{F}_{r_1}^i \neq \emptyset$. In the r_0-th repetition of the merge

procedure, T_i^* was contained in the subtour T_{r_0} (selected in Step U2) since otherwise no edges incident to T_i^* would have been added to T^*. Therefore $j = \mathrm{ind}(T_{r_0}) \leq i$. Now consider a subtour T in $\tilde{F}_{r_1}^i$. First, recall that we have assumed that T, being a subtour in F_{r_1}, does not only visit a subset $V(T)$ of vertices $V(T')$ visited by a subtour T' in T^*. In particular, since T_{r_0} is a subset of some subtour T' in T^* during the r_1-th repetition, we have $V(T) \not\subseteq V(T_{r_0})$. Second, by Claim 3, we have $w(T) \leq \alpha \,\mathrm{lb}(T) \leq \alpha \,\mathrm{lb}(T_i^*)$.

In short, T is a subtour that connects T_{r_0} to another component and it has weight at most $\alpha \,\mathrm{lb}(T_i^*) \leq \alpha \,\mathrm{lb}(T_j^*)$, where $j = \mathrm{ind}(T_{r_0}) \leq i$. As T is Eulerian, it can be decomposed into cycles. One of these cycles, say C, connects T_{r_0} to another component and

$$w(C) \leq w(T) \leq \alpha \,\mathrm{lb}(T_j^*). \tag{4.3}$$

In other words, there exists a cycle C that, in the r_0-th repetition of the merge procedure, satisfied the if-condition of Step U3, which contradicts the fact that C was not added during the r_0-th repetition. □

4.3.2 Polynomial-time algorithm

In this section we describe how to modify the arguments in Section 4.3.1 to obtain an algorithm that runs in time polynomial in the number n of vertices, in $1/\varepsilon$, and in the running time of $\mathcal{A}$.

By Lemma 4.8, the update phase can be implemented in time polynomial in n. Therefore, the merge procedure described in Section 4.3.1 runs in time polynomial in n and in the running time of $\mathcal{A}$. The problem is the initialization: as mentioned in Remark 4.5, it seems difficult to give a polynomial-time algorithm for finding subtours $T_1^*, \ldots, T_k^*$ that satisfy I1 and I2 together with the third condition I3 that we should maximize the lexicographic order of

$$\langle \mathrm{lb}(T_1^*), \mathrm{lb}(T_2^*), \ldots, \mathrm{lb}(T_k^*) \rangle.$$

We overcome this obstacle by first identifying the properties that we actually use from selecting the subtours as above. We then show that we can obtain an initialization that satisfies these properties in polynomial time. Our initialization will still satisfy I1 and a relaxed variant of I2 that we now describe. To simplify notation, define

$$\overline{\mathrm{lb}}(T) = \mathrm{lb}(T) + \varepsilon \cdot \frac{|V(T)|}{n} \cdot \mathrm{lb}(\bar{B})$$

for a subtour T. Note that $\overline{\mathrm{lb}}(T)$ is a slightly increased version of $\mathrm{lb}(T)$. This increase is used to lower-bound the progress in Lemma 4.12. Also note that $\overline{\mathrm{lb}}$, like lb, is additive over vertex-disjoint subtours. Finally, we reprise the convention that $\overline{\mathrm{lb}}(T_\infty^*) = \mathrm{lb}(T_\infty^*) = 0$.

Our initializations will be collections $T_1^*, \ldots, T_k^*$ of subtours satisfying

I1: $B, T_1^*, T_2^*, \ldots, T_k^*$ are disjoint subtours;

I2': $w(T_i^*) \leq 3\alpha \,\overline{\mathrm{lb}}(T_i^*)$ for $i = 1, 2, \ldots, k$.

While we do not maximize the lexicographic order, we assume that the subtours are indexed so that $\overline{\mathrm{lb}}(T_1^*) \geq \overline{\mathrm{lb}}(T_2^*) \geq \ldots \geq \overline{\mathrm{lb}}(T_k^*)$. Note that the difference between I2' and I2 is that here we require $w(T_i^*) \leq 3\alpha \,\overline{\mathrm{lb}}(T_i^*)$ instead of $w(T_i^*) \leq 2\alpha \,\mathrm{lb}(T_i^*)$. The reason why we use a factor of 3 instead of 2 is that it leads to a better constant when balancing the parameters and, as previously mentioned, we use $\overline{\mathrm{lb}}$ instead of lb to lower-bound the progress in Lemma 4.12.

The main change to our initialization to achieve polynomial running time is that we do *not* maximize the lexicographic order (I3). As mentioned in the analysis in Section 4.3.1, the only way we use that the initialization maximizes the lexicographic order is for the proof of Lemma 4.10. In particular, this is used in the proofs of Claims 3 and 4. Instead of maximizing the lexicographic order, our polynomial-time algorithm will ensure a relaxed variant of those claims (formalized in the lemma below: see Condition (4.4)). The claimed polynomial-time algorithm is then obtained by first proving that a slight modification of the merge procedure returns a tour of value at most $9(1+\varepsilon)\alpha \operatorname{lb}(\bar{B}) + \beta + w(B)$ if Condition (4.4) holds, and then showing that an initialization satisfying this condition (and I1,I2') can be found in time polynomial in n and in the running time of $\mathcal{A}$. We start by describing the modification to the merge procedure.

Modified merge procedure The only modification to the merge procedure in Section 4.3.1 is that we change the update phase by relaxing the condition of the if-statement in Step U3 from $w(C) \leq \alpha \operatorname{lb}(T_j^*)$ to $w(C) \leq 3\alpha \overline{\operatorname{lb}}(T_j^*)$, where $j = \operatorname{ind}(T)$ and T is the subtour selected in Step U2. In other words, Step U3 is replaced by:

U3': If there exists a cycle C of weight $w(C) \leq 3\alpha \overline{\operatorname{lb}}(T_j^*)$ that connects T to other vertices, i.e., $V(T) \cap V(C) \neq \emptyset$ and $V(C) \not\subseteq V(T)$, then add C to X and repeat from Step U2.

Clearly the modified merge procedure still runs in time polynomial in n and in the running time of $\mathcal{A}$. Moreover, we show that if Condition (4.4) holds then the returned tour will have the desired weight. Recall from Section 4.3.1 that $\tilde{F}_r$ denotes the subset of F and $\tilde{X}_r$ denotes the subset of X that were added in the r-th repetition of the (modified) merge procedure. Furthermore, we define (as in the previous section) $\tilde{F}_r^i = \{T \text{ subtour in } \tilde{F}_r : \operatorname{ind}(T) = i\}$.

Lemma 4.11

Suppose that the algorithm is initialized with subtours $T_1^*, T_2^*, \ldots, T_k^*$ satisfying I1 and I2'. If in each repetition r of the modified merge procedure we add a subset $\tilde{F}_r$ such that

$$\operatorname{lb}(\tilde{F}_r^i) \leq 3\overline{\operatorname{lb}}(T_i^*) \qquad \text{for all } i \in \{1, 2, \ldots, k, \infty\}, \tag{4.4}$$

then the returned tour has weight at most $9(1+\varepsilon)\alpha \operatorname{lb}(\bar{B}) + \beta + w(B)$.

Let us comment on the above statement before giving its proof. The bound (4.4) is a relaxation of the bound of Claim 4 from $\operatorname{lb}(\tilde{F}_r^i) \leq 2\operatorname{lb}(T_i^*)$ to $\operatorname{lb}(\tilde{F}_r^i) \leq 3\overline{\operatorname{lb}}(T_i^*)$; and it also implies a relaxed version of Claim 3: from $\operatorname{lb}(T) \leq \operatorname{lb}(T_i^*)$ to $\operatorname{lb}(T) \leq 3\overline{\operatorname{lb}}(T_i^*)$ (for every T in $\tilde{F}_r^i$). It is because of this relaxed bound that we modified the if-condition of the update phase (by relaxing it by the same amount); this will be apparent in the proof.

Proof.

As in the analysis of the performance guarantee in Section 4.3.1, we can write the weight of the returned tour as

$$w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right) + w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right) + w(B) + \sum_{i=1}^{k} w(T_i^*).$$

To bound $w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right)$, we observe that the proof of Lemma 4.9 generalizes verbatim except that the weight of a cycle marked by i is now bounded by $3\alpha \overline{\operatorname{lb}}(T_i^*)$ instead of by $\alpha \operatorname{lb}(T_i^*)$

(because of the relaxation of the bound in the if-condition in Step U3'). Hence

$$w\left(\bigcup_{r=1}^{R}\tilde{X}_r\right) \leq \sum_{i=1}^{k} 3\alpha\,\overline{\mathrm{lb}}(T_i^*)\,.$$

We proceed to bound $w\left(\bigcup_{r=1}^{R}\tilde{F}_r\right)$. Using the same arguments as in the proof of Lemma 4.10, we get

$$\begin{aligned} w\left(\bigcup_{r=1}^{R}\tilde{F}_r\right) &= \sum_{r=1}^{R}\left(w(\tilde{F}_r^0)+w(\tilde{F}_r^\infty)+\sum_{i=1}^{k}w(\tilde{F}_r^i)\right) \leq \beta+\sum_{r=1}^{R}\sum_{i=1}^{k}w(\tilde{F}_r^i) \\ &\leq \beta+\alpha\sum_{i=1}^{k}\sum_{r:\,\tilde{F}_r^i\neq\emptyset}\mathrm{lb}(\tilde{F}_r^i) \leq \beta+\alpha\sum_{i=1}^{k}\sum_{r:\,\tilde{F}_r^i\neq\emptyset}3\,\overline{\mathrm{lb}}(T_i^*) \end{aligned}$$

where, for the first inequality, we used that Claim 2: $\sum_{r=1}^{R} w(\tilde{F}_r^0) \leq \beta$ generalizes verbatim from the non-constructive analysis and we have $w(\tilde{F}_r^\infty) \leq \alpha\,\mathrm{lb}(\tilde{F}_r^\infty) \leq 3\alpha\,\overline{\mathrm{lb}}(T_\infty^*) = 0$ by the assumption of the lemma; similarly, the last inequality is by the assumption of the lemma.

Now using that the subtours are indexed so that $\overline{\mathrm{lb}}(T_1^*) \geq \overline{\mathrm{lb}}(T_2^*) \geq \ldots \geq \overline{\mathrm{lb}}(T_k^*)$ we apply exactly the same arguments as in the end of the proof of Lemma 4.10 to prove that $\tilde{F}_r^i$ is nonempty for at most one repetition r of the merge procedure. The only difference is that (4.3) becomes

$$w(C) \leq w(T) \leq 3\alpha\,\overline{\mathrm{lb}}(T_j^*)$$

(because (4.4) can be seen as a relaxed version of Claim 3). However, as we also updated the bound in the if-condition, the argument that C would satisfy the if-condition of Step U3' is still valid. Hence, we conclude that $\tilde{F}_r^i$ is nonempty in at most one repetition and therefore

$$w\left(\bigcup_{r=1}^{R}\tilde{F}_r\right) \leq \beta+\alpha\sum_{i=1}^{k}\sum_{r:\,\tilde{F}_r^i\neq\emptyset}3\,\overline{\mathrm{lb}}(T_i^*) \leq \beta+\sum_{i=1}^{k}3\alpha\,\overline{\mathrm{lb}}(T_i^*)\,.$$

By the above bounds and since the initialization $T_1^*, T_2^*, \ldots, T_k^*$ satisfies I1 and I2', the weight of the returned tour is

$$\begin{aligned} &w\left(\bigcup_{r=1}^{R}\tilde{F}_r\right)+w\left(\bigcup_{r=1}^{R}\tilde{X}_r\right)+w(B)+\sum_{i=1}^{k}w(T_i^*) \\ &\qquad\leq \beta+\sum_{i=1}^{k}3\alpha\,\overline{\mathrm{lb}}(T_i^*)+\sum_{i=1}^{k}3\alpha\,\overline{\mathrm{lb}}(T_i^*)+w(B)+\sum_{i=1}^{k}w(T_i^*) \\ &\qquad\leq 9\alpha\sum_{i=1}^{k}\overline{\mathrm{lb}}(T_i^*)+\beta+w(B) \\ &\qquad\leq 9(1+\varepsilon)\alpha\,\mathrm{lb}(\bar{B})+\beta+w(B). \end{aligned}$$

□

Finding a good initialization in polynomial time By the above Lemma 4.11, it is sufficient to find an initialization such that I1, I2' are satisfied and Condition (4.4) holds during the execution of the modified merge procedure. However, how can we do it in polynomial time? We proceed as follows. First, we select the trivial *empty* initialization that consists of no subtours. Then we run the modified merge procedure and, in each repetition, we verify that Condition (4.4) holds. Note that this condition is easy to verify in time polynomial in n. If it holds until we return a tour, then we know by Lemma 4.11 that the tour has weight at most $9(1+\varepsilon)\alpha \operatorname{lb}(\bar{B}) + \beta + w(B)$. If it does not hold during some repetition, then we will restart the algorithm with a new initialization that we find using the following lemma. We continue in this manner until the merge procedure executes without violating Condition (4.4) and therefore returns a tour of weight at most $9(1+\varepsilon)\alpha \operatorname{lb}(\bar{B}) + \beta + w(B)$.

Lemma 4.12

Suppose that some repetition of the (modified) merge procedure violates Condition (4.4) *when run starting from an initialization $T_1^*, T_2^*, \ldots, T_k^*$ satisfying I1 and I2'. Then we can, in time polynomial in n, find a new initialization $T_1', T_2', \ldots, T_{k'}'$ such that I1, I2' are satisfied and*

$$\sum_{j=1}^{k'} \overline{\operatorname{lb}}(T_j')^2 - \sum_{j=1}^{k} \overline{\operatorname{lb}}(T_j^*)^2 \geq \frac{\varepsilon^2}{3n^2} \operatorname{lb}(\bar{B})^2. \tag{4.5}$$

Note that the above lemma implies that we will reinitialize (in polynomial time) at most $3n^2(1+\varepsilon)^2/\varepsilon^2$ times, because any initialization $T_1^*, \ldots, T_k^*$ has $\sum_{i=1}^{k} \overline{\operatorname{lb}}(T_i^*)^2 \leq \big((1+\varepsilon)\operatorname{lb}(\bar{B})\big)^2$. As each execution of the merge procedure takes time polynomial in n and in the running time of $\mathcal{A}$, we can therefore find a tour of weight at most $9(1+\varepsilon)\alpha \operatorname{lb}(\bar{B}) + \beta + w(B)$ in the time claimed in Theorem 4.3, i.e., polynomial in n, $1/\varepsilon$, and the running time of $\mathcal{A}$. It remains to prove the lemma.

Proof.
Suppose the r-th repetition of the merge procedure violates Condition (4.4), that is, there is an $i \in \{1, 2, \ldots, k, \infty\}$ such that

$$\operatorname{lb}(\tilde{F}_r^i) > 3\,\overline{\operatorname{lb}}(T_i^*)\,.$$

Suppose first $i = \infty$. Then $\tilde{F}_r^\infty \neq \emptyset$. Let T be a subtour in $\tilde{F}_r^\infty$. We have $w(T) \leq \alpha \operatorname{lb}(T) \leq \alpha\,\overline{\operatorname{lb}}(T)$ by property (ii) of $\mathcal{A}$ and T is disjoint from $T_1^*, \ldots, T_k^*$ and B by the definition of $\tilde{F}_r^\infty$. We can therefore compute (in polynomial time) a new initialization $T_1' = T_1^*, T_2' = T_2^*, \ldots, T_k' = T_k^*, T_{k+1}' = T$ with $k' = k+1$ such that I1, I2' are satisfied and

$$\sum_{j=1}^{k'} \overline{\operatorname{lb}}(T_j')^2 - \sum_{j=1}^{k} \overline{\operatorname{lb}}(T_j^*)^2 = \overline{\operatorname{lb}}(T_{k+1}')^2 = \overline{\operatorname{lb}}(T)^2 \geq \frac{\varepsilon^2}{n^2} \operatorname{lb}(\bar{B})^2\,,$$

where the last inequality is by the definition of $\overline{\operatorname{lb}}$ and $|V(T)| \geq 1$.

We now consider the case when $\operatorname{lb}(\tilde{F}_r^i) > 3\,\overline{\operatorname{lb}}(T_i^*)$ for an $i \in \{1, \ldots, k\}$. Let $I \subseteq \{1, 2, \ldots, k\}$ be the indices of those subtours of $T_1^*, T_2^*, \ldots, T_k^*$ that intersect subtours in $\tilde{F}_r^i$. Note that, by definition, we have $i \in I$ and $j \geq i$ for all $j \in I$. We construct a new initialization as follows:

- Sort $I \setminus \{i\} = \{t_1, \ldots, t_{|I|-1}\}$ so that

$$\frac{\overline{\text{lb}}(T^*_{t_j} \setminus \tilde{F}^i_r)}{\overline{\text{lb}}(T^*_{t_j} \cap \tilde{F}^i_r)} \geq \frac{\overline{\text{lb}}(T^*_{t_{j+1}} \setminus \tilde{F}^i_r)}{\overline{\text{lb}}(T^*_{t_{j+1}} \cap \tilde{F}^i_r)},$$

  where for a subtour T we simplify notation by writing $\overline{\text{lb}}(T \setminus \tilde{F}^i_r)$ and $\overline{\text{lb}}(T \cap \tilde{F}^i_r)$ for $\overline{\text{lb}}(V(T) \setminus V(\tilde{F}^i_r))$ and $\overline{\text{lb}}(V(T) \cap V(\tilde{F}^i_r))$, respectively.

- Let S be the minimal (possibly empty) prefix of indices $t_1, t_2, \ldots, t_s$ such that

$$\sum_{j \in S} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r) \geq \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r) - \overline{\text{lb}}(T^*_i).$$

- Define T^* to be the subtour obtained by taking T^*_i, the union of all the subtours in $\tilde{F}^i_r$, and all the subtours $\{T^*_j\}_{j \in S}$. Note that this is a single (i.e., connected) subtour since every subtour in $\tilde{F}^i_r$ intersects T^*_i, and every subtour T^*_j with $j \in S \subseteq I$ intersects a subtour in $\tilde{F}^i_r$.

- Reinitialize with subtours T^* and $\{T^*_j\}_{j \notin I}$.

All the above steps can be computed in polynomial time. Moreover, the new initialization still satisfies I1 by the definition of I and since T^* does not intersect B. We now use the way S was selected to prove that I2' still holds and that the "potential" function has increased as stated in (4.5). As we will calculate below, the increase of the potential function is simply because we required that $\sum_{j \in S} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r) \geq \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r) - \overline{\text{lb}}(T^*_i)$. That I2' holds, i.e., that $w(T^*) \leq 3\alpha\, \overline{\text{lb}}(T^*)$, follows since we selected S to be the minimal prefix with respect to the prescribed ordering, which prefers subtours that have small overlap with $\tilde{F}^i_r$ and therefore contribute significantly to $\overline{\text{lb}}(T^*)$. We now formalize this intuition.

Claim 5

We have $w(T^*) \leq 3\alpha\, \overline{\text{lb}}(T^*)$.

Proof.
As S is selected to be a minimal prefix and every $j \in I$ satisfies $\overline{\text{lb}}(T^*_j) \leq \overline{\text{lb}}(T^*_i)$, we claim that

$$\sum_{j \in S} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r) \leq \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r). \tag{4.6}$$

This is trivially true if $S = \emptyset$; otherwise, since the prefix $S \setminus \{t_s\}$ was not chosen, we had

$$\sum_{j \in S \setminus \{t_s\}} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r) < \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r) - \overline{\text{lb}}(T^*_i)$$

and thus indeed

$$\sum_{j \in S} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r) < \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\text{lb}}(T^*_j \setminus \tilde{F}^i_r) \underbrace{- \overline{\text{lb}}(T^*_i) + \overline{\text{lb}}(T^*_{t_s} \setminus \tilde{F}^i_r)}_{\leq 0}.$$

Moreover, by the sorting of the indices in $I \setminus \{i\}$ we must then also have

$$\sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i) \leq \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i)\,, \tag{4.7}$$

which is again trivially true if $S = \emptyset$; otherwise we write

$$\begin{aligned}
\frac{2}{3} \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i) \cdot \frac{\overline{\mathrm{lb}}(T_{t_s}^* \setminus \tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_{t_s}^* \cap \tilde{F}_r^i)} &\leq \frac{2}{3} \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i) \cdot \frac{\overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i)} = \frac{2}{3} \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) \\
&\leq \frac{1}{3} \sum_{j \in I \setminus \{i\} \setminus S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) = \frac{1}{3} \sum_{j \in I \setminus \{i\} \setminus S} \overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i) \cdot \frac{\overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i)} \\
&\leq \frac{1}{3} \sum_{j \in I \setminus \{i\} \setminus S} \overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i) \cdot \frac{\overline{\mathrm{lb}}(T_{t_s}^* \setminus \tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_{t_s}^* \cap \tilde{F}_r^i)}\,,
\end{aligned}$$

where the middle inequality is by subtracting $\frac{1}{3} \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i)$ from both sides of (4.6), and the remaining two are due to our sorting of indices. Next, we divide both sides by $\frac{\overline{\mathrm{lb}}(T_{t_s}^* \setminus \tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_{t_s}^* \cap \tilde{F}_r^i)}$ (which is nonzero by minimality of S) and add $\frac{1}{3} \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i)$ to both sides to obtain (4.7).

By (4.7) we have

$$\begin{aligned}
\sum_{j \in S} w(T_j^*) \leq 3\alpha \sum_{j \in S} \overline{\mathrm{lb}}(T_j^*) &= 3\alpha \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) + 3\alpha \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i) \\
&\leq 3\alpha \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) + \alpha \sum_{j \in I \setminus \{i\}} \overline{\mathrm{lb}}(T_j^* \cap \tilde{F}_r^i) \\
&\leq 3\alpha \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) + \alpha\, \overline{\mathrm{lb}}(\tilde{F}_r^i)\,,
\end{aligned}$$

where the first inequality holds because $T_1^*, \ldots, T_k^*$ satisfy I2' and the last inequality holds because they are disjoint (I1). By property (ii) of $\mathcal{A}$ and by the assumption that $\mathrm{lb}(\tilde{F}_r^i) > 3\,\overline{\mathrm{lb}}(T_i^*)$ we also have respectively that

$$w(\tilde{F}_r^i) \leq \alpha\, \mathrm{lb}(\tilde{F}_r^i) \leq \alpha\, \overline{\mathrm{lb}}(\tilde{F}_r^i) \qquad \text{and} \qquad w(T_i^*) \leq 3\alpha\, \overline{\mathrm{lb}}(T_i^*) < \alpha\, \overline{\mathrm{lb}}(\tilde{F}_r^i)\,.$$

These inequalities imply the claim since

$$\begin{aligned}
w(T^*) &= w(\tilde{F}_r^i) + w(T_i^*) + \sum_{j \in S} w(T_j^*) \\
&< \alpha\, \overline{\mathrm{lb}}(\tilde{F}_r^i) + \alpha\, \overline{\mathrm{lb}}(\tilde{F}_r^i) + 3\alpha \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) + \alpha\, \overline{\mathrm{lb}}(\tilde{F}_r^i) \\
&= 3\alpha\, \overline{\mathrm{lb}}(\tilde{F}_r^i) + 3\alpha \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) \\
&\leq 3\alpha\, \overline{\mathrm{lb}}(\tilde{F}_r^i) + 3\alpha \sum_{j \in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) + 3\alpha\, \overline{\mathrm{lb}}(T_i^* \setminus \tilde{F}_r^i) \\
&= 3\alpha\, \overline{\mathrm{lb}}(T^*)\,.
\end{aligned}$$

□

It remains to verify the increase of the "potential" function as stated in (4.5). By the definition of the new initialization, the increase is

$$\overline{\mathrm{lb}}(T^*)^2 - \sum_{j\in I} \overline{\mathrm{lb}}(T_j^*)^2 \,.$$

Let us concentrate on the first term:

$$\begin{aligned}\overline{\mathrm{lb}}(T^*)^2 &= \left(\overline{\mathrm{lb}}(\tilde{F}_r^i) + \overline{\mathrm{lb}}(T_i^* \setminus \tilde{F}_r^i) + \sum_{j\in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i)\right)^2 \\ &\geq \overline{\mathrm{lb}}(\tilde{F}_r^i)\left(\overline{\mathrm{lb}}(\tilde{F}_r^i) + \overline{\mathrm{lb}}(T_i^* \setminus \tilde{F}_r^i) + \sum_{j\in S} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i)\right).\end{aligned}$$

By the selection of S, the expression inside the parenthesis is at least

$$\begin{aligned}\overline{\mathrm{lb}}(\tilde{F}_r^i) + \overline{\mathrm{lb}}(T_i^* \setminus \tilde{F}_r^i) + \frac{1}{3}\sum_{j\in I\setminus\{i\}} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) - \overline{\mathrm{lb}}(T_i^*) \\ \geq \overline{\mathrm{lb}}(\tilde{F}_r^i) + \frac{1}{3}\sum_{j\in I} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i) - \overline{\mathrm{lb}}(T_i^*).\end{aligned}$$

Using $\overline{\mathrm{lb}}(T_i^*) < \overline{\mathrm{lb}}(\tilde{F}_r^i)/3$, we can further lower-bound this expression by

$$\frac{1}{3}\,\overline{\mathrm{lb}}(\tilde{F}_r^i) + \frac{1}{3}\left(\overline{\mathrm{lb}}(\tilde{F}_r^i) + \sum_{j\in I} \overline{\mathrm{lb}}(T_j^* \setminus \tilde{F}_r^i)\right) \geq \frac{1}{3}\,\overline{\mathrm{lb}}(\tilde{F}_r^i) + \frac{1}{3}\sum_{j\in I} \overline{\mathrm{lb}}(T_j^*).$$

Finally, as $\overline{\mathrm{lb}}(T_j^*) \leq \overline{\mathrm{lb}}(T_i^*)$ for all $j \in I$, we have

$$\begin{aligned}\overline{\mathrm{lb}}(T^*)^2 - \sum_{j\in I} \overline{\mathrm{lb}}(T_j^*)^2 &\geq \overline{\mathrm{lb}}(T^*)^2 - \overline{\mathrm{lb}}(T_i^*)\sum_{j\in I} \overline{\mathrm{lb}}(T_j^*) \\ &\geq \overline{\mathrm{lb}}(\tilde{F}_r^i)\left(\frac{1}{3}\,\overline{\mathrm{lb}}(\tilde{F}_r^i) + \frac{1}{3}\sum_{j\in I} \overline{\mathrm{lb}}(T_j^*)\right) - \overline{\mathrm{lb}}(T_i^*)\sum_{j\in I} \overline{\mathrm{lb}}(T_j^*) \\ &= \frac{1}{3}\,\overline{\mathrm{lb}}(\tilde{F}_r^i)^2 + \underbrace{\left(\frac{\overline{\mathrm{lb}}(\tilde{F}_r^i)}{3} - \overline{\mathrm{lb}}(T_i^*)\right)}_{>0}\sum_{j\in I} \overline{\mathrm{lb}}(T_j^*) \\ &\geq \frac{1}{3}\left(\varepsilon\frac{\mathrm{lb}(\bar{B})}{n}\right)^2 = \frac{\varepsilon^2}{3n^2}\,\mathrm{lb}(\bar{B})^2\,,\end{aligned}$$

which completes the proof of Lemma 4.12. □

Chapter 5

Obtaining Structured Instances

In Chapter 4 we have reduced the problem of approximating ATSP to that of designing algorithms for Subtour Partition Cover. Our approach for dealing with general instances is to first simplify their structure and then to solve Subtour Partition Cover on the resulting structured instances. In this chapter, we show that we can obtain very structured instances by only increasing the approximation guarantee by a constant factor. In Chapter 6 we will then solve Subtour Partition Cover on those instances.

5.1 Paths in tight sets

An instance $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$ will be fixed throughout this section. We say that a path P *traverses* a set S if both endpoints of P are in $V \setminus S$ and P contains at least one vertex in S. We now exhibit properties of paths traversing tight sets. In particular, we show that the strongly connected components of a tight set S enjoy a nice path-like structure as depicted in Figure 5.1.

Recall from Section 3.2 that a set S of vertices is tight if $x(\delta(S)) = 2$. Moreover, S_{in} and S_{out} denote those vertices of S that have an incoming edge from outside of S and those that have an outgoing edge to outside of S, respectively.

Lemma 5.1

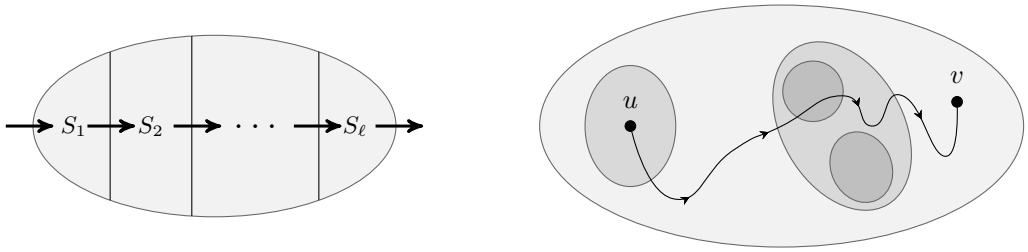
For a tight set $S \subsetneq V$ we have the following properties:

(a) *Every path from a vertex $u \in S_{in}$ to a vertex $v \in S_{out}$ (and thus every path traversing S) visits every strongly connected component of S.*

(b) *For every $u \in S_{in}$ and $v \in S$ there is a path from u to v inside S. The same holds for every $u \in S$ and $v \in S_{out}$.*

Proof.
We remark that *(a)* can also be seen to follow from the "τ-narrow cut" structure as introduced in [AKS15] by setting $\tau = 0$. We give a different proof that we find simpler for our setting.

Let $S_1, S_2, \ldots, S_\ell$ be the vertex sets of the strongly connected components of S, indexed using a topological ordering. We thus have that each subgraph $G[S_i]$ is strongly connected and that there is no edge from a vertex in S_i to a vertex in S_j if $i > j$.



(a) (b)

Figure 5.1: (a) The structure of a tight set S with strongly connected components $S_1, \ldots, S_\ell$. Every path traversing S enters at a vertex in $S_{\text{in}} \subseteq S_1$, then visits all strongly connected components, which form a "path" structure, before it exits from a vertex in $S_{\text{out}} \subseteq S_\ell$.
(b) The structure of the path P from $u \in S_{\text{in}}$ to $v \in S_{\text{out}}$ for a tight set S as given by Lemma 5.2. The path crosses the set that contains u but not v once and it crosses the sets of $\mathcal{L}$ that are disjoint from $\{u, v\}$ at most twice.

By the above, we must have $\delta^-(S_1) \subseteq \delta^-(S)$. Moreover, since $x^\star(\delta^-(S_1)) \geq 1 = x^\star(\delta^-(S))$, we can further conclude that $\delta^-(S_1) = \delta^-(S)$ (recall that all edges have positive $x^\star$-value) and $x^\star(\delta^-(S_1)) = x^\star(\delta^+(S_1)) = 1$ (i.e., S_1 is a tight set).

Similarly, we can show by induction on $k \geq 2$ that

$$\delta^-(S_k) = \delta^+(S_{k-1}) \quad \text{and} \quad x^\star(\delta^-(S_k)) = x^\star(\delta^+(S_k)) = 1.$$

To see this, note that $\delta^-(S_k) \subseteq \delta^-(S) \cup \bigcup_{i<k} \delta^+(S_i)$. However, $\delta^-(S) = \delta^-(S_1)$ (which is disjoint from $\delta^-(S_k)$), and for $i < k-1$, the induction hypothesis gives that $\delta^+(S_i) = \delta^-(S_{i+1})$ (which is also disjoint from $\delta^-(S_k)$). The only term left in the union is $i = k-1$ and so $\delta^-(S_k) \subseteq \delta^+(S_{k-1})$. Moreover, $1 \leq x^\star(\delta^-(S_k)) \leq x^\star(\delta^+(S_{k-1})) = 1$, which implies the statement for k.

Finally, we have that $\delta^+(S_\ell) = \delta^+(S)$. To recap, all incoming edges of S are into S_1, the set of outgoing edges of every component is the set of incoming edges of the next one, and all outgoing edges of S are from S_ℓ. This shows *(a)*, i.e., that every path traversing S needs to enter through S_1, exit through S_ℓ, and pass through every component on the way.

Finally, *(b)* follows because $S_{\text{in}} \subseteq S_1$ (similarly $S_{\text{out}} \subseteq S_\ell$), each two consecutive components are connected by an edge, and each component is strongly connected. □

Lemma 5.2

Let $S \subsetneq V$ be a nonempty set such that $\mathcal{L} \cup \{S\}$ is a laminar family. Suppose $u, v \in S$ are two vertices such that there is a path from u to v inside S. Then we can in polynomial time find a path P from u to v inside S that crosses every set in $\mathcal{L}$ at most twice. Thus, the path satisfies $w(P) \leq \sum_{R \in \mathcal{L}:\ R \subsetneq S} 2 \cdot y_R = \text{value}(S)$.

In addition, if $u \in S_{in}$ or $v \in S_{out}$, then P crosses every tight set $R \in \mathcal{L}$, $R \subsetneq S$ at most $2 - |R \cap \{u, v\}|$ times. Thus, it satisfies $w(P) \leq \sum_{R \in \mathcal{L}:\ R \subsetneq S} (2 - |R \cap \{u, v\}|) \cdot y_R$.

Proof.
Since $\mathcal{L} \cup \{S\}$ is a laminar family, any path inside S only crosses those sets $R \in \mathcal{L}$ that have $R \subsetneq S$. Now, to prove both statements, it is enough to find, in polynomial time, a path P inside

S that for each $R \in \mathcal{L}$ with $R \subsetneq S$ satisfies

$$|P \cap \delta(R)| \leq \begin{cases} 2 & \text{if } |R \cap \{u,v\}| = 0, \\ 1 & \text{if } |R \cap \{u,v\}| = 1, \\ 2 & \text{if } |R \cap \{u,v\}| = 2, \\ 0 & \text{if } |R \cap \{u,v\}| = 2 \text{ and } u \in S_{\text{in}} \text{ or } v \in S_{\text{out}}. \end{cases}$$

The algorithm for finding P starts with any path P from u to v inside S. Such a path is guaranteed to exist by the assumptions of the lemma and can be easily found in polynomial time. Now, while P does not satisfy the above conditions, select a set $R \in \mathcal{L}$ of *maximum* cardinality that violates one of the above conditions. We remark that the selected set R is tight since $R \in \mathcal{L}$. Therefore Lemma 5.1*(b)* implies that there is a path from any $u' \in R$ to any $v' \in R$ inside R if $u' \in R_{\text{in}}$ or $v' \in R_{\text{out}}$. Using this, the algorithm now modifies P depending on which of the above conditions is violated:

Case 1: $|R \cap \{u,v\}| = 0$. Let u' be the first vertex visited by P in R and let v' be the last. Then $u' \in R_{\text{in}}$ and $v' \in R_{\text{out}}$, which implies by Lemma 5.1*(b)* that there is a path Q from u' to v' inside R. We update P by letting Q replace the segment of P from u' to v'. This ensures that the set R is no longer violated, since the path P now only enters and exits R once.

Case 2: $|R \cap \{u,v\}| = 1$. This case is similar to the previous one. Suppose that $u \in R$ and $v \notin R$ (the other case is analogous). Let v' be the last vertex visited by P in R. Then $v' \in R_{\text{out}}$, and again by Lemma 5.1*(b)* there is a path Q from u to v' inside R. We update P by letting Q replace the segment of P from u to v'. This ensures that the set R is no longer violated, since the path P now only exits R once.

Case 3: $|R \cap \{u,v\}| = 2$**.** Let u' be the first vertex visited by P in R_{in}. By Lemma 5.1*(b)*, there is a path Q from u' to v inside R. We modify P by letting Q replace the segment of P from u' to v. This ensures that the set R is no longer violated, since the path P now only enters and exits R at most once.

Case 4: $|R \cap \{u,v\}| = 2$ and $u \in S_{\text{in}}$ or $v \in S_{\text{out}}$. Suppose that $u \in S_{\text{in}}$ (the case $v \in S_{\text{out}}$ is analogous). Then, as $R \subseteq S$, $R \cap S_{\text{in}} \subseteq R_{\text{in}}$. So, by Lemma 5.1*(b)*, there is a path Q from u to v inside R. We replace P by Q and the set R is no longer violated.

At termination, the above algorithm returns a path satisfying all the desired conditions and thus the lemma. It remains to argue that the algorithm terminates in polynomial time. A laminar family contains at most $2n-1$ sets, so it is easy to efficiently identify a violated set R of maximum cardinality. The algorithm then, in polynomial time, modifies P by simple path computations so that the set R is no longer violated. Moreover, since the modifications are such that new edges are only added within the set R, they may only introduce new violations to sets contained in R – sets of smaller cardinality. It follows, since we always select a violated set of maximum cardinality, that any set R in $\mathcal{L}$ is selected in at most one iteration. Hence, the algorithm runs for at most $|\mathcal{L}| \leq 2n-1$ iterations (and so it terminates in polynomial time). □

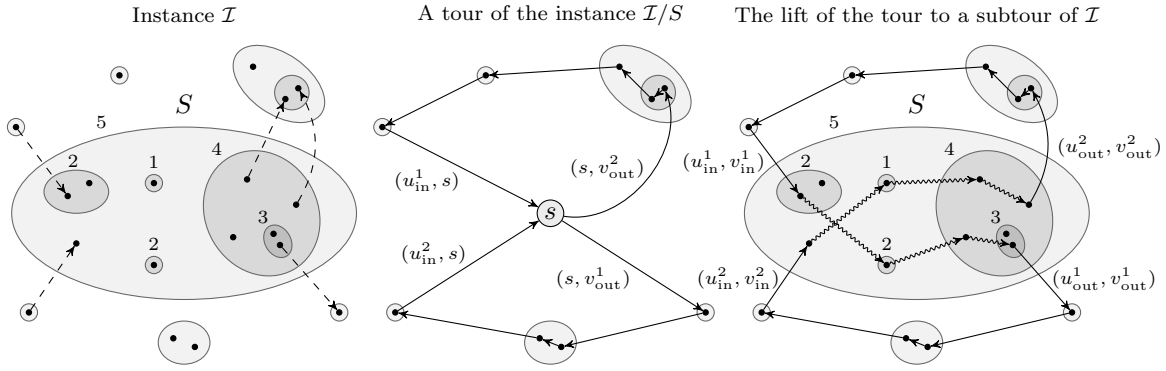


Figure 5.2: An example of the contraction of a tight set S and the lift of a tour. Only y-values of the sets $R \in \mathcal{L} : R \subseteq S$ are depicted. On the left, only edges that have one endpoint in S are shown. These are exactly the edges that are incident to s in the contracted instance. In the center, a tour of $\mathcal{I}/S$ is illustrated, and on the right we depict the lift of that tour.

5.2 Contracting and inducing on a tight set

In this section we generalize two natural graph-theoretic constructions that allow one to decompose the problem of finding a tour with respect to a vertex set S. The first relies on contracting S (see Definition 5.4 in Section 5.2.1) and the second relies on inducing on S (see Definition 5.8 in Section 5.2.2).

5.2.1 Contracting a tight set

Consider an instance $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$. Before defining the contraction of a set $S \in \mathcal{L}$, we need to define the "distance" functions d_S and D_S. For $S \in \mathcal{L}$ and $u, v \in S$, define $d_S(u, v)$ to be the minimum weight of a path inside S from u to v (if no such path exists, $d_S(u, v) = \infty$). We also let

$$D_S(u, v) = \sum_{R \in \mathcal{L}:\ u \in R \subsetneq S} y_R + d_S(u, v) + \sum_{R \in \mathcal{L}:\ v \in R \subsetneq S} y_R \,,$$

which equals $d_S(u, v) + \sum_{R \in \mathcal{L}:\ R \subsetneq S} |R \cap \{u, v\}| \cdot y_R$. We remark that $D_S(u, u)$ might be strictly positive.

The intuition of the definition of D_S is as follows. After contracting S, all sets of the laminar family are still present in the contracted instance, except for the sets strictly contained in S. Now, after finding a tour in the contracted instance, we need to lift it back to a subtour in the original instance. This is done as depicted in Figure 5.2: for each visit of the tour to s (the vertex corresponding to the contraction of S) on the edges $(u^i_{\text{in}}, s), (s, v^i_{\text{out}})$, we obtain a subtour of the original instance by replacing $(u^i_{\text{in}}, s), (s, v^i_{\text{out}})$ by the corresponding edges (i.e., by their preimages) $(u^i_{\text{in}}, v^i_{\text{in}}), (u^i_{\text{out}}, v^i_{\text{out}})$ of G together with the minimum-weight path inside S from v^i_{in} to u^i_{out}. The value $D_S(v^i_{\text{in}}, u^i_{\text{out}})$ is the weight increase incurred by this operation. For example,

in Figure 5.2 we have

$$\underbrace{D_S(v^1_{\text{in}}, u^1_{\text{out}})}_{=22} = \underbrace{\sum_{R\in\mathcal{L}:\ v^1_{\text{in}}\in R\subsetneq S} y_R}_{=2} + \underbrace{d_S(v^1_{\text{in}}, u^1_{\text{out}})}_{=2+2\cdot 2+4+3} + \underbrace{\sum_{R\in\mathcal{L}:\ u^1_{\text{out}}\in R\subsetneq S} y_R}_{=3+4} .$$

Before formally defining the notions of contraction and lift, we state the following useful bound on $D_S(u, v)$.

Fact 5.3

For any $u, v \in S$ with $u \in S_{in}$ or $v \in S_{out}$ we have

$$D_S(u, v) \leq \text{value}(S).$$

Proof.
Lemma 5.1*(b)* says that there is a path from u to v inside S. Select P to be the path from u to v as guaranteed by Lemma 5.2. Since $u \in S_{\text{in}}$ or $v \in S_{\text{out}}$, we have

$$d_S(u, v) \leq w(P) \leq \sum_{R\in\mathcal{L}:\ R\subsetneq S} (2 - |R \cap \{u, v\}|) \cdot y_R$$

and thus

$$D_S(u, v) = d_S(u, v) + \sum_{R\in\mathcal{L}: R\subsetneq S} |R \cap \{u, v\}| \cdot y_R \leq \sum_{R\in\mathcal{L}:\ R\subsetneq S} 2 \cdot y_R = \text{value}(S)\,.$$

□

We now define the notion of contracting a tight set for an ATSP instance. In short, the *contraction* is the instance obtained by performing the classic graph contraction of S, modifying $\mathcal{L}$ to remove the sets contained in S, and increasing the y-value of the new singleton $\{s\}$ corresponding to S so as to become $y_S + 1/2 \max_{u\in S_{\text{in}}, v\in S_{\text{out}}} D_S(u, v)$. This increase is done in order to pay for the maximum possible weight increase incurred when lifting a tour in the contraction back to a subtour in the original instance (as depicted in Figure 5.2, defined in Definition 5.6, and analyzed in Lemma 5.7).

Definition 5.4 (*Contracting a tight set*)

The instance $(G', \mathcal{L}', x', y')$ obtained from $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$ by contracting *$S \in \mathcal{L}$, denoted by $\mathcal{I}/S$, is defined as follows:*

- *The graph G' equals G/S, i.e., the graph obtained from G by contracting S. Let s denote the new vertex of G' that corresponds to the set S.*
- *For each edge $e' \in E(G')$, $x'(e')$ equals $x^\star(e)$, where $e \in E(G)$ is the preimage of e' in G.*[7]
- *The laminar family $\mathcal{L}'$ contains all remaining sets of $\mathcal{L}$:*

$$\mathcal{L}' = \{R \setminus S \cup \{s\} : R \in \mathcal{L}, S \subseteq R\} \cup \{R : R \in \mathcal{L}, S \cap R = \emptyset\}\,.$$

- *The vector y' equals y (via the natural mapping) on all sets but $\{s\}$. For $\{s\}$ we define*

$$y'_s = y_S + \frac{1}{2} \max_{u \in S_{in}, v \in S_{out}} D_S(u, v)\,.$$

We remark that $\mathcal{I}/S$ as defined above is indeed an instance: $\mathcal{L}'$ is a laminar family of tight sets (since for each $R \in \mathcal{L}'$ we have $x'(\delta(R')) = x(\delta(R))$, where R is the preimage of R' in $\mathcal{L}$ via the natural mapping), $y'_R \geq 0$ is defined only for $R \in \mathcal{L}'$, and x' is a feasible solution to the Held-Karp relaxation for G' that is strictly positive on all edges.

The way we defined the new dual weight y'_s implies the natural property that the value of the linear programming solution does not increase after contracting a tight set:

Fact 5.5

$\text{value}(\mathcal{I}/S) = \text{value}(\mathcal{I}) - (\text{value}_{\mathcal{I}}(S) - \max_{u \in S_{in}, v \in S_{out}} D_S(u, v)) \leq \text{value}(\mathcal{I})$.

Proof.
By definition,

$$\begin{aligned}
\text{value}(\mathcal{I}/S) = 2 \cdot \sum_{R \in \mathcal{L}'} y'_R &= 2 \cdot y'_s + 2 \cdot \sum_{R \in \mathcal{L}:\ R \not\subseteq S} y_R \\
&= \max_{u \in S_{\text{in}}, v \in S_{\text{out}}} D_S(u, v) + 2 \cdot y_S + 2 \cdot \sum_{R \in \mathcal{L}:\ R \not\subseteq S} y_R \\
&= \max_{u \in S_{\text{in}}, v \in S_{\text{out}}} D_S(u, v) + 2 \cdot \sum_{R \in \mathcal{L}} y_R - 2 \cdot \sum_{R \in \mathcal{L}:\ R \subsetneq S} y_R \\
&= \max_{u \in S_{\text{in}}, v \in S_{\text{out}}} D_S(u, v) + \text{value}(\mathcal{I}) - \text{value}_{\mathcal{I}}(S)
\end{aligned}$$

and so the equality of the statement holds. Finally, the inequality of the statement follows from Fact 5.3, which implies that $\max_{u \in S_{\text{in}}, v \in S_{\text{out}}} D_S(u, v) \leq \text{value}_{\mathcal{I}}(S)$. □

Having defined the contraction of a tight set $S \in \mathcal{L}$, we define the aforementioned operation of lifting a tour of the contracted instance $\mathcal{I}/S$ to a subtour in the original instance $\mathcal{I}$. When

[7] Recall that for notational convenience we allow parallel edges in G/S and therefore the preimage is uniquely defined.

considering a tour (or a subtour), we order the edges according to an arbitrary but fixed Eulerian walk. This allows us to talk about consecutive edges.

Definition 5.6

For a tour T of $\mathcal{I}/S$, we define its lift *to be the subtour of $\mathcal{I}$ obtained from T by replacing each consecutive pair $(u_{in}, s), (s, v_{out})$ of incoming and outgoing edges incident to s by their preimages (u_{in}, v_{in}) and (u_{out}, v_{out}) in G, together with a minimum-weight path from v_{in} to u_{out} inside S.*[8]

See Figure 5.2 for an illustration. It follows that the lift is a subtour (i.e., an Eulerian multiset of edges that forms a single component), because we added paths between consecutive edges in the tour of $\mathcal{I}/S$. However, the lift is usually not a tour of the instance $\mathcal{I}$, as it is not guaranteed to visit all the vertices in S. To extend the lift to a tour, we use the concept of inducing on the tight set S, which we introduce in Section 5.2.2.

We complete this section by bounding the weight of the lift of T.

Lemma 5.7

Let T be a tour of the instance $\mathcal{I}/S$. Then the lift F of T satisfies $w_{\mathcal{I}}(F) \leq w_{\mathcal{I}/S}(T)$.

Proof.
Consider the tour T and let $(u_{\text{in}}^{(1)}, s), (s, v_{\text{out}}^{(1)}), \ldots, (u_{\text{in}}^{(k)}, s), (s, v_{\text{out}}^{(k)})$ be the edges that T uses to visit the vertex s (which corresponds to the contracted set S). That is, $(u_{\text{in}}^{(i)}, s)$ and $(s, v_{\text{out}}^{(i)})$ are the incoming and outgoing edge of the i-th visit of T to s. By the definition of contraction, we can write the weight of T as

$$\begin{aligned} w_{\mathcal{I}/S}(T) &= \sum_{R \in \mathcal{L}:\ R \not\subseteq S} \alpha_R y_R + 2k \cdot y'_s \\ &= \sum_{R \in \mathcal{L}:\ R \not\subseteq S} \alpha_R y_R + k \cdot \left(2 \cdot y_S + \max_{u \in S_{\text{in}}, v \in S_{\text{out}}} D_S(u, v) \right), \end{aligned}$$

where $\alpha_R = |\delta(R) \cap T|$. We now compare this weight to that of the lift F. Let $(u_{\text{in}}^{(i)}, v_{\text{in}}^{(i)})$ and $(u_{\text{out}}^{(i)}, v_{\text{out}}^{(i)})$ be the edges of G that are the preimages of $(u_{\text{in}}^{(i)}, s)$ and $(s, v_{\text{out}}^{(i)})$. The lift F is obtained from T by replacing $(u_{\text{in}}^{(i)}, s), (s, v_{\text{out}}^{(i)})$ by $(u_{\text{in}}^{(i)}, v_{\text{in}}^{(i)}), (u_{\text{out}}^{(i)}, v_{\text{out}}^{(i)})$ and adding a minimum-weight path inside S from $v_{\text{in}}^{(i)}$ to $u_{\text{out}}^{(i)}$. So F crosses every $R \in \mathcal{L} : R \not\subseteq S$ the same number of times α_R as T. To bound the weight incurred by crossing the tight sets "inside" S, note that the i-th visit to the set S incurs a weight from crossing sets $R \in \mathcal{L} : R \subseteq S$ that equals

$$2y_S + \sum_{R \in \mathcal{L}:\ v_{\text{in}}^{(i)} \in R \subsetneq S} y_R + d_S(v_{\text{in}}^{(i)}, u_{\text{out}}^{(i)}) + \sum_{R \in \mathcal{L}:\ u_{\text{out}}^{(i)} \in R \subsetneq S} y_R = 2y_S + D_S(v_{\text{in}}^{(i)}, u_{\text{out}}^{(i)}) .$$

[8] We remark that it is not crucial that the minimum-weight path from v_{in} to u_{out} be selected to be inside S; a minimum-weight path without this restriction would also work. We have chosen this definition as we find it more intuitive and it simplifies some arguments.

Hence

$$\begin{aligned} w_{\mathcal{I}}(F) &= \sum_{R\in\mathcal{L}:\ R\not\subseteq S} \alpha_R y_R + \sum_{i=1}^{k} \left(2\cdot y_S + D_S(v_{\text{in}}^{(i)}, u_{\text{out}}^{(i)})\right) \\ &\leq \sum_{R\in\mathcal{L}:\ R\not\subseteq S} \alpha_R y_R + \sum_{i=1}^{k} \left(2\cdot y_S + \max_{u\in S_{\text{in}}, v\in S_{\text{out}}} D_S(u,v)\right) \\ &= w_{\mathcal{I}/S}(T)\,. \end{aligned}$$

□

5.2.2 Inducing on a tight set

In this section we introduce our notion of induced instances. This concept will be used for completing a lift of a tour of a contracted instance into a tour of the original instance (see Definition 5.10 of "contractible" below). Inducing on a tight set S is similar to contracting its complement $V\setminus S$ into a single vertex $\bar{s}$ (see Definition 5.4), though the resulting laminar family and dual values are somewhat different: namely, we let $y'_{\bar{s}} = \text{value}(S)/2$ and we remove S (as well as all supersets of S) from $\mathcal{L}'$. The intuitive reason for the setting of $y'_{\bar{s}}$ is that each visit to $\bar{s}$ should pay for the most expensive shortest paths in the strongly connected components of S (see Figure 5.3 and the proof of Lemma 5.11).

We remark that the notion of inducing on S for ATSP instances differs compared to the graph obtained by inducing on S (in the usual graph-theoretic sense), as here we also have the vertex $\bar{s}$ corresponding to the contraction of the vertices not in S. This is needed to make sure that we obtain an ATSP instance (in particular, that we obtain a feasible solution x' to the linear programming relaxation).

Definition 5.8

The instance $(G', \mathcal{L}', x', y')$ obtained from $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$ by inducing *on a tight set $S \in \mathcal{L}$, denoted by $\mathcal{I}[S]$, is defined as follows:*

- *The graph G' equals $G/\bar{S}$, i.e., the graph obtained from G by contracting $\bar{S} = V\setminus S$. Let $\bar{s}$ denote the new vertex of G' that corresponds to the set $\bar{S}$.*
- *For each edge $e' \in E(G')$, $x'(e')$ equals $x^\star(e)$, where $e \in E(G)$ is the preimage of e' in G.*[9]
- *The laminar family $\mathcal{L}'$ contains $\{\bar{s}\}$ and all sets that are strict subsets of S:*

$$\mathcal{L}' = \{R \in \mathcal{L} : R \subsetneq S\} \cup \{\{\bar{s}\}\}\,.$$

- *The vector y' equals y on the sets common to $\mathcal{L}'$ and $\mathcal{L}$. For the new set $\{\bar{s}\}$ we define $y'_{\bar{s}} = \text{value}(S)/2$.*

We remark that $\mathcal{I}[S]$ in an instance: $\mathcal{L}'$ is a laminar family of tight sets, $y'_R \geq 0$ is defined only for $R \in \mathcal{L}'$, and x' is a feasible solution to the Held-Karp relaxation for G' that is strictly positive on all edges.

[9] We again recall that parallel edges are allowed in $G/\bar{S}$, and thus the preimage of an edge is uniquely defined.

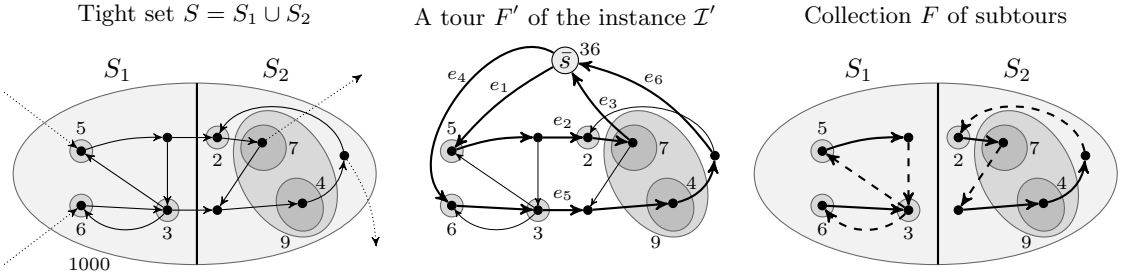


Figure 5.3: In the left figure we depict a tight set $S \in \mathcal{L}$ with two strongly connected components S_1 and S_2. The induced instance (center figure) is obtained by contracting $\bar{S} = V \setminus S$ into a vertex $\bar{s}$ and removing the tight set S from $\mathcal{L}$. The solid edges are paths and edges of a tour in the induced instance. In Lemma 5.11 we obtain a collection of subtours in the original instance (right figure) by adding the dashed paths, resulting in a tour of each strongly connected component.

As for the value of $\mathcal{I}[S]$, it is comprised of the y-values of sets strictly inside S, which contribute value(S), and that of $\{\bar{s}\}$, which also contributes $2y'_{\{\bar{s}\}} = \text{value}(S)$. Thus we have

Fact 5.9

value$(\mathcal{I}[S]) = 2\,\text{value}(S)$.

As alluded to above, we will use the instance $\mathcal{I}[S]$ to find a collection F of subtours in the original instance $\mathcal{I}$ such that F plus a lift of a tour in $\mathcal{I}/S$ form a tour of the instance $\mathcal{I}$. We say that such a set F makes S *contractible*:

Definition 5.10

We say that $S \in \mathcal{L}$ is contractible *with respect to a collection $F \subseteq E$ of subtours (i.e., F is an Eulerian multiset of edges) if the lift of any tour of $\mathcal{I}/S$ plus the edge set F is a tour of $\mathcal{I}$.*

As an example, if F were a subtour visiting every vertex of S, then S would be contractible with respect to F. (Of course, such a subtour F can only exist if S is strongly connected.) The following lemma shows that, in general, it is sufficient to find a tour of $\mathcal{I}[S]$ in order to make S contractible (see also Figure 5.3).

Lemma 5.11

Given a tour T of $\mathcal{I}[S]$, we can in polynomial time find a collection $F \subseteq E$ of subtours such that S is contractible with respect to F and $w_{\mathcal{I}}(F) \leq w_{\mathcal{I}[S]}(T)$.

Proof.
Let $S_1, \ldots, S_\ell$ be the strongly connected components of S indexed using a topological ordering. We will use T to obtain a low-weight tour F_i inside each S_i, and define F to be the union of these tours. Then S is contractible with respect to F. Indeed, the lift of any tour of $\mathcal{I}/S$ must contain a path traversing S, and any such path visits every connected component by Lemma 5.1*(a)*.

Let us fix one component S_i. We obtain the tour F_i of S_i by reproducing the movements of T inside S_i (recall that we think of T as a cyclically ordered Eulerian walk). More precisely, we retain those edges of T that are inside S_i and, every time T exits S_i on an edge $(u_{\text{out}}, v_{\text{out}}) \in$

$\delta^+(S_i)$ and then returns to S_i on an edge $(u_{\text{in}}, v_{\text{in}}) \in \delta^-(S_i)$, we also insert a minimum-weight path from u_{out} to v_{in} inside S_i. Such a path exists because S_i is strongly connected. (This step corresponds to adding the dashed paths in Figure 5.3.) Then we set $F = F_1 \cup \ldots \cup F_\ell$.

It remains to show that F has low weight, i.e., that $w_{\mathcal{I}}(F) \leq w_{\mathcal{I}[S]}(T)$. For this, let k be the number of times the tour T visits the auxiliary vertex $\bar{s}$. The weight incurred by every such visit is at least $2y'_{\bar{s}} = \text{value}(S)$ (since the set $\{\bar{s}\}$ is crossed twice in each visit). Thus we have

$$w_{\mathcal{I}[S]}(T) \geq k \cdot \text{value}(S) + \sum_{i=1}^{\ell} w_{\mathcal{I}[S]}(T \cap E(S_i)) = k \cdot \text{value}(S) + \sum_{i=1}^{\ell} w_{\mathcal{I}}(T \cap E(S_i)) \,.$$

On the other hand, each tour F_i consists of all those edges of T that are inside S_i, as well as k shortest paths between some pairs of vertices in S_i. Indeed, if T makes k visits to the auxiliary vertex $\bar{s}$, then we add exactly k paths inside S_i due to the path-like structure of the strongly connected components of a tight set S (Lemma 5.1). We now have the following claim, which allows us to bound the length of these paths by applying Lemma 5.2.

Claim. *For each $i = 1, \ldots, \ell$, $\mathcal{L} \cup \{S_i\}$ is a laminar family.*

Proof.
Suppose toward a contradiction that $\mathcal{L} \cup \{S_i\}$ is not a laminar family. Then there must be a set $R \in \mathcal{L}$ such that $R \setminus S_i, S_i \setminus R$, and $S \cap R$ are all nonempty. Furthermore, since $\mathcal{L}$ is a laminar family and $S_i \subseteq S \in \mathcal{L}$, we must have $R \subsetneq S$. We can thus partition R into the three sets

$$R_{<i} = R \cap (S_1 \cup \cdots \cup S_{i-1}), \qquad R_i = R \cap S_i, \qquad R_{>i} = R \cap (S_{i+1} \cup \cdots \cup S_\ell) \,.$$

In words, $R_{<i}$ is the part of R that intersects vertices of the strongly connected components that are ordered topologically before S_i. Similarly, $R_{>i}$ is the part of R that intersects vertices of the strongly connected components that are ordered topologically after S_i. Note that since R is not contained in S_i, we have that either $R_{<i}$ or $R_{>i}$ is nonempty. We suppose $R_{<i} \neq \emptyset$ (the case $R_{>i} \neq \emptyset$ is analogous).

As x is a feasible solution to the Held-Karp relaxation for G, we have $x(\delta^-(R_{<i})) \geq 1$. Moreover, since $\delta(R_i \cup R_{>i}, R_{<i}) = \emptyset$ due to the topological ordering, we have $\delta^-(R_{<i}) \subseteq \delta^-(R)$ and thus

$$1 = x(\delta^-(R)) \geq x(\delta^-(R_{<i})) + x(\delta(S_i \setminus R_i, R_i)) \geq 1 + x(\delta(S_i \setminus R_i, R_i)) \,,$$

where the first equality follows since $R \in \mathcal{L}$ is a tight set. However, this is a contradiction because $x(\delta(S_i \setminus R_i, R_i)) > 0$; that holds since $S_i \setminus R_i = S_i \setminus R \neq \emptyset$ and $R_i \neq \emptyset$, S_i is a strongly connected component, and G only contains edges with strictly positive x-value. $\diamond$

By the above claim, we can apply Lemma 5.2 to obtain that a shortest path between two vertices inside S_i has weight at most $\text{value}(S_i)$. Recall that F_i consists of all those edges of T that are inside S_i, as well as k shortest paths between some pairs of vertices in S_i. Therefore,

the weight of F is

$$\begin{aligned} w_{\mathcal{I}}(F) &= \sum_{i=1}^{\ell} w_{\mathcal{I}}(F_i) \\ &\leq \sum_{i=1}^{\ell} [k \cdot \text{value}(S_i) + w_{\mathcal{I}}(T \cap E(S_i))] \\ &\leq k \cdot \text{value}(S) + \sum_{i=1}^{\ell} w_{\mathcal{I}}(T \cap E(S_i)) \\ &\leq w_{\mathcal{I}[S]}(T)\,, \end{aligned}$$

as required. □

5.3 Reduction to irreducible instances

In this section we reduce the problem of approximating ATSP on general (laminarly-weighted) instances to that of approximating ATSP on irreducible instances. Specifically, Theorem 5.14 says that any approximation algorithm for irreducible instances can be turned into an algorithm for general instances while losing only a constant factor in the approximation guarantee.

We now define the notions of *reducible sets* and *irreducible instances*. The intuition behind them is as follows. The operations of contracting and inducing on a tight set S introduced in the last section naturally lead to the following recursive algorithm:

1. Select a tight set $S \in \mathcal{L}$.
2. Find a tour T_S in the induced instance $\mathcal{I}[S]$. Via Lemma 5.11, T_S yields a set F_S that makes S contractible.
3. Recursively find a tour T in the contraction $\mathcal{I}/S$.
4. Output F_S plus the lift of T.

For this scheme to yield a good approximation guarantee, we need to ensure that we can find a good approximate tour T_S in $\mathcal{I}[S]$ and that contracting the set S results in a "significant" decrease in the value of the LP solution. If it does, we refer to the set S as *reducible*:

Definition 5.12

We say that a set $S \in \mathcal{L}$ is reducible *if*

$$\max_{u \in S_{in}, v \in S_{out}} D_S(u, v) < \delta \cdot \text{value}(S)\,,$$

otherwise we say that S is irreducible. *We also say that the instance $\mathcal{I}$ is* irreducible *if no set $S \in \mathcal{L}$ is reducible.*

We will use the value $\delta = 0.78$; however, we keep it as a parameter to exhibit the dependence of the approximation ratio on this value.

Note that singleton sets are never reducible. Moreover, we have the following observation:

Fact 5.13

Consider an instance $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$ and a set $S \in \mathcal{L}$. If every set $R \in \mathcal{L} : R \subsetneq S$ is irreducible, then $\mathcal{I}[S]$ is irreducible. In particular, if $\mathcal{I}$ is an irreducible instance, then $\mathcal{I}[S]$ is irreducible for every $S \in \mathcal{L}$.

Proof.
Let $\mathcal{I}[S] = (G', \mathcal{L}', x', y')$. By definition, $\mathcal{L}' = \{R \in \mathcal{L} : R \subsetneq S\} \cup \{\{\bar{s}\}\}$. Clearly, the singleton set $\{\bar{s}\}$ is irreducible. Now consider a set $R \in \mathcal{L} : R \subsetneq S$; we need to show that R is irreducible in $\mathcal{I}[S]$. Note that R is also present in $\mathcal{I}$ and that the sets $\{Q \in \mathcal{L} : Q \subsetneq R\}$ and $\{Q \in \mathcal{L}' : Q \subsetneq R\}$ are identical. This implies that the distance function D_R is identical in the instances $\mathcal{I}$ and $\mathcal{I}[S]$. Moreover, the sets R_{in} and R_{out} are also the same in the two instances. Therefore, as R is irreducible in $\mathcal{I}$ by assumption, we have that R is also irreducible in $\mathcal{I}[S]$. □

The above fact implies that if we select $S \in \mathcal{L}$ to be a *minimal* reducible set, then the instance $\mathcal{I}[S]$ is irreducible. Hence, we only need to be able to find an approximate tour T_S for *irreducible* instances (in Step 2 of the above recursive algorithm). This is the idea behind the following theorem, and its proof is based on formally analyzing the aforementioned approach.

Theorem 5.14

Let $\mathcal{A}$ be a polynomial-time ρ-approximation algorithm for irreducible instances. Then there is a polynomial-time $\frac{2\rho}{1-\delta}$-approximation algorithm for general instances.

Proof.
Consider a general instance $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$. If it is irreducible, we can simply return the result of a single call to $\mathcal{A}$. So assume that $\mathcal{I}$ is not irreducible, i.e., that $\mathcal{L}$ contains a reducible set. Let $S \in \mathcal{L}$ be a minimal (inclusion-wise) reducible set, i.e., one such that all subsets $R \in \mathcal{L} : R \subsetneq S$ are irreducible.

We will work with the induced instance $\mathcal{I}[S]$. Recall that $\text{value}(\mathcal{I}[S]) = 2\,\text{value}(S)$ (Fact 5.9). Moreover, $\mathcal{I}[S]$ is irreducible by Fact 5.13. We can therefore use $\mathcal{A}$ to find a tour T_S of $\mathcal{I}[S]$. Since $\mathcal{A}$ is a ρ-approximation algorithm, we have

$$w_{\mathcal{I}[S]}(T_S) \leq \rho \cdot \text{value}(\mathcal{I}[S]) = 2\rho\,\text{value}(S)\,.$$

Next, we invoke the algorithm of Lemma 5.11 to obtain a collection $F_S \subseteq E$ of subtours such that S is contractible with respect to F_S and

$$w_{\mathcal{I}}(F_S) \leq w_{\mathcal{I}[S]}(T_S) \leq 2\rho\,\text{value}(S)\,. \tag{5.1}$$

Now we recursively solve the contraction $\mathcal{I}/S$. (This is a smaller instance than $\mathcal{I}$, because $|\mathcal{I}/S| = |\mathcal{I}| - |S| + 1$ and $|S| \geq 2$ since a singleton set S would not have been reducible.) Let T be the tour obtained from the recursive call, and let F be the lift of T to $\mathcal{I}$. We finally return $F_S \cup F$. This is a tour of $\mathcal{I}$, as S is contractible with respect to F_S.

The running time of this algorithm is polynomial since each recursive call consists at most of: one call to $\mathcal{A}$, the algorithm of Lemma 5.11, simple graph operations and one recursive call (for a smaller instance).

Finally, let us show that this is a $\frac{2\rho}{1-\delta}$-approximation algorithm by induction on the instance size. We have

$$\begin{aligned} w_{\mathcal{I}}(F \cup F_S) &= w_{\mathcal{I}}(F) + w_{\mathcal{I}}(F_S) \\ &\leq w_{\mathcal{I}/S}(T) + w_{\mathcal{I}}(F_S) \\ &\leq \frac{2\rho}{1-\delta} \operatorname{value}(\mathcal{I}/S) + 2\rho \operatorname{value}(S) \\ &< \frac{2\rho}{1-\delta} [\operatorname{value}(\mathcal{I}) - (1-\delta) \operatorname{value}(S)] + 2\rho \operatorname{value}(S) \\ &= \frac{2\rho}{1-\delta} \operatorname{value}(\mathcal{I}), \end{aligned}$$

where the first inequality is by Lemma 5.7, the second follows since T is a $\frac{2\rho}{1-\delta}$-approximate solution for $\mathcal{I}/S$ and by (5.1), and the strict inequality is by Fact 5.5 and the reducibility of S. This shows that $F \cup F_S$ is a $\frac{2\rho}{1-\delta}$-approximate solution for $\mathcal{I}$. □

5.4 Backbones and reduction to vertebrate pairs

In this section we further reduce the task of approximating ATSP to that of finding a tour in instances with a *backbone*. For an example of such an instance see the right part of Figure 3.1 on page 23.

Definition 5.15

We say that an instance $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$ and a subtour B form a vertebrate pair *if every $S \in \mathcal{L}$ with $|S| \geq 2$ is visited by B, i.e., $S \cap V(B) \neq \emptyset$. The set B is referred to as the* backbone *of the instance.*

Specifically, the main result of this section, Theorem 5.18, says that an algorithm for vertebrate pairs can be turned into an approximation algorithm for irreducible instances while losing only a constant factor in the guarantee. Combining this with Theorem 5.14 allows us to reduce the problem of approximating ATSP on general instances to that of approximating ATSP on vertebrate pairs.

The proof of Theorem 5.18 is done in two steps. First, in Section 5.4.1, we give an efficient algorithm for finding a *quasi-backbone* B of an irreducible instance – a subtour that visits a large (weighted) fraction of the sets in $\mathcal{L}$. We use the term *quasi-backbone* as B might not visit *all* non-singleton sets, as would be required for a backbone. Then, in Section 5.4.2, we give the reduction to vertebrate pairs via a recursive algorithm (similar to the proof of Theorem 5.14 in the previous section).

5.4.1 Finding a quasi-backbone

We give an efficient algorithm for calculating a low-weight *quasi-backbone* of an irreducible instance.

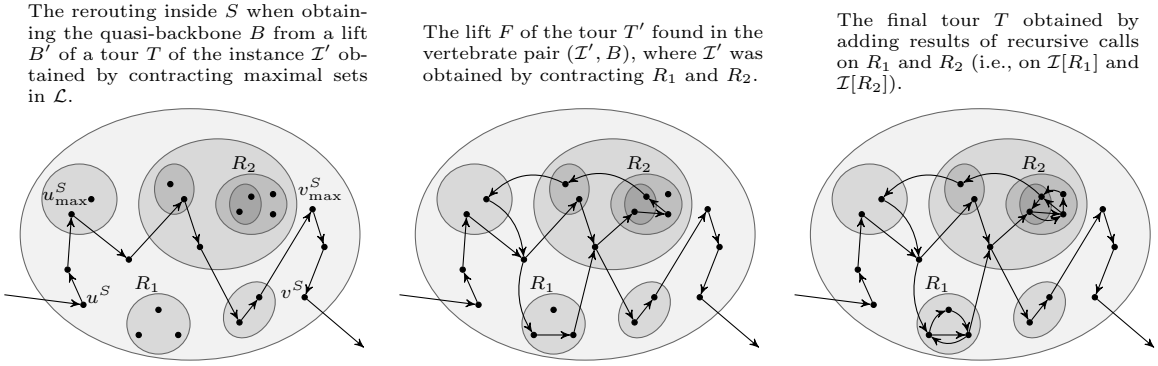


Figure 5.4: An illustration of the steps in the proofs of Lemma 5.17 (left) and Theorem 5.18 (center and right). Only one maximal set $S \in \mathcal{L}$ is shown.

Definition 5.16

For an instance $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$, we call a subtour B a quasi-backbone *if*

$$2 \sum_{S \in \mathcal{L}^*} y_S \leq (1 - \delta)\,\mathrm{value}(\mathcal{I})\,,$$

where $\mathcal{L}^* = \{S \in \mathcal{L} : S \cap V(B) = \emptyset\}$ contains those laminar sets that B does not *visit.*

Recall that $\delta = 0.78$ is the parameter in Definition 5.12 (of irreducible instances). Also note that a backbone is not necessarily a quasi-backbone, as it may not satisfy the above inequality if much y-value is on singleton sets. Recall that $\alpha_{\mathrm{S}} = 18 + \epsilon$ denotes the approximation guarantee for singleton instances as in Corollary 4.4.

Lemma 5.17

There is a polynomial-time algorithm that, given an irreducible instance $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$, constructs a quasi-backbone B such that $w(B) \leq (\alpha_s + 3)\,\mathrm{value}(\mathcal{I})$ and $S \cap V(B) \neq \emptyset$ for every maximal non-singleton set $S \in \mathcal{L}$.

Proof.
Let $\mathcal{L}_{\max}$ be the family of all maximal sets in $\mathcal{L}$. We define $\mathcal{I}'$ to be the instance obtained from $\mathcal{I}$ by contracting all sets in $\mathcal{L}_{\max}$. By Fact 5.5, the LP value does not increase, i.e., $\mathrm{value}(\mathcal{I}') \leq \mathrm{value}(\mathcal{I})$. In $\mathcal{I}'$, all laminar tight sets are singletons, therefore the new instance is a singleton instance and we can use the α_{S}-approximation algorithm (Corollary 4.4) to find a tour T in $\mathcal{I}'$ with $w_{\mathcal{I}'}(T) \leq \alpha_{\mathrm{S}}\,\mathrm{value}(\mathcal{I}') \leq \alpha_{\mathrm{S}}\,\mathrm{value}(\mathcal{I})$.

Now, to obtain a subtour B of the original instance $\mathcal{I}$, we consider the lift B' of T back to $\mathcal{I}$ (see Definition 5.6). The lift B' is a subtour of low weight. Indeed, $w_{\mathcal{I}}(B') \leq w_{\mathcal{I}'}(T) \leq \alpha_{\mathrm{S}}\,\mathrm{value}(\mathcal{I})$ by Lemma 5.7. It also visits every maximal set $S \in \mathcal{L}_{\max}$. However, it might not yet satisfy the inequality of Definition 5.16. We therefore slightly modify the subtour B' to obtain B as follows. For each set $S \in \mathcal{L}_{\max}$:

1. Suppose the first visit[10] to S in the subtour B' arrives at a vertex $u^S \in S_{\mathrm{in}}$ and departs

[10] Recall that the edges of the subtour B' are ordered by an Eulerian walk.

from a vertex $v^S \in S_{\text{out}}$.

2. Replace the segment of B' from u^S to v^S by the union of:
   - a shortest path from u^S to $u^S_{\max}$,
   - a path from $u^S_{\max}$ to $v^S_{\max}$ inside S as given by Lemma 5.2,
   - and a shortest path from $v^S_{\max}$ to v^S,

   where $u^S_{\max} \in S_{\text{in}}$ and $v^S_{\max} \in S_{\text{out}}$ are selected to maximize $D_S(u^S_{\max}, v^S_{\max})$.

See the left part of Figure 5.4 for an illustration. The existence of the second path above is guaranteed by Lemma 5.1*(b)* since $u^S_{\max} \in S_{\text{in}}$. It is clear that the obtained multiset B is a subtour (since B' is a subtour), that it visits every set in $\mathcal{L}_{\max}$, and that the algorithm for finding B runs in polynomial time. It remains to bound the weight of B and to show that B satisfies the property of a quasi-backbone, i.e., the inequality of Definition 5.16.

For the former, note that the weight of B is at most the weight of the lift B' plus the weight of the three paths added for each set $S \in \mathcal{L}_{\max}$. For such a set $S \in \mathcal{L}_{\max}$, the weight of the path from u^S to $u^S_{\max}$ is at most value(S) since there is a path from $u^S \in S_{\text{in}}$ to $u^S_{\max}$ *inside* S by Lemma 5.1*(b)* and such a path can be selected to have weight at most value(S) by Lemma 5.2. By the same argument, we have that the weight of the path from $v^S_{\max}$ to $v^S \in S_{\text{out}}$ is at most value(S). Finally, by applying Lemma 5.2 again, we have that the path added from $u^S_{\max}$ to $v^S_{\max}$ is also bounded by value(S). It follows that

$$w(B) \le w(B') + 3 \cdot \sum_{S \in \mathcal{L}_{\max}} \text{value}(S) \le w(B') + 3\,\text{value}(\mathcal{I}) \le (\alpha_{\text{S}} + 3)\,\text{value}(\mathcal{I})\,,$$

as required. (In the second inequality we used that the sets $S \in \mathcal{L}_{\max}$ are disjoint.)

We proceed to prove that B satisfies the inequality of Definition 5.16. Recall that $\mathcal{L}^* = \{S \in \mathcal{L} : S \cap V(B) = \emptyset\}$ contains those laminar sets that B does not visit. As B visits every $S \in \mathcal{L}_{\max}$ (i.e., $\mathcal{L}_{\max} \cap \mathcal{L}^* = \emptyset$), it is enough to show the following:

Claim. *For every $S \in \mathcal{L}_{max}$ we have*

$$\sum_{R \in \mathcal{L}^*:\ R \subsetneq S} 2y_R \le (1-\delta)\,\text{value}(S)\,.$$

Once we have this claim, the property of a quasi-backbone indeed follows:

$$2\sum_{R \in \mathcal{L}^*} y_R = \sum_{S \in \mathcal{L}_{\max}} \sum_{R \in \mathcal{L}^*:\ R \subsetneq S} 2y_R \le \sum_{S \in \mathcal{L}_{\max}} (1-\delta)\,\text{value}(S) \le (1-\delta)\,\text{value}(\mathcal{I})\,.$$

Proof of Claim.
The intuition behind the claim is that, when forming B, we have added a path P from $u^S_{\max}$ to $v^S_{\max}$. Since S is irreducible, this path P has a large weight. However, it is chosen so that it crosses each set in $\mathcal{L}$ at most twice. Thus it must cross most (weighted by value) sets of $\mathcal{L}$ contained in S.

Now we proceed with the formal proof. As $u^S_{\max} \in S_{\text{in}}$, the path P inside S from $u^S_{\max}$ to $v^S_{\max}$ that we have obtained from Lemma 5.2 crosses every tight set $R \in \mathcal{L}$ at most $2 - |R \cap \{u^S_{\max}, v^S_{\max}\}|$

times. Moreover, P (a subset of B) does not cross any set $R \in \mathcal{L}^*$. Therefore

$$d_S(u^S_{\max}, v^S_{\max}) \le w(P) \le \sum_{R \in \mathcal{L} \setminus \mathcal{L}^*:\ R \subsetneq S} \big(2 - |R \cap \{u^S_{\max}, v^S_{\max}\}|\big) \cdot y_R \,.$$

Furthermore, we have that the quasi-backbone B visits all sets in $\mathcal{L}_{\max}$ and visits both vertices $u^S_{\max}$ and $v^S_{\max}$. Therefore it must visit all sets $R \in \mathcal{L}$ for which $R \cap \{u^S_{\max}, v^S_{\max}\}$ is nonempty; i.e., for all $R \in \mathcal{L}^*$ we have $|R \cap \{u^S_{\max}, v^S_{\max}\}| = 0$. It follows that that the quasi-backbone visits most (weighted by value) laminar sets:

$$\begin{aligned}
\sum_{R \in \mathcal{L} \setminus \mathcal{L}^*:\ R \subsetneq S} 2y_R &= \sum_{R \in \mathcal{L} \setminus \mathcal{L}^*:\ R \subsetneq S} (2 - |R \cap \{u^S_{\max}, v^S_{\max}\}|) \cdot y_R + \sum_{R \in \mathcal{L} \setminus \mathcal{L}^*:\ R \subsetneq S} |R \cap \{u^S_{\max}, v^S_{\max}\}| \cdot y_R \\
&\ge d_S(u^S_{\max}, v^S_{\max}) + \sum_{R \in \mathcal{L}:\ R \subsetneq S} |R \cap \{u^S_{\max}, v^S_{\max}\}| \cdot y_R \\
&= D_S(u^S_{\max}, v^S_{\max}) \\
&\ge \delta \operatorname{value}(S) \,,
\end{aligned}$$

where the last inequality is by the choice of $u^S_{\max}, v^S_{\max}$ and by the irreducibility of S. The claim now follows:

$$\sum_{R \in \mathcal{L}^*:\ R \subsetneq S} 2y_R = \operatorname{value}(S) - \sum_{R \in \mathcal{L} \setminus \mathcal{L}^*:\ R \subsetneq S} 2y_R \le \operatorname{value}(S) - \delta \operatorname{value}(S) = (1 - \delta) \operatorname{value}(S) \,.$$

◇

The proof of the above claim completes the proof of Lemma 5.17. □

5.4.2 Obtaining a vertebrate pair via recursive calls

We now prove the main result of Section 5.4. Recall the notation $\mathrm{lb}_{\mathcal{I}}(\bar{B})$ introduced in (4.2) on page 31.

Theorem 5.18

Let $\mathcal{A}$ be a polynomial-time algorithm that, given a vertebrate pair $(\mathcal{I}', B)$ on vertex set V', returns a tour of $\mathcal{I}'$ with weight at most

$$\kappa \operatorname{value}(\mathcal{I}') + \eta \, \mathrm{lb}_{\mathcal{I}'}(\bar{B}) + w_{\mathcal{I}'}(B)$$

for some $\kappa, \eta \ge 0$. Then there is a polynomial-time ρ-approximation algorithm for ATSP for irreducible instances, where

$$\rho = \alpha_S \frac{\kappa + \eta(1 - \delta) + \alpha_S + 3}{2\delta - 1}.$$

The essence of the theorem is that if we have an algorithm for vertebrate pairs where the approximation factor is bounded by a constant factor of the value of the instance and the weight of the backbone, then this translates to a constant-factor approximation for ATSP in arbitrary

irreducible instances (with no backbone given). The proof of this theorem is somewhat similar to that of Theorem 5.14, in that the algorithm presented here will call itself recursively on smaller instances, as well as invoking the black-box algorithm $\mathcal{A}$ (once per recursive call). The complicated dependence on the parameters is due to the recursive arguments. We will optimize the parameters κ and η in Section 6.2.

Proof.
We briefly discuss the intuition first. Consider an irreducible instance $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$. By Lemma 5.17, we can find a quasi-backbone B – a subtour such that $2\sum_{S\in\mathcal{L}^*} y_S \le (1-\delta)\,\mathrm{value}(\mathcal{I})$, where as before $\mathcal{L}^* = \{S \in \mathcal{L} : S \cap V(B) = \emptyset\}$ contains the laminar sets that the quasi-backbone does *not* visit. This is a small fraction of the entire optimum value($\mathcal{I}$), so we can afford to run an expensive procedure (say, a 2ρ-approximation) on the unvisited sets (using recursive calls) so as to make them contractible. Once we contract all these sets, B will become a backbone in the contracted instance and we will have thus obtained a vertebrate pair, on which the algorithm $\mathcal{A}$ can be applied.[11] See Figure 5.4 for an illustration.

We now formally describe the ρ-approximation algorithm $\mathcal{A}_{\mathrm{irr}}$ for irreducible instances. Given an irreducible instance $\mathcal{I} = (G, \mathcal{L}, x^\star, y)$, it proceeds as follows:

1. Invoke the algorithm of Lemma 5.17 to obtain a quasi-backbone B with $w_{\mathcal{I}}(B) \le (\alpha_{\mathrm{S}} + 3)\,\mathrm{value}(\mathcal{I})$. Denote by $\mathcal{L}^*_{\max}$ the family of maximal (inclusion-wise) *non-singleton* sets in $\mathcal{L}^* = \{S \in \mathcal{L} : S \cap V(B) = \emptyset\}$. (For example, in Figure 5.4, R_1 and R_2 are two such sets.)

2. For each $S \in \mathcal{L}^*_{\max}$, recursively call $\mathcal{A}_{\mathrm{irr}}$ to find a tour T_S in the instance $\mathcal{I}[S]$ (which is irreducible by Fact 5.13). Then use T_S and the algorithm of Lemma 5.11 to find a collection F_S of subtours such that S is contractible with respect to F_S and $w_{\mathcal{I}}(F_S) \le w_{\mathcal{I}[S]}(T_S)$.

3. Let $\mathcal{I}' = (G', \mathcal{L}', x', y')$ be the instance obtained from $\mathcal{I}$ by contracting all the maximal sets $S \in \mathcal{L}^*_{\max}$; let V' denote the contracted ground set. We have that $(\mathcal{I}', B)$ is a vertebrate pair by construction: we have contracted all tight sets that were not visited by B into single vertices, and so B is a backbone of $\mathcal{I}'$. Note that

$$\mathrm{lb}_{\mathcal{I}'}(\bar{B}) = 2\sum_{v\in V'\setminus V(B)} y'_v \le 2\sum_{S\in\mathcal{L}^*} y_S \le (1-\delta)\,\mathrm{value}(\mathcal{I}).$$

   The first inequality follows by the definition of contraction, using Fact 5.3. We can invoke the algorithm $\mathcal{A}$ on the vertebrate pair $(\mathcal{I}', B)$; by the hypothesis of the theorem, it returns a tour T' of $\mathcal{I}'$ such that

$$w_{\mathcal{I}'}(T') \le \kappa\,\mathrm{value}(\mathcal{I}') + \eta(1-\delta)\,\mathrm{value}(\mathcal{I}) + w_{\mathcal{I}'}(B). \tag{5.2}$$

4. Finally, return the tour T consisting of the lift F of T' to $\mathcal{I}$ together with $\bigcup_{S\in\mathcal{L}^*_{\max}} F_S$. (See the center and right parts of Figure 5.4 for an illustration.)

We remark that T is indeed a tour of $\mathcal{I}$ since all sets $S \in \mathcal{L}^*_{\max}$ are contractible with respect to $\bigcup_{S\in\mathcal{L}^*_{\max}} F_S$.

Having described the algorithm, it remains to show that $\mathcal{A}_{\mathrm{irr}}$ runs in polynomial time and that it has an approximation guarantee of ρ.

[11] Note that we never actually find a backbone of the original, uncontracted instance.

For the former, we bound the total number of recursive calls that $\mathcal{A}_{\text{irr}}$ makes. We claim that the total number of recursive calls on input $\mathcal{I} = (G, \mathcal{L}, x, y)$ is at most the cardinality of $\mathcal{L}_{\geq 2} = \{S \in \mathcal{L} : |S| \geq 2\}$. The proof is by induction on $|\mathcal{L}_{\geq 2}|$. For the base case, i.e., when $|\mathcal{L}_{\geq 2}| = 0$, there are no recursive calls since there are no non-singleton sets in $\mathcal{L}^* \subseteq \mathcal{L}$ and so $\mathcal{L}^*_{\max} = \emptyset$. For the inductive step, suppose that $\mathcal{L}^*_{\max} = \{S_1, S_2, \ldots, S_\ell\} \subseteq \mathcal{L}^*$ and so there are ℓ recursive calls in this iteration – on the instances $\mathcal{I}[S_1], \mathcal{I}[S_2], \ldots, \mathcal{I}[S_\ell]$. If we let $\mathcal{L}^i_{\geq 2}$ denote the non-singleton laminar sets of $\mathcal{I}[S_i]$ then, by the definition of inducing on a tight set, for every $R \in \mathcal{L}^i_{\geq 2}$ we have $R \subsetneq S_i$ and $R \in \mathcal{L}_{\geq 2}$. It follows by the induction hypothesis that the total number of recursive calls that $\mathcal{A}_{\text{irr}}$ makes is

$$\ell + \sum_{i=1}^{\ell} |\mathcal{L}^i_{\geq 2}| \leq \ell + |\mathcal{L}_{\geq 2}| - \ell = |\mathcal{L}_{\geq 2}|\,,$$

where the inequality holds because the sets $\mathcal{L}^i_{\geq 2}$ are disjoint and $\mathcal{L}^1_{\geq 2} \cup \mathcal{L}^2_{\geq 2} \cup \cdots \cup \mathcal{L}^\ell_{\geq 2} \subseteq \mathcal{L}_{\geq 2} \setminus \{S_1, S_2, \ldots, S_\ell\}$. Hence, the total number of recursive calls $\mathcal{A}_{\text{irr}}$ makes is $|\mathcal{L}_{\geq 2}| \leq |\mathcal{L}|$, which is at most linear in $|V|$. The fact that $\mathcal{A}_{\text{irr}}$ runs in polynomial time now follows because each call runs in polynomial time. Indeed, the algorithm of Lemma 5.17, the algorithm of Lemma 5.11, and $\mathcal{A}$ all run in polynomial time.

We now complete the proof of the theorem by showing that $\mathcal{A}_{\text{irr}}$ is a ρ-approximation algorithm. From (5.2) and by Lemma 5.7 we have that the weight $w_{\mathcal{I}}(F)$ of the lift F of T' is at most

$$w_{\mathcal{I}'}(T') \leq \kappa\,\text{value}(\mathcal{I}') + \eta(1-\delta)\,\text{value}(\mathcal{I}) + w_{\mathcal{I}'}(B) \leq (\kappa + \eta(1-\delta) + \alpha_{\text{S}} + 3)\,\text{value}(\mathcal{I}),$$

where the second inequality follows by Fact 5.5 and since $w_{\mathcal{I}'}(B) = w_{\mathcal{I}}(B)$ ($\mathcal{I}'$ arises by contracting only sets not visited by B, which preserves the weight of B) and $w_{\mathcal{I}}(B) \leq (\alpha_{\text{S}} + 3)\,\text{value}(\mathcal{I})$.

Now, to show that $w(T) = w(F) + w\left(\bigcup_{S \in \mathcal{L}^*_{\max}} F_S\right) \leq \rho\,\text{value}(\mathcal{I})$, we proceed by induction on the total number of recursive calls. In the base case, when no recursive calls are made, we have $w(T) = w(F) \leq w_{I'}(T') \leq (\kappa + \eta(1-\delta) + \alpha_{\text{S}} + 3)\,\text{value}(\mathcal{I}) \leq \rho\,\text{value}(\mathcal{I})$. For the inductive step, the induction hypothesis yields that for each $S \in \mathcal{L}^*_{\max}$ we have

$$w(F_S) \leq w_{\mathcal{I}[S]}(T_S) \leq \rho\,\text{value}(\mathcal{I}[S]) = 2\rho\,\text{value}(S)\,,$$

where the equality is by Fact 5.9. Hence

$$\begin{aligned} w\left(\bigcup_{S \in \mathcal{L}^*_{\max}} F_S\right) &= \sum_{S \in \mathcal{L}^*_{\max}} w(F_S) \leq \sum_{S \in \mathcal{L}^*_{\max}} 2\rho\,\text{value}(S) \\ &= \sum_{S \in \mathcal{L}^*_{\max}} 2\rho \sum_{R \in \mathcal{L}^*:\ R \subsetneq S} 2y_R \leq 2\rho \sum_{R \in \mathcal{L}^*} 2y_R \leq 2\rho(1-\delta)\,\text{value}(\mathcal{I})\,. \end{aligned}$$

The second equality holds because Lemma 5.17 guarantees that B visits each maximal set of $\mathcal{L}$, which implies that every $R \in \mathcal{L}$ that is a subset of a set $S \in \mathcal{L}^*_{\max}$ is not visited by B, i.e., $R \in \mathcal{L}^*$: thus we have $\sum_{R \in \mathcal{L}:\ R \subsetneq S} 2y_R = \sum_{R \in \mathcal{L}^*:\ R \subsetneq S} 2y_R$ for all $S \in \mathcal{L}^*_{\max}$. The last inequality holds because B is a quasi-backbone of $\mathcal{I}$ (see Definition 5.16). Summing up the weight of the

lift F of T' and of $\bigcup_{S\in\mathcal{L}^*_{\max}} F_S$ we get

$$w(T) \leq (\kappa + \eta(1-\delta) + \alpha_{\mathrm{S}} + 3 + 2\rho(1-\delta))\,\mathrm{value}(\mathcal{I}) = \rho\,\mathrm{value}(\mathcal{I})\,,$$

by the selection of ρ to equal $(\kappa + \eta(1-\delta) + \alpha_{\mathrm{S}} + 3)/(2\delta - 1)$. This concludes the inductive step and the proof of the theorem. □

Chapter 6

Solving Subtour Partition Cover

In this chapter we solve Subtour Partition Cover on vertebrate pairs. By the reductions in Chapters 4 and 5, this is sufficient for obtaining a constant-factor approximation algorithm for general ATSP. We combine all the ingredients and calculate the obtained ratio in Section 6.2.

6.1 Algorithm for vertebrate pairs

In this section we consider a vertebrate pair $(\mathcal{I}, B)$ and prove the following theorem and corollary. This provides the algorithm required in Theorem 5.18.

Theorem 6.1

There exists a $(4, 2\operatorname{value}(\mathcal{I}) + \operatorname{lb}_{\mathcal{I}}(\bar{B}))$-light algorithm for Subtour Partition Cover for vertebrate pairs $(\mathcal{I}, B)$.

Combined with Theorem 4.3, we immediately obtain the following.

Corollary 6.2

For every $\epsilon > 0$ there is a polynomial-time algorithm that, given a vertebrate pair $(\mathcal{I}, B)$, returns a tour T of $\mathcal{I}$ with $w(T) \leq 2\operatorname{value}(\mathcal{I}) + (37 + 36\varepsilon)\operatorname{lb}_{\mathcal{I}}(\bar{B}) + w(B)$.

Throughout this section we will assume that $B \neq \emptyset$. In the special case when $B = \emptyset$, it must be the case that $\mathcal{L}_{\geq 2} = \emptyset$ and thus the instance is singleton; in that case, we simply apply the strictly better $(2, 0)$-light algorithm of Theorem 4.2.

We now formulate our main technical lemma. Let $\mathcal{L}_{\geq 2}$ denote the family of non-singleton sets in $\mathcal{L}$.

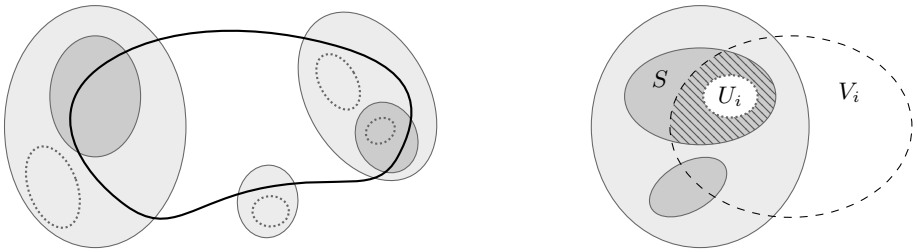


Figure 6.1: On the left, the "dotted" sets $U_1, \ldots, U_\ell$ of Lemma 6.3 are depicted.
On the right, we show how the set U_i is obtained by the algorithm for Subtour Partition Cover in the proof of Theorem 6.1: V_i is intersected with a minimal non-singleton set S to obtain V_i' (the striped area). Then, U_i is a source component in the decomposition of V_i' into strongly connected components. This implies that there are no edges from $V_i' \setminus U_i$ to U_i and so any edge in $\delta(V_i \setminus U_i, U_i)$ must come from outside of V_i' and thus cross the tight set S.

Lemma 6.3

There is a polynomial-time algorithm that solves the following problem. Let $(\mathcal{I}, B)$ be a vertebrate pair, and let $U_1, \ldots, U_\ell \subseteq V \setminus V(B)$ be disjoint nonempty vertex sets such that the subgraphs $G[U_1], \ldots, G[U_\ell]$ are strongly connected and for every $S \in \mathcal{L}_{\geq 2}$ and $i = 1, \ldots, \ell$ we have either $U_i \cap S = \emptyset$ or $U_i \subseteq S$. Then the algorithm finds a collection of subtours $F \subseteq E$ such that:

(a) $w(F) \leq 2\,\mathrm{value}(\mathcal{I}) + \mathrm{lb}_{\mathcal{I}}(\bar{B})$,

(b) $|\delta_F^-(U_i)| \geq 1$ *for every* $i = 1, \ldots, \ell$,

(c) $|\delta_F^-(v)| \leq 4$ *whenever* $x(\delta^-(v)) = 1$,

(d) *any subtour in F that crosses a set in $\mathcal{L}_{\geq 2}$ visits a vertex of the backbone.*

Notice that the requirements on the disjoint sets $U_1, \ldots, U_\ell$ imply that $\mathcal{L}_{\geq 2} \cup \{U_1, \ldots, U_\ell\}$ is a laminar family in which the sets $U_1, \ldots, U_\ell$ are minimal (see the left part of Figure 6.1). We also remark that property *(d)* will be important for analyzing the lightness of our tour. Indeed, it will imply that any subtour in our solution $F^\star$ to Subtour Partition Cover that is disjoint from the backbone does not cross a set in $\mathcal{L}_{\geq 2}$. Thus any edge (u, v) in such a subtour will have weight equal to $y_u + y_v$. Intuitively, this (almost) reduces the problem to the singleton case.

The proof will be given in Section 6.1.1, using the concept of *witness flows*. The witness flow is a tool that allows us to enforce the crucial property *(d)* in our solution to Subtour Partition Cover. It is inspired by a general method of ensuring connectivity in integer/linear programming formulations for graph problems, which requires the existence of a flow (supported on the LP solution) between the pairs of vertices that should be connected. The same role was played in our previous work on ATSP on graphs with two different edge weights [STV18b] and in our conference paper on the general ATSP result [STV18a] by a concept called the *split graph*.

Proof of Theorem 6.1.
Let $(V_1, V_2, \ldots, V_k)$ be the input partition of $V \setminus V(B)$ in the Subtour Partition Cover problem. We will apply Lemma 6.3 for a collection $(U_1, U_2, \ldots, U_k)$ of disjoint subsets with $U_i \subseteq V_i$, defined as follows. For $i = 1, \ldots, k$, let V_i' be the intersection of V_i with a minimal set $S \in \mathcal{L}_{\geq 2} \cup \{V\}$ with $S \cap V_i \neq \emptyset$. Then consider a decomposition of V_i' into strongly connected components (with

respect to $G[V_i']$). Let $U_i \subseteq V_i'$ be the vertex set of a source component in this decomposition. That is, there is no edge from $V_i' \setminus U_i$ to U_i in G (see also the right part of Figure 6.1). By construction, the sets $U_1, \dots, U_k$ satisfy the conditions of Lemma 6.3; in particular, V_i' (and thus U_i) is a subset of the minimal set S chosen above, which makes U_i a subset of any laminar superset of S and disjoint from any laminar set that is not a superset of S. We let F be the Eulerian multiset guaranteed by Lemma 6.3.

The rest of the proof is dedicated to showing that F satisfies the requirement of an $(4, 2\,\mathrm{value}(\mathcal{I}) + \mathrm{lb}_{\mathcal{I}}(\bar{B}))$-light algorithm. Let us start with the connectivity requirement.

Claim 6

We have $|\delta_F^-(V_i)| \geq 1$ for $i = 1, 2, \dots, k$.

Proof.
By property *(b)* of Lemma 6.3, there exists an edge $e \in \delta_F^-(U_i)$. Then either $e \in \delta_F^-(V_i)$ (in which case we are done), or $e \in \delta_F(V_i \setminus U_i, U_i)$. Assume the latter case.

Using that U_i was a source component in the decomposition of V_i' into strongly connected components, e must enter a set in $\mathcal{L}_{\geq 2}$. Indeed, recall that U_i was selected so that there is no edge from $V_i' \setminus U_i$ to U_i. Since $e \in \delta_F(V_i \setminus U_i, U_i)$ and $\delta(V_i' \setminus U_i, U_i) = \emptyset$, we must have $e \in \delta(V_i \setminus V_i', U_i) \subseteq \delta(V_i \setminus V_i', V_i')$. However, V_i' was obtained by intersecting V_i with a minimal set $S \in \mathcal{L}_{\geq 2} \cup \{V\}$ with $S \cap V_i \neq \emptyset$. Thus we must have $e \in \delta_F^-(S)$ (and $S \neq V$). Now, property *(d)* of Lemma 6.3 guarantees that the connected component (i.e., the subtour) of F containing e must visit $V(B)$. This subtour thus visits both V_i (the head of e is in $U_i \subseteq V_i$) and $V(B)$, which is disjoint from V_i. As such, the subtour must cross V_i, i.e., we have $|\delta_F^-(V_i)| \geq 1$ as required. □

Next, let us consider subtours in F that are disjoint from B.

Claim 7

Let T be a subtour in F with $V(T) \cap V(B) = \emptyset$. Then $w(T) \leq 4\,\mathrm{lb}(T)$.

Proof.
Recall that the lower bound is

$$\mathrm{lb}(T) = 2 \sum_{v \in \tilde{V}(T)} y_v \,.$$

To bound the weight of T, note that by property *(d)* of Lemma 6.3, the edges of T do not cross any tight set in $\mathcal{L}_{\geq 2}$. Therefore any edge (u, v) in T has weight $y_u + y_v$ and so

$$w(T) = \sum_{e \in T} w(e) = \sum_{v \in V(T)} |\delta_F(v)| y_v \leq 8 \sum_{v \in V(T)} y_v = 4\,\mathrm{lb}(T)\,,$$

where for the inequality we used that y_v is only strictly positive if $x^\star(\delta^-(v)) = 1$ (see Definition 3.3), in which case $|\delta_F(v)| = 2|\delta_F^-(v)| \leq 8$ using property *(c)* of Lemma 6.3. □

Finally, let $F_B \subseteq F$ be the collection of subtours in F that intersect B. Then, $w(F_B) \leq w(F) \leq 2\,\mathrm{value}(\mathcal{I}) + \mathrm{lb}_{\mathcal{I}}(\bar{B})$ by property *(a)* of Lemma 6.3. This completes the proof that F is a $(4, 2\,\mathrm{value}(\mathcal{I}) + \mathrm{lb}_{\mathcal{I}}(\bar{B}))$-light edge set. □

The rest of this section is devoted to the proof of the main technical Lemma 6.3.

6.1.1 Witness flows

Recall that $\mathcal{L}_{\geq 2}$ denotes the family of non-singleton sets in $\mathcal{L}$. Let us use an indexing $\mathcal{L}_{\geq 2} \cup \{V\} = \{S_1, S_2, \ldots, S_\ell\}$ such that $2 \leq |S_1| \leq |S_2| \leq \cdots \leq |S_\ell| = |V|$. For a vertex $v \in V$ let

$$\text{level}(v) = \min\{i : v \in S_i\}$$

be the index of the first (smallest) set that contains v. We use these levels to define a partial order $\prec$ on the vertices: let $v \prec v'$ if $\text{level}(v) < \text{level}(v')$. This partial order is used to classify the edges as follows. An edge $(u, v) \in E$ is a

- *forward* edge if $v \prec u$,
- *backward* edge if $u \prec v$,

and otherwise it is a *neutral* edge. Let E_f, E_b and E_n denote the sets of forward, backward, and neutral edges respectively.

Definition 6.4

Let $z : E \to \mathbb{R}$ be a circulation. We say that $f : E \to \mathbb{R}$ is a witness flow *for z if*

(a) $f \leq z$,
(b) $f(\delta^+(v)) \geq f(\delta^-(v))$ *for every* $v \in V \setminus V(B)$,
(c) $f(e) = 0$ *for each backward edge* $e \in E_b$,
(d) $f(e) = z(e)$ *for each forward edge* $e \in E_f$.

We say that a circulation z is witnessed *if there exists a witness flow for z.*

The following lemma reveals the importance of witness flows. By a component of a circulation z we mean a connected component of $\text{supp}(z)$.

Lemma 6.5

Let z be a witnessed circulation. Any component C of z that crosses a set in $\mathcal{L}_{\geq 2}$ must intersect B.

Proof.
Let f be a witness flow for z. Take i to be the smallest value such that $S_i \cap V(C) \neq \emptyset$. Then we must also have $V(C) \setminus S_i \neq \emptyset$ and so C must have an edge entering S_i; moreover, all edges of C entering S_i are forward edges, and all edges of C exiting S_i are backward edges. Let $U = S_i \cap V(C)$. Then $f(\delta^-(U)) > 0$ and $f(\delta^+(U)) = 0$. If we had $U \cap V(B) = \emptyset$, then *(b)* would imply that

$$0 > f(\delta^+(U)) - f(\delta^-(U)) = \sum_{v \in U} f(\delta^+(v)) - f(\delta^-(v)) \geq 0\,,$$

a contradiction. □

The edge set F in Lemma 6.3 will be obtained from a witnessed integer circulation; in particular, we will use the witness flow to derive property *(d)*. We start by showing that the Held-Karp solution x is a witnessed circulation.

Lemma 6.6

The Held-Karp solution x is a witnessed circulation.

Before giving the full proof, let us motivate the existence of a witness flow f in a simple example scenario where there is only one non-singleton set $S \in \mathcal{L}_{\geq 2}$. Then we have $E_f = \delta^-(S)$ and $E_b = \delta^+(S)$, i.e., the forward/backward edges are exactly the incoming/outgoing edges of S. The subtour elimination constraints imply (via the min-cut max-flow theorem) that x supports a unit flow between any pair of vertices. Let f be such a flow from any vertex outside S to a vertex $v \in S \cap V(B)$ (such a v exists by the backbone property). It is easy to see that f satisfies the conditions of the claim. Indeed, since S is a tight set, f saturates all incoming (forward) edges. It also does not leave S, i.e., use any backward edges. The proof of the general case uses an argument based on LP duality to argue the existence of f.

Proof of Lemma 6.6.
We find a witness flow f in polynomial time by solving the following linear program:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E_f} f(e) \\ \text{subject to } & f(\delta^+(v)) \geq f(\delta^-(v)) \quad \text{for } v \in V \setminus V(B), \\ & f(e) = 0 \qquad\qquad\qquad \text{for } e \in E_b, \\ & 0 \leq f \leq x^\star. \end{aligned}$$

By the constraints of the linear program, we have that f satisfies *(a)*, *(b)*, and *(c)*. It remains to verify *(d)*, or equivalently, to show that the optimum value of this program equals $x(E_f)$.

This will be shown using the dual linear program. The variables $(\pi_v)_{v \in V}$ correspond to the first set of constraints, and $(z(e))_{e \in E_f \cup E_n}$ to the capacity constraints on forward and neutral edges. No such variables are needed for backward edges. For notational simplicity, we introduce π_v also for $v \in V(B)$, and set $\pi_v = 0$ in this case. The dual program can be written as follows.

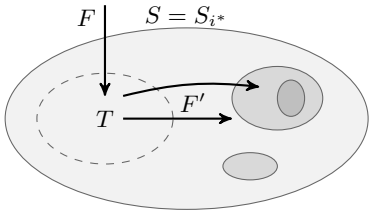
$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E_f \cup E_n} x(e) z(e) \\ \text{subject to } & \pi_v - \pi_u + z(u,v) \geq 1 \quad \text{for } (u,v) \in E_f, \\ & \pi_v - \pi_u + z(u,v) \geq 0 \quad \text{for } (u,v) \in E_n, \\ & \pi_v = 0 \qquad \text{for } v \in V(B), \\ & \pi, z \geq 0. \end{aligned}$$

Note that setting $\pi = \vec{0}$, $z(e) = 1$ if $e \in E_f$, and $z(e) = 0$ if $e \in E_n$ is a feasible solution with objective value $x(E_f)$. We complete the proof by showing that this is an optimal dual solution.

Let us select a dual optimal solution (π, z) that minimizes $\pi(V)$. We show that this minimum value is 0, that is, there exists a dual optimal solution with $\pi = \vec{0}$. This immediately implies that the above solution is a dual optimal one, because given $\pi = \vec{0}$ we get a constraint $z(u,v) \geq 1$ for all $(u,v) \in E_f$, making the objective value at least $x(E_f)$.

Towards a contradiction, assume that π is not everywhere zero. Let us select a vertex $t \in V$ such that $\pi_t > 0$, and $\text{level}(t) = i^*$ is minimal among all such vertices. Let $S = S_{i^*}$, and define $T = \{u \in S : \pi_u > 0\}$. Since $S \cap V(B) \neq \emptyset$, and $\pi_u = 0$ for all $u \in V(B)$, we see that T is

a proper subset of S. Let $F = \delta(V \setminus S, T)$ and $F' = \delta(T, S \setminus T)$. A depiction of the sets is as follows:



Let us show that the edges between T and $S \setminus T$ can be only of certain types.

Claim. *$F' = \delta(T, S \setminus T) \subseteq E_f \cup E_n$ and $\delta(S \setminus T, T) \subseteq E_b \cup E_n$.*

Proof of Claim.
By the choice of i^*, for any $S_i \subsetneq S$ we have that $\pi_v = 0$ for all $v \in S_i$; thus $S_i \cap T = \emptyset$, and so for every $u \in T$ we have $\text{level}(u) = i^*$. Therefore, for every $(u, v) \in F'$, we must have $\text{level}(u) = i^* \geq \text{level}(v)$, and for every $(u, v) \in \delta(S \setminus T, T)$, $\text{level}(v) = i^* \geq \text{level}(u)$. ◇

Let us construct another dual solution (π', z') as follows. Let $\varepsilon = \min\{\pi_u : u \in T\}$, and let

$$\pi'_u = \begin{cases} \pi_u - \varepsilon & \text{if } u \in T, \\ \pi_u & \text{otherwise,} \end{cases} \quad \text{and} \quad z'(e) = \begin{cases} z(e) + \varepsilon & \text{if } e \in F \cap (E_f \cup E_n), \\ z(e) - \varepsilon & \text{if } e \in F', \\ z(e) & \text{otherwise.} \end{cases}$$

We show that (π', z') is another optimal solution. Then, $\pi'(V) < \pi(V)$ gives a contradiction to the choice of (π, z). The proof proceeds in two steps: first we show feasibility and then optimality.

Feasibility. We have $\pi' \geq 0$ by the choice of ε. To show $z' \geq 0$, note that for $e = (u, v) \in F'$, by the Claim and the dual constraints for $e \in E_f \cup E_n$ we have $0 \leq \pi_v - \pi_u + z(u, v) \leq z(u, v) - \varepsilon$, where for the second inequality we used that $\pi_u \geq \varepsilon$ and $\pi_v = 0$. Thus, $z(e) \geq \varepsilon$ and $z'(e) = z(e) - \varepsilon \geq 0$ for every $e \in F'$. For an edge $e \notin F'$, we have $z'(e) \geq z(e) \geq 0$ and so we can conclude that $z' \geq 0$. We also have $\pi'_v = 0$ for $v \in V(B)$ since $T \cap V(B) = \emptyset$.

It remains to verify the constraints for $(u, v) \in E_f \cup E_n$. For every $(u, v) \in (F \cup F') \cap (E_f \cup E_n)$, as well as for edges not in $\delta(T)$, we have $\pi'_v - \pi'_u + z'(u, v) = \pi_v - \pi_u + z(u, v)$, and therefore the constraint remains valid. Thus we may have $\pi'_v - \pi'_u + z'(u, v) \neq \pi_v - \pi_u + z(u, v)$ in only two cases: either **(i)** if $(u, v) \in \delta(T, V \setminus S)$, or **(ii)** if $(u, v) \in \delta(S \setminus T, T)$.

In case **(i)**, the constraint on (u, v) remains valid since $\pi'_v - \pi'_u + z'(u, v) = \pi_v - \pi_u + z(u, v) + \varepsilon$. In case **(ii)**, $\pi'_u = 0$, $\pi'_v \geq 0$, and $z'(u, v) \geq 0$. Further, we have shown in the above Claim that $(u, v) \in E_b \cup E_n$. There is no constraint for $(u, v) \in E_b$, and $\pi'_v - \pi'_u + z'(u, v) \geq 0$ holds if $(u, v) \in E_n$.

Optimality. When changing (π, z) to (π', z'), the objective value increases by $\varepsilon(x(F \cap (E_f \cup E_n)) - x(F' \cap (E_f \cup E_n))) = \varepsilon(x(F \setminus E_b) - x(F'))$; we will show that this is non-positive. Recall that S is either a tight set or V, so that $x(\delta^-(S)) \leq 1$. Since $T \subsetneq S$, we have $1 \leq x(\delta^-(S \setminus T)) = x(\delta^-(S)) - x(F) + x(F') \leq 1 - x(F) + x(F')$, and therefore $x(F') \geq x(F) \geq x(F \setminus E_b)$. Thus, the objective value does not increase, therefore (π', z') must be optimal.

The existence of the optimal solution (π', z') with $\pi'(V) < \pi(V)$ contradicts the choice of (π, z), which completes the proof of Lemma 6.6. □

Let us state one more lemma that enables rounding fractional witness flows to integer ones. Recall that in the proof of Theorem 4.2 for singleton instances a key step was to round a fractional circulation to an integer one, a simple corollary of the integrality of the network flow polyhedron. We will now need a stronger statement as we need to round a circulation z along with a witness flow f consistently. We formulate the following general statement (which does not assume that we have a vertebrate pair or that f is a witness flow).

Lemma 6.7

For a directed graph $G = (V, E)$ and edge weights $w : E \to \mathbb{R}$, consider $z, f : E \to \mathbb{R}_+$ such that z is a circulation and $f \leq z$. Then there exist integer-valued vectors $\bar{z}, \bar{f} : E \to \mathbb{Z}_+$ with $w^\top \bar{z} \leq w^\top z$ satisfying the following properties:

(a) $\bar{z}$ is a circulation,

(b) $\bar{f}(\delta^+(v)) \geq \bar{f}(\delta^-(v))$ whenever $f(\delta^+(v)) \geq f(\delta^-(v))$,

(c) $\lfloor f(\delta^-(v)) \rfloor \leq \bar{f}(\delta^-(v)) \leq \lceil f(\delta^-(v)) \rceil$ for every node $v \in V$,

(d) $\lfloor g(\delta^-(v)) \rfloor \leq \bar{g}(\delta^-(v)) \leq \lceil g(\delta^-(v)) \rceil$ for every node $v \in V$, where $g = z - f$ and $\bar{g} = \bar{z} - \bar{f}$.

(e) $\bar{f} \leq \bar{z}$; $\bar{f}_e = \bar{z}_e$ whenever $f_e = z_e$, and $\bar{f}_e = 0$ whenever $f_e = 0$.

The proof relies on the total unimodularity of network matrices. Let (V, E) be a directed graph and $T = (V, E_T)$ a directed tree on the same node set. These define a *network matrix* $B \in \mathbb{Z}^{|E_T| \times |E|}$ as follows. For every $e = (u, v) \in E$, let P_e be the unique undirected $u - v$ path in the tree T. Then we set $B_{e_T,e} = 1$ if e_T occurs in forward direction in P_e, $B_{e_T,e} = -1$ if e_T occurs in the backward direction in P_e, and $B_{e_T,e} = 0$ if e_T does not occur in P_e. We will use the following result (see e.g. [Sch03, Theorem 13.20]):

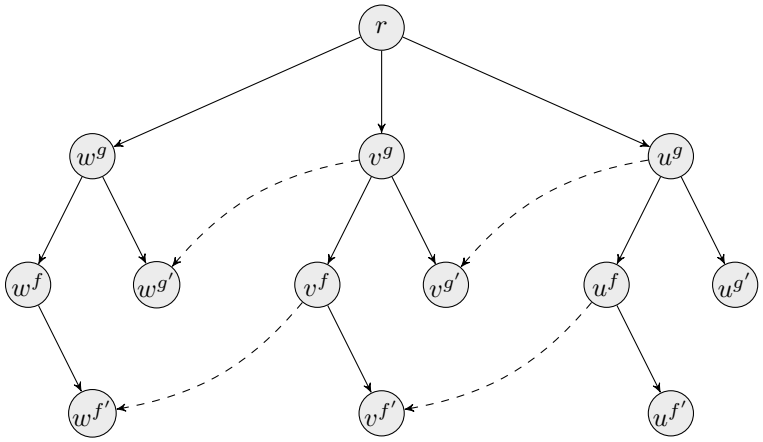
Theorem 6.8 (*[Tut65]*)

Every network matrix is totally unimodular.

Consequently, any LP of the form $\min\{c^\top x : Bx = b, x \geq 0\}$ has an integer optimal solution for any integer vector $b \in \mathbb{Z}^{|E_T|}$. A fundamental example of network matrices is the incidence matrix of a directed graph $G = (V, E)$. Consider the directed graph $(V \cup \{r\}, E)$ obtained by adding a new vertex r and the tree $T = (V \cup \{r\}, \{(r, u) : u \in V\})$ that forms a star. The associated network matrix is identical to the incidence matrix of G. The proof below extends this simple construction to capture all degree requirements.

Proof of Lemma 6.7.
Let us introduce new variables $\bar{g} = \bar{z} - \bar{f}$ (the notation already used in property (d)). The integrality of $\bar{z}$ is equivalent to the integrality of $\bar{f}$ and $\bar{g}$.

Let us construct a network matrix $B \in \mathbb{Z}^{4|V| \times 2|E|}$ as follows. We define a tree $T = (V', E_T)$, where V' contains a root node r, and for each $v \in V$, we include four nodes v^g, v^f, $v^{g'}$, $v^{f'}$ and four edges (r, v^g), (v^g, v^f), $(v^f, v^{f'})$ and $(v^g, v^{g'})$ in E_T. Let us define the directed graph (V', E'), where $E' = E'_f \cup E'_g$ is obtained as follows: for each $(u, v) \in E$, we add an edge $(u^f, v^{f'})$ to E'_f and an edge $(u^g, v^{g'})$ to E'_g. Let B be the network matrix corresponding to (V', E') and T (see Figure 6.2 for an example). That is, B is an $|E_T| \times |E'|$ matrix such that



part of v's:	inc. f-flow $(u^f, v^{f'})$	inc. g-flow $(u^g, v^{g'})$	out. f-flow $(v^f, w^{f'})$	out. g-flow $(v^g, w^{g'})$	corresponds to v's constraint:
(r, v^g)	1	1	-1	-1	(a)
(v^g, v^f)	1	0	-1	0	(b)
$(v^f, v^{f'})$	1	0	0	0	(c)
$(v^g, v^{g'})$	0	1	0	0	(d)
(r, u^g)	-1	-1	0	0	
(u^g, u^f)	-1	0	0	0	
$(u^f, u^{f'})$	0	0	0	0	
$(u^g, u^{g'})$	0	0	0	0	

Figure 6.2: The network matrix in the proof of Lemma 6.7.
Top: fragment of the tree $T = (V', E_T)$ corresponding to three vertices u, v, w and two edges (u, v), (v, w) (whose images are dashed).
Bottom: fragment of the network matrix B corresponding to vertices u, v and edges (u, v), (v, w), illustrating how rows of B encode the degree requirements on vertex v.

- the column corresponding to $(u^f, v^{f'}) \in E'_f$ has a -1 entry in the rows corresponding to the edges (r, u^g) and (u^g, u^f), and $+1$ entries corresponding to the edges (r, v^g), (v^g, v^f), and $(v^f, v^{f'})$; all other entries are 0;
- the column corresponding to $(u^g, v^{g'}) \in E'_g$ has a -1 entry in the row corresponding to the edge (r, u^g), and $+1$ entries corresponding to the edges (r, v^g) and $(v^g, v^{g'})$; all other entries are 0.

Then B is a totally unimodular matrix according to Theorem 6.8. Using the variables $\bar{f}$ and $\bar{g}$, the system (a)-(e) can be equivalently written in the form

$$b^{\text{lower}} \leq B(\bar{f}, \bar{g}) \leq b^{\text{upper}}$$
$$d^{\text{lower}} \leq (\bar{f}, \bar{g}) \leq d^{\text{upper}},$$

where $b^{\text{lower}}, b^{\text{upper}}, d^{\text{lower}}, d^{\text{upper}}$ are vectors with integer or infinite entries. Namely, the variable $\bar{f}(u, v)$ corresponds to the column for $(u^f, v^{f'}) \in E'_f$, and the variable $\bar{g}(u, v)$ corresponds to the

column for $(u^g, v^{g'}) \in E'_g$. See also Figure 6.2.

- The rows for $(r, v^g) \in E_T$ describe the flow conservation constraints as in (a) with $b^{\text{lower}}(r, v^g) = b^{\text{upper}}(r, v^g) = 0$.
- The rows for $(v^g, v^f) \in E_T$ describe the constraints as in (b), with $b^{\text{lower}}(r, v^f) = -\infty$ and $b^{\text{upper}}(r, v^f) = 0$ for those $v \in V$ with $f(\delta^+(v)) \geq f(\delta^-(v))$, and $b^{\text{upper}}(r, v^f) = +\infty$ for other $v \in V$.
- The rows for $(v^f, v^{f'}) \in E_T$ describe the constraints as in (c) with $b^{\text{lower}}(v^f, v^{f'}) = \lfloor f(\delta^-(v)) \rfloor$ and $b^{\text{upper}}(v^f, v^{f'}) = \lceil f(\delta^-(v)) \rceil$.
- The rows for $(v^g, v^{g'}) \in E_T$ describe the constraints as in (d) with $b^{\text{lower}}(v^g, v^{g'}) = \lfloor g(\delta^-(v)) \rfloor$ and $b^{\text{upper}}(v^f, v^{f'}) = \lceil g(\delta^-(v)) \rceil$ (recall that $g = z - f$).
- The capacity constraints $d^{\text{lower}} \leq (\bar{f}, \bar{g}) \leq d^{\text{upper}}$ are used to describe (e).

With $g = z - f$, the vector (f, g) is a feasible solution to the system. Consider the cost function $(w, w) \in \mathbb{R}^{2|E|}$. Using total unimodularity, there must be an integer solution $(\bar{f}, \bar{g})$ with $(w, w)^\top (\bar{f}, \bar{g}) \leq (w, w)^\top (f, g) = w^\top z$. Thus, $\bar{z} = \bar{f} + \bar{g}$ and $\bar{f}$ satisfy the statement of the lemma. □

Equipped with the above lemmas, we are ready to prove Lemma 6.3.

Proof of Lemma 6.3.
We are given a vertebrate pair $(\mathcal{I}, B)$ and disjoint nonempty vertex sets $U_1, \ldots, U_\ell \subseteq V \setminus V(B)$ such that the subgraphs $G[U_1], \ldots, G[U_\ell]$ are strongly connected and for every $S \in \mathcal{L}_{\geq 2}$ and $i = 1, \ldots, \ell$, we have either $U_i \cap S = \emptyset$ or $U_i \subseteq S$. For the Held-Karp solution x, let f be the witness flow guaranteed by Lemma 6.6.

We can assume that each edge e has either $f(e) = 0$ or $f(e) = x(e)$ without loss of generality, by breaking e up into two parallel copies and dividing its x and f values between them appropriately. We say that an edge e is *marked* if $f(e) = x(e)$ and *unmarked* otherwise.

Let us now introduce a convenient decomposition of x that we will obtain and utilize in our algorithm. By a *2-cycle* we mean a closed walk that visits every vertex at most twice and contains every edge at most once. A 2-cycle $C \subseteq E$ is *consistent* if, for any two consecutive edges $(u, v), (v, v') \in C$ with $v \notin V(B)$, if (u, v) is marked then (v, v') is also marked.

Claim 8

We can in polynomial time decompose x into consistent 2-cycles. That is, there is a polynomial-time algorithm that outputs consistent 2-cycles $C_1, C_2, \ldots, C_\ell$ and multipliers $\lambda_1, \lambda_2, \ldots, \lambda_\ell \geq 0$ such that $x = \sum_{i=1}^{\ell} \lambda_i \mathbb{1}_{C_i}$, where $\mathbb{1}_C \in \{0, 1\}^E$ denotes the indicator vector of a 2-cycle C.

Proof.
The proof uses a variant of the standard cycle decomposition argument. We identify 2-cycles by constructing walks on edges in the support of x. The algorithm identifies and removes 2-cycles one by one. After the removal of every cycle, we maintain property *(b)* of witness flows: for every $v \in V \setminus V(B)$ the outgoing flow amount on marked edges is greater or equal to the incoming flow amount on marked edges.

We find walks using the following procedure. We start on an arbitrary (marked or unmarked) edge. If the walk uses a marked edge then let us select the next edge as a marked edge whenever

possible, and after an unmarked edge let us continue on an unmarked edge if possible. We terminate according to the following rules:

1. If we visit a node $v \in V(B)$ the second time, then we terminate the current walk. We select C as the segment of the walk between the first and second visits to v. (If we started the walk from v, we also count this as a visit.)

2. When visiting a node $v \in V \setminus V(B)$ for the second time, we terminate if the outgoing edge of the first visit and the incoming edge of the second visit are of the same type (marked or unmarked). We also terminate if every outgoing edge of v is marked. We select C as the segment of the walk between the first and second visits to v.

3. If we visit a node in $v \in V \setminus V(B)$ the third time, then we always terminate. Let (v, u_1) and (v, u_2) be the edges where the walk left v for the first and second time, and let (z_1, v) and (z_2, v) be the edges where we arrived to v the second and third times. If (v, u_2) and (z_2, v) are both unmarked or both marked, then we let C be the segment of the walk between (v, u_2) and (z_2, v). Otherwise, we let C be the segment of the walk between (v, u_1) and (z_2, v) (visiting v twice). (These edges are of the same type, and so are (z_1, v) and (v, u_2).)

Given C, we let λ denote the minimum flow value on any edge of C. We decrease the value of x on every edge of C by λ (deleting an edge if its x-value becomes zero), and also the value of f on every marked edge of C. We add C to the decomposition with coefficient λ, and proceed to finding the next 2-cycle.

Assume property *(b)* of witnessed flows holds when we start the walk. Thus, whenever the walk enters a node $v \in V \setminus V(B)$ on a marked edge, it can continue on a marked edge. By rule 2, if the walk enters on an unmarked edge, it can continue on an unmarked edge. The above rules guarantee that C will be a consistent 2-cycle: if we close C in a node $v \in V \setminus V(B)$, then if the incoming edge is marked, the outgoing edge will also be marked. Moreover, if the walk is not terminated, then it can continue on a yet-unused edge.

It remains to show that property *(b)* is maintained after removing C. Assume we decrease the outgoing flow at a node $v \in V \setminus V(B)$ on a marked edge. First, notice that there can be at most one outgoing marked edge from v on C. If there was also an incoming marked edge on C, then the marked flow decreases by the same amount on these two edges. If all incoming edges were unmarked, then our rules implies that all outgoing edges at v are marked. In this case, *(b)* is trivially maintained (as x is, and remains, a circulation).

The proof can be immediately turned into a polynomial-time algorithm. Notice that the number of edges decreases by at least one at the removal of every cycle. □

We will construct an auxiliary graph $G' = (V', E')$ similarly as in the proof of Theorem 4.2. For convenience, Figure 6.3 gives an overview of the different steps, graphs and flows used by our algorithm. We select edge sets X_i^-, X_i^+ and flows x_i and f_i as follows:

- For every U_i we can select (possibly by subdividing edges as above) a subset of incoming edges $X_i^- \subseteq \delta^-(U_i)$ with $x(X_i^-) = 1/2$ such that either all edges $e \in X_i^-$ are marked or all are unmarked. This is possible since $x(\delta^-(U_i)) \geq 1$ (and all edges are either marked or unmarked).

- Take the decomposition of x into 2-cycles as guaranteed by Claim 8, and follow the incoming edges in X_i^- in the decomposition. We let X_i^+ be the set of edges on which these walks first leave U_i after entering on an edge in X_i^-. We let x_i denote the respective x-flow on

	graph	integral	obtained from the previous by
x	G	no	input to Lemma 6.3
x, f	G	no	finding a witness flow (Lemma 6.6)
x', f'	G'	no	introducing a_i, redirecting edges, subtracting x_i and f_i
$\bar{z}', \bar{f}'$	G'	yes	rounding (Lemma 6.7 applied to $z = 2x'$ and $2f'$)
$\bar{z}, \bar{f}$	G	yes	un-redirecting edges, mapping back to G
z^*, f^*	G	yes	adding walks P_i (to $\bar{f}$ only if $\bar{f}'(\delta^-(a_i)) = 1$)

Figure 6.3: This table summarizes the various circulations and flows that appear in our algorithm, in order.

the segments of these walks connecting the heads of edges in X_i^- and the tails of edges in X_i^+. On the same edges, we define f_i by $f_i(e) = x_i(e)$ for every marked edge and $f_i(e) = 0$ for every unmarked edge.

Note that we have $0 \leq f - f_i \leq x - x_i$. We further claim that $(f - f_i)(\delta^+(v)) \geq (f - f_i)(\delta^-(v))$ for every $v \in V \setminus V(B)$ except for heads of edges in X_i^-. Indeed, at every node we consider those 2-cycles in the decomposition that were not used for x_i. In each of these 2-cycles an incoming marked edge must be followed by an outgoing marked edge.

We now transform G into a new graph G', x into a new circulation x', and f into a new f', as follows. For every $i = 1, \ldots, \ell$, we introduce a new auxiliary vertex a_i and redirect all edges in X_i^- to point to a_i, and those in X_i^+ to point from a_i. We subtract the flow $\hat{x}_i$ from x and $\hat{f}_i$ from f inside U_i; hence the resulting vector x' will be a circulation in G', and $f' \leq x'$. We now have $f'(\delta^+(v)) \geq f'(\delta^-(v))$ for all $v \in V \setminus V(B)$. We also have $x'(\delta^-(a_i)) = x'(\delta^+(a_i)) = 1/2$, and either $f'(\delta^-(a_i)) = 0$ (in the case when $f(X_i^-) = 0$) or $f'(\delta^-(a_i)) = f'(\delta^+(a_i)) = 1/2$ (in the case when $f(X_i^-) = 1/2$: since in this case all edges in X_i^+ must also be marked). We define the weights $w'(e) = w(e)$ if e was not modified, and for every redirected edge e, we set $w'(e)$ as the weight of the edge it was redirected from. Thus, the total weight may only decrease: $\sum_{e \in E'} w'(e)x'(e) \leq \sum_{e \in E} w(e)x(e) = \text{value}(\mathcal{I})$ (it decreases if the flows $\hat{x}_i$ are nonzero on edges with positive weight).

Let us now apply Lemma 6.7 in the graph G' with weights w' and $z = 2x'$ and $2f'$ in place of f. We thus obtain the integer vectors $\bar{z}'$ and $\bar{f}'$ with ${w'}^\top \bar{z}' \leq {w'}^\top z$. Note in particular that properties (c), (d) imply that $\bar{z}'(\delta^-(a_i)) = 1$ for every auxiliary vertex a_i; this is because $z(\delta^-(a_i)) = 1$ and $2f'(\delta^-(a_i)) \in \{0, 1\}$.

Now we map $\bar{z}'$ and $\bar{f}'$ back to the vectors $\bar{z}$ and $\bar{f}$ in the original graph G. Namely, if e is incident to an auxiliary vertex a_i, then we reverse the redirection of this edge, and move $\bar{z}$ and $\bar{f}$ to the original graph.

The vector $\bar{z}$ may not be a circulation. Specifically, since the in- and out-degree of a_i were exactly 1 in $\bar{z}'$, in each component U_i there is a pair of vertices u_i, v_i which are the head and tail, respectively, of the mapped-back edges adjacent to a_i. These are the only vertices whose in-degree may differ from their out-degree. (They differ unless $u_i = v_i$.) To repair this, for each $i = 1, \ldots, \ell$ with $u_i \neq v_i$, we route a walk P_i from u_i to v_i in U_i; this is always possible as we assumed that U_i is strongly connected.

We obtain a circulation z^* from $\bar{z}$ by increasing the value by 1 on every edge of every such path P_i. Further, we obtain f^* from $\bar{f}$ as follows. If $\bar{f}'(\delta^-(a_i)) = 0$, then we let f^* be identical to $\bar{f}$ inside U_i. If $\bar{f}'(\delta^-(a_i)) = 1$, then we increase the value of $\bar{f}$ by 1 on every edge of P_i.

Claim 9

We have $w^\top z^* \le 2\,\mathrm{value}(\mathcal{I}) + \mathrm{lb}_{\mathcal{I}}(\bar{B})$, and f^* is a witness flow for z^*.

Proof.
By the construction, $w^\top \bar{z} = w'^\top \bar{z}' \le w'^\top z = 2w'^\top x' \le 2w^\top x = 2\,\mathrm{value}(\mathcal{I})$. We obtained z^* from $\bar{z}$ by increasing it on a set of disjoint paths P_i in $V \setminus V(B)$, and these paths do not cross any set in $\mathcal{L}_{\ge 2}$. Thus, their total cost is bounded by $\mathrm{lb}_{\mathcal{I}}(\bar{B}) = 2\sum_{v \in V \setminus V(B)} y_v$.

We next verify that f^* is a witness flow for z^*. Properties *(a)*, *(c)*, and *(d)* easily follow. For *(b)*, recall that we had $f'(\delta^+(v)) \ge f'(\delta^-(v))$ for every $v \in V \setminus V(B)$, and hence the same holds for $\bar{f}'$ by Lemma 6.7. For $\bar{f}$, this condition can be violated only at some nodes u_i for some values of $i = 1, 2, \ldots, \ell$. We obtain f^* from $\bar{f}$ by increasing the flow value on the path P_i from u_i to v_i; this increases $f^*(\delta^+(u_i))$ so that $f^*(\delta^+(u_i)) \ge f^*(\delta^-(u_i))$, while not violating $f^*(\delta^+(v_i)) \ge f^*(\delta^-(v_i))$ since for such an i we must have $\bar{f}'(\delta^-(a_i)) = \bar{f}'(\delta^+(a_i)) = 1$ and so $\bar{f}(\delta^+(v_i)) \ge \bar{f}(\delta^-(v_i)) + 1$. □

Let F be the Eulerian edge multiset obtained by taking z^*_e copies of edge e. The proof concludes by showing that F satisfies all requirements of Lemma 6.3. The cost bound *(a)* follows by the claim above since $w(F) = w^\top z^*$. The connectivity requirement *(b)*, namely that $|\delta^-_F(U_i)| \ge 1$ for every $i = 1, \ldots, \ell$, is immediate by the construction.

Let us now show *(c)*, that is, $|\delta^-_F(v)| = z^*(\delta^-(v)) \le 4$ for every $v \in V$ such that $x(\delta^-(v)) = 1$. Note that we have $x'(\delta^-(v)) \le 1$. Denote $g' = x' - f'$ and $\bar{g}' = \bar{x}' - \bar{f}'$. By properties (c), (d) of Lemma 6.7 we have

$$\bar{z}'(\delta^-(v)) = \bar{f}'(\delta^-(v)) + \bar{g}'(\delta^-(v)) \le \lceil 2f'(\delta^-(v)) \rceil + \lceil 2g'(\delta^-(v)) \rceil \le 3\,,$$

as the maximum value of the function $\lceil 2p \rceil + \lceil 2q \rceil$ subject to $p + q \le 1$ is 3. Adding the paths P_i may further increase $z^*(\delta^-(v))$ over $\bar{z}'(\delta^-(v))$ by 1.

Property *(d)* requires that every subtour in F that crosses a tight set in $\mathcal{L}_{\ge 2}$ visit a vertex of the backbone. This follows from Lemma 6.5 since z^* is a witnessed circulation. □

6.2 Completing the puzzle: proof of Theorem 3.1

We now combine the techniques and algorithms of the previous sections to obtain a constant-factor approximation algorithm for ATSP. In multiple steps, we have reduced ATSP to finding tours for vertebrate pairs. Every reduction step was polynomial-time and increased the approximation ratio by a constant factor. Hence, together they give a constant-factor approximation algorithm for ATSP.

We now give an overview of these reductions and set the parameters. Throughout, $\varepsilon > 0$ will be a fixed small value. We set $\delta = 0.78$. All approximation guarantees are with respect to the optimum value of the Held-Karp relaxation $\mathrm{LP}(G, w)$. The reduction proceeds using the following algorithmic subroutines.

- Corollary 4.4 provides a polynomial-time α_{S}-approximation $\mathcal{A}_{\mathrm{S}}$ for singleton instances, with $\alpha_{\mathrm{S}} = 18 + \epsilon$. This will be used to find a (quasi-)backbone for irreducible instances (Lemma 5.17).

- Algorithm $\mathcal{A}_{\text{ver}}$, which, for a vertebrate pair $(\mathcal{I}, B)$, finds a tour of cost $\kappa\,\text{value}(\mathcal{I}) + \eta\,\text{lb}_B(V) + w(B)$, where $\kappa = 2$ and $\eta = 37 + 36\varepsilon$ (Corollary 6.2). Algorithm $\mathcal{A}_{\text{ver}}$ uses the reduction Theorem 4.3 from Subtour Partition Cover to ATSP.
- Algorithm $\mathcal{A}_{\text{irr}}$ which, provided $\mathcal{A}_{\text{ver}}$ as above, obtains a polynomial-time ρ-approximation algorithm for irreducible instances, where $\rho = (\kappa + \eta(1-\delta) + \alpha_{\text{S}} + 3)/(2\delta - 1) < 55.61$ for sufficiently small $\varepsilon > 0$ (Theorem 5.18).
- Algorithm $\mathcal{A}_{\text{lam}}$, which converts the ρ-approximation algorithm $\mathcal{A}_{\text{irr}}$ to a $2\rho/(1-\delta)$-approximation algorithm for an arbitrary laminarly-weighted instance $\mathcal{I}$. Here, $2\rho/(1-\delta) < 506$ (Theorem 5.14).
- Our final Algorithm $\mathcal{A}_{\text{ATSP}}$, which reduces an arbitrary input weighted directed graph (G, w) to a laminarly-weighted instance, keeping the same approximation ratio (Theorem 3.4).

All in all we have thus obtained a polynomial-time algorithm for ATSP that returns a tour of value at most 506 times the Held-Karp lower bound.

Integrality gap Theorem 3.1 of course implies an upper bound of 506 on the integrality gap of the Held-Karp relaxation. However, if we do not require a polynomial-time algorithm, then the loss factor of $9(1+\varepsilon)$ in the reduction of Theorem 4.3 can be decreased to 5. Therefore non-constructively we can have $\alpha'_{\text{S}} = 10$ and $\eta' = 1 + 5 \cdot 4$ (instead of $\eta = 1 + 9 \cdot (1+\varepsilon) \cdot 4$), which yields $\rho' < 35.04$ and a final integrality gap of at most 319.

Theorem 6.9

The integrality gap of the asymmetric Held-Karp relaxation is at most 319.

Asymmetric Traveling Salesman *Path* Problem In the Asymmetric Traveling Salesman Path Problem (ATSPP), in addition to the usual ATSP input, we are also given two special vertices $s, t \in V$ and wish to find a walk that visits all vertices, starts from s, and ends at t. Feige and Singh [FS07] proved that if there is a β-approximation algorithm for ATSP, then there is a $((2+\epsilon)\beta)$-approximation algorithm for ATSPP for any $\epsilon > 0$. Together with Theorem 3.1, this implies:

Corollary 6.10

There is a polynomial-time algorithm for ATSPP that returns a tour of value at most 1014 *times the integral optimum.*

Note that the approximation ratio here is not bounded in terms of the Held-Karp lower bound[12]. Köhne, Traub and Vygen [KTV18] give a similar reduction as Feige and Singh, but for integrality gaps, with a loss of $\beta \mapsto 4\beta - 3$. Together with Theorem 6.9, this implies:

Corollary 6.11

The integrality gap of the asymmetric Held-Karp relaxation for ATSPP is at most 1273.

Finally, since their reduction is constructive, together with Theorem 3.1 we get:

[12]For ATSPP, this is defined as the optimal value of a relaxation that is similar to $\text{LP}(G, w)$, with the differences that $x(\delta^+(s)) - x(\delta^-(s)) = 1$, $x(\delta^-(t)) - x(\delta^+(t)) = 1$, and the cuts $S \ni s$ are only required to have at least one outgoing edge (but possibly no incoming edge).

Corollary 6.12

There is a polynomial-time algorithm for ATSPP that returns a tour of value at most 2021 *times the Held-Karp lower bound.*

See the conclusions (Section 9) for a discussion of future research directions.

Part II

Matching is in Quasi-NC

Chapter 7

Perfect Matching and Parallel Algorithms

In this chapter we present background notions and facts on parallel algorithms and the class NC, linear-algebraic techniques for matchings, and isolating weight functions. See Section 2.4 for the definition of the problem and Section 2.5 for Edmonds' LP formulation for perfect matching.

Throughout Part II we consider a fixed graph $G = (V, E)$ with n vertices. For notational convenience, we assume that $\log_2 n$ evaluates to an integer; otherwise simply replace $\log_2 n$ by $\lceil \log_2 n \rceil$. We also assume that n is sufficiently large.

7.1 Parallel complexity classes

The matching problem has been considered in a multitude of computational models, including distributed and parallel settings. In this thesis we focus on the class NC. NC, standing for "Nick's Class" – named after Nicholas Pippenger by Stephen Cook – intuitively comprises those problems that parallelize completely.

Definition 7.1

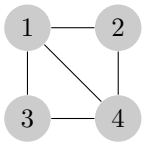
For $d \geq 0$, a problem is in NC^d if it is recognized by a logspace-uniform family of circuits of polynomial size and depth $O(\log^d n)$. We define $\mathsf{NC} = \bigcup_{d \geq 0} \mathsf{NC}^d$.

Alternatively, one can think of NC being the class of problems that have PRAM algorithms running in polylogarithmic time on a polynomial number of processors [AB09, Theorem 6.27].

The class Randomized NC, or RNC in short, is obtained if one allows the algorithm access to (polynomially many) random bits and one-sided error: when the correct answer is YES, the algorithm should return YES with probability at least $1/2$, and otherwise it must return NO. Thus RNC relates to NC similarly to how RP relates to P.

Finally, the complexity class quasi-NC is defined as $\mathsf{quasi\text{-}NC} = \bigcup_{d \geq 0} \mathsf{quasi\text{-}NC}^d$, where $\mathsf{quasi\text{-}NC}^d$ is the class of problems having polylogspace-uniform circuits of quasi-polynomial size $2^{\log^{O(1)} n}$ and polylogarithmic depth $O(\log^d n)$ [Bar92].

Many basic problems are in NC. One example that will be crucial in our algorithm is computing determinants:



$$T(G) = \begin{pmatrix} 0 & X_{12} & X_{13} & X_{14} \\ -X_{12} & 0 & 0 & X_{24} \\ -X_{13} & 0 & 0 & X_{34} \\ -X_{14} & -X_{24} & -X_{34} & 0 \end{pmatrix}$$

Figure 7.1: Example of a Tutte matrix of an undirected graph.

Theorem 7.2 (*[Csa76, Ber84, MV97]*)

There is an NC^2 *algorithm for computing the determinant of an integer-valued matrix.*

7.2 Linear-algebraic techniques for matchings

A successful approach to the perfect matching problem has been the linear-algebraic one. It involves the *Tutte matrix* associated with a graph:

Definition 7.3

Given an undirected graph $G = (V, E)$*, the Tutte matrix* $T(G)$ *of* G *is a* $n \times n$ *matrix defined as follows:*

$$T(G)_{u,v} = \begin{cases} X_{(u,v)} & \text{if } (u,v) \in E \text{ and } u < v, \\ -X_{(v,u)} & \text{if } (u,v) \in E \text{ and } u > v, \\ 0 & \text{if } (u,v) \notin E, \end{cases}$$

where $X_{(u,v)}$ *for* $(u,v) \in E$ *are variables.*

See Figure 7.1 for an example. The Tutte matrix is skew-symmetric, i.e., $T(G)^\top = -T(G)$. Its significance is explained by the following theorem:

Theorem 7.4 (*Tutte's Theorem [Tut47]*)

G *has a perfect matching if and only if* $\det T(G) \neq 0$.

Tutte's Theorem is perhaps not as algorithmic as it sounds, for $T(G)$ is defined over a ring of many indeterminates; exponential time may be needed to even write down the determinant $\det T(G)$ explicitly as a sum of monomials. However, if we allow for randomization, then the Schwartz-Zippel Lemma gives a way to test whether the determinant is nonzero:

Lemma 7.5 (*[Zip79, Sch80]*)

For p *a prime, let* $f \in \mathbb{F}_p[X_1, ..., X_k]$ *be a nonzero polynomial. Pick* $r_1, ..., r_k \in \mathbb{F}_p$ *uniformly at random and independently. Then* $\mathbb{P}[f(r_1, ..., r_k) = 0] \leq \frac{\deg(f)}{p}$.

Theorem 7.4 and Lemma 7.5 give rise to a randomized algorithm due to Lovász:

Theorem 7.6 (*[Lov79]*)

The decision version of the perfect matching problem is in RNC.

Proof.
The algorithm is as follows. Let $T'(G) \in \mathbb{F}_p^{n \times n}$ be Tutte's matrix $T(G)$ with each variable $X_{(u,v)}$

replaced by a random value $r_{(u,v)} \in \mathbb{F}_p$, for a sufficiently large prime p (say $p \geq 2n$). Compute the determinant of $T'(G)$ over $\mathbb{F}_p$. Return YES if and only if $\det T'(G) = 0$.

If G has a perfect matching, then by Tutte's Theorem we have $\det T(G) \neq 0$ and thus $\det T'(G) \neq 0$ with probability at least $1 - \frac{n}{p} \geq \frac{1}{2}$ by Lemma 7.5. On the other hand, if G has no perfect matching, then $\det T(G) = 0$, which of course implies $\det T'(G) = 0$. Computing the determinant can be done in NC (Theorem 7.2). Finally, it is easy to find some prime p with $2n < p < 4n$ in NC: simply try all possible integers in this range. □

Note that Theorem 7.6 gives an RNC algorithm for the decision version of the perfect matching problem. From this, using a simple reduction, one easily gets an RP algorithm for the search version, i.e., given a graph, *find* a perfect matching if one exists; however, the resulting algorithm is inherently sequential. The search version has proved to be more difficult and it was found to be in RNC several years later by Karp, Upfal and Wigderson [KUW86] and Mulmuley, Vazirani and Vazirani [MVV87]. We explain the latter algorithm in the following Section 7.3.

As an aside, the linear-algebraic approach leads, among others, to the fastest known sequential running times for matching on dense graphs:

Theorem 7.7 (*[MS04, Har09]*)

There is a (sequential) algorithm that finds a maximum matching in an undirected graph in time $O(n^\omega)$, where $\omega < 2.376$ is the matrix multiplication exponent.

7.3 Isolating weight functions

In this section we discuss the RNC algorithm of Mulmuley, Vazirani and Vazirani [MVV87]. It is similar to the algorithm of Lovász in that it also replaces the variables $X_{(u,v)}$ in Tutte's matrix by random values. However, these values are random powers of two, and the determinant is computed over $\mathbb{Z}$ rather than over a prime field.

Somewhat unintuitively, their approach consists in reducing the unweighted problem of finding a perfect matching to the weighted problem of finding a minimum-weight perfect matching. It is crucial that the weight function can be chosen smartly. In fact, it is shown that if it yields a single minimum-weight perfect matching, then this matching can be found in NC.

Definition 7.8

Let $w : E \to \mathbb{Z}_+$ be a weight function. We say that w is isolating *if G has at most a* unique *minimum-weight perfect matching with respect to w.*

For example, it is easy to see that the weight function w obtained by enumerating $E = \{e_1, ..., e_{|E|}\}$ and setting $w(e_i) = 2^i$ is isolating.

Given an isolating weight function w, we can define a new matrix $T^w(G) \in \mathbb{Z}^{n \times n}$ to be the Tutte's matrix $T(G)$ with each variable $X_{(u,v)}$ replaced by the value $2^{w(u,v)}$. The crucial property of this matrix is captured by the following lemma.

Lemma 7.9

Let w be an isolating weight function. If G has no perfect matching, then $\det T^w(G) = 0$. Otherwise let M be the unique minimum-weight perfect matching. We have $\det T^w(G) \neq 0$, and $2^{2w(M)}$ is the highest power of two that divides $\det T^w(G)$.

The proof extends that of Tutte's Theorem.

Proof.
Let us write the determinant as

$$\det T^w(G) = \sum_{\sigma \in S_n} P_\sigma$$

where

$$P_\sigma = \operatorname{sgn}(\sigma) \prod_{i=1}^{n} T^w(G)_{i,\sigma(i)} .$$

The nonzero summands P_σ correspond to directed cycle covers of G (which can contain 2-cycles). First, we claim that those cycle covers that contain an odd cycle are unimportant.

Claim 10

Let

$$O_n = \{\sigma \in S_n : \sigma \text{ contains an odd cycle}\} .$$

Then $\sum_{\sigma \in O_n} P_\sigma = 0$.

Proof.
The elements of O_n can be paired up: for every $\sigma \in O_n$, reverse an odd cycle to get $\hat{\sigma}$ (to be precise, reverse the odd cycle that contains the lowest-indexed vertex). We have $\hat{\hat{\sigma}} = \sigma$ and $\operatorname{sgn}(\sigma) = \operatorname{sgn}(\hat{\sigma})$; moreover, when going from σ to $\hat{\sigma}$, an odd number of factors $T^w(G)_{i,\sigma(i)}$ change their sign (as the arc $\langle i, \sigma(i)\rangle$ is replaced by $\langle \sigma(i), i\rangle$). Hence $P_\sigma + P_{\hat{\sigma}} = 0$. □

Thus we have

$$\det T^w(G) = \sum_{\sigma \in S_n \setminus O_n} P_\sigma . \tag{7.1}$$

If G has no perfect matching, then it also has no even cycle cover and thus $\det T^w(G) = 0$ (in fact, $\det T(G) = 0$ as a polynomial). Suppose otherwise, and let M be the unique minimum-weight perfect matching. For the permutation $\sigma_M \in S_n \setminus O_n$ corresponding to M (σ_M consists of 2-cycles), we have

$$P_{\sigma_M} = \pm \prod_{(u,v) \in M} 2^{w(u,v)} \cdot 2^{w(v,u)} = \pm 2^{2w(M)} . \tag{7.2}$$

On the other hand:

Claim 11

For every $\sigma \in S_n \setminus O_n$ with $\sigma \neq \sigma_M$, P_σ is a multiple of $2^{2w(M)+1}$.

Proof.
Every $\sigma \in S_n \setminus O_n$ gives rise to two perfect matchings M_1 and M_2: order edges on every even cycle arbitrarily and let M_1 be the odd-numbered edges while M_2 is the even-numbered edges. (If σ consists of 2-cycles, then $M_1 = M_2$.) We have $P_\sigma = \pm 2^{w(M_1)+w(M_2)}$. If $\sigma \neq \sigma_M$, then $M_1 \neq M$ or $M_2 \neq M$, thus implying that $w(M_1) + w(M_2) \geq 2w(M) + 1$. □

By (7.1) and (7.2), Claim 11 implies the statement. □

Lemma 7.10 (*[MVV87]*)

There is an NC *algorithm that, given an undirected graph $G = (V, E)$ and an isolating weight function $w : E \to \mathbb{Z}$ with polynomially bounded entries, outputs the minimum-weight perfect matching of G if one exists.*

Proof.
An algorithm for the decision version of the perfect matching problem is implied immediately by Lemma 7.9. Indeed, as each $w(e)$ for $e \in E$ is polynomially bounded, the bit-length of the entries $2^{w(e)}$ of the matrix $T^w(G)$ is polynomial, and the determinant can still be computed in NC.

For the search version, suppose G has a unique minimum-weight perfect matching M. Then the lowest set bit in $\det T^w(G)$ corresponds to the term $2^{2w(M)}$, and thus the algorithm can easily learn $w(M)$. Now, for every edge $e \in E$ in parallel, we recompute the determinant for the graph $G - e$. The proof of Lemma 7.9 implies the following. If $e \notin M$, the lowest set bit does not change. Otherwise M is not a perfect matching in $G - e$ and thus every term in $\det T^w(G - e)$ is a multiple of $2^{2w(M)+1}$. Therefore the algorithm can find M by returning the set of those $e \in E$ for which $2^{2w(M)+1}$ does not divide $\det T^w(G - e)$. □

Note that the algorithm of Lemma 7.10 is deterministic. The randomized component of Mulmuley, Vazirani and Vazirani's algorithm deals with how to obtain an isolating weight function with polynomially-bounded values. The following statement, known as the Isolation Lemma, says that it is enough to simply sample the weights uniformly at random. It turns out to be true in the much more general setting of arbitrary set families:

Lemma 7.11 (*Isolation Lemma*)

Let $\mathcal{M} \subseteq 2^E$ be any nonempty family of subsets of a universe $E = \{1, 2, ..., |E|\}$. Suppose we define a weight function $w : E \to \{1, 2, ..., 2|E|\}$ by selecting each $w(e)$ for $e \in E$ independently and uniformly at random. Then with probability at least $1/2$, there is a unique *set $M \in \mathcal{M}$ that minimizes the weight $w(M) = \sum_{e \in M} w(e)$.*

For our purposes, we take $\mathcal{M}$ in Lemma 7.11 to be the set of all perfect matchings. The proof is very simple:

Proof.
Let us say that an element is special if it belongs to some but not to every minimum-weight set. Clearly, a minimum-weight set is not unique if and only if there exists a special element. Thus

$$\mathbb{P}[\text{minimum-weight set is not unique}] \leq \sum_{e \in E} \mathbb{P}[e \text{ is special}] .$$

We will be done if we show that $\mathbb{P}[e \text{ is special}] \leq \frac{1}{2|E|}$ for every $e \in E$. For this, condition on the random choices of all $w(e')$ with $e' \neq e$; now, there is at most one possible value that $w(e)$ can have if it is to be special. (Indeed, if e is special, then decreasing $w(e)$ causes it to be in every minimum-weight set, and increasing $w(e)$ causes it to be in no minimum-weight set.) The probability of $w(e)$ taking that value is either $\frac{1}{2|E|}$ or 0. □

Lemmas 7.10 and 7.11 together imply an RNC algorithm for (the search version of) the perfect matching problem [MVV87].

7.4 Derandomizing the Isolation Lemma

To obtain an NC algorithm for matching, it would be enough to be able to generate isolating weight functions (that are polynomially bounded) in NC. We do not know how to do this, even for bipartite graphs. However, Fenner, Gurjar and Thierauf have proved the following:

Theorem 7.12 (*[FGT16]*)

For any number n, we can in quasi-NC *construct $n^{O(\log n)}$ weight functions on $\{1, 2, ..., (n/2)^2\}$ with weights bounded by $n^{O(\log n)}$ such that for any bipartite graph on n vertices*[13]*, one of these weight functions is isolating.*

Corollary 7.13

The perfect matching problem for bipartite graphs is in quasi-NC.

Proof.
For every weight function in parallel, run the algorithm of Lemma 7.10 (which will be in quasi-NC, as the weights are quasi-polynomially rather than polynomially bounded). If the graph has a perfect matching, it will be found by one of the runs; otherwise they will all return NO. □

Note that there are two aspects in which Theorem 7.12 is weaker than computing an isolating weight function in NC. First, it yields a quasi-NC rather an NC algorithm. Second, we obtain many weight functions, one of which is guaranteed to be isolating, but does not give an efficient way to ascertain which one. At the same time, the oblivious aspect of the construction is an advantage: the family of weight functions is constructed knowing only the number n of vertices (and the construction is very simple). We explain the main ideas of the very elegant proof of Theorem 7.12 in Section 8.1.1.

The main result of this part of the thesis is the following version of Theorem 7.12 for general graphs:

Theorem 7.14 (*[ST17]*)

For any number n, we can in quasi-NC *construct $n^{O(\log^2 n)}$ weight functions on $\{1, 2, ..., \binom{n}{2}\}$ with weights bounded by $n^{O(\log^2 n)}$ such that for any graph on n vertices, one of these weight functions is isolating.*

Chapter 8 is dedicated to the proof of Theorem 7.14. We remark that the implied quasi-NC algorithm for perfect matching is very simple. The complexity lies in the analysis, i.e., proving that one of the weight functions is isolating (see Theorem 8.17).

We conclude this section with a formal statement of the implication:

Corollary 7.15

The perfect matching problem is in quasi-NC^3*: it has a parallel algorithm that uses $n^{O(\log^2 n)}$ processors and $O(\log^3 n)$ time.*

Proof.
We follow the proof of Corollary 7.13. Some care is required to obtain our postulated parameters. We could get a quasi-NC^4 algorithm by applying the results of [MV97, Section 6.1] to compute the

[13] Here, bipartite graphs on n vertices correspond to subsets of $\{1, 2, ..., (n/2)^2\}$.

determinant(s). To shave off one $\log n$ factor, we use the following Chinese remaindering method, pointed out to us by Rohit Gurjar. We first compute determinants modulo small primes; since the determinant has $2^{O(\log^3 n)}$ bits, we need as many primes (each of $O(\log^3 n)$ bits). For one prime this can be done in NC^2 [Ber84]. Then we reconstruct the true value from the remainders. Doing this for an n-bits result would be in NC^1 [BCH86], and thus for a result with $2^{O(\log^3 n)}$ bits it is in $\mathsf{quasi\text{-}NC}^3$. □

7.5 The weight function construction

For our derandomization of the Isolation Lemma we will use families of weight functions that are possible to generate obliviously, i.e., by only using the number of vertices in G. We define them below.

Definition 7.16

Given $t \geq 7$, we define the family of weight functions $\mathcal{W}(t)$ as follows. Let $w_k : \{1, 2, ..., \binom{n}{2}\} \to \mathbb{Z}$ be given by $w_k(j) = (4n^2 + 1)^j \bmod k$ for $k = 2, ..., t$. We define $\mathcal{W}(t) = \{w_k : k = 2, ..., t\}$. For brevity, we write $\mathcal{W} := \mathcal{W}(n^{20})$.

Recall that throughout Part II we are dealing with a fixed graph $G = (V, E)$ on n vertices. We consider E to be a subset of $\{1, 2, ..., \binom{n}{2}\}$ and, by an abuse of notation, we identify weight functions w with their restrictions $w|_E$ to the argument set E.

In our argument we will obtain a decreasing sequence of faces (see Definition 2.9). Each face arises from the previous by minimizing over a linear objective (given by a weight function).

Definition 7.17

Let F be a face and w a weight function. The subface of F minimizing w will be called $F[w]$:

$$F[w] := \operatorname{argmin}\{\langle w, x\rangle : x \in F\}.$$

Instead of minimizing over one weight function and then over another, we can *concatenate* them in such a way that minimizing over the concatenation yields the same subface. In particular, we will argue that one just needs to try all possible concatenations of $O(\log^2 n)$ weight functions from $\mathcal{W}$ in order to find one which isolates a unique perfect matching in G (i.e., it produces a single extreme point as the minimizing subface).

Definition 7.18

For two weight functions w and w', where $w : E \to \mathbb{Z}$ and $w' \in \mathcal{W}$, we define their concatenation $w \circ w' := n^{21} w + w'$, *i.e.,*

$$(w \circ w')(e) := n^{21} \cdot w(e) + w'(e).$$

We also define $\mathcal{W}^k$ to be the set of all concatenations of k weight functions from $\mathcal{W}$, i.e.,

$$\mathcal{W}^k := \{w_1 \circ w_2 \circ ... \circ w_k : w_1, w_2, ..., w_k \in \mathcal{W}\},$$

where by $w_1 \circ w_2 \circ ... \circ w_k$ we mean $((w_1 \circ w_2) \circ ...) \circ w_k$.

Fact 7.19

We have $F[w][w'] = F[w \circ w']$.

Proof.
Both faces are integral and so we only need to show that $F[w][w'] \cap \mathbb{Z}^E = F[w \circ w'] \cap \mathbb{Z}^E$. The first set consists of matchings in F minimizing w and, among such matchings, minimizing w'. The second set consists of matchings in F minimizing $w \circ w'$. These two sets are equal because for any M:

- $(w \circ w')(M) = n^{21} \cdot w(M) + w'(M)$,
- $w' \in \mathcal{W} = \mathcal{W}(n^{20})$ implies that $0 \leq w'(M) < n^{20} \cdot \frac{n}{2} < n^{21}$ for any matching M,
- $w(M) \in \mathbb{Z}$, so that for any two matchings M_1 and M_2, $w(M_1) > w(M_2)$ implies $(w \circ w')(M_1) - (w \circ w')(M_2) = n^{21}\,(w(M_1) - w(M_2)) + (w'(M_1) - w'(M_2)) > 0$.

Hence, the ordering given by $w \circ w'$ is the same as the lexicographic ordering given by (w, w'). □

Chapter 8

A Quasi-NC Algorithm for Perfect Matching

8.1 Introduction

The perfect matching problem is a fundamental question in graph theory. See Section 2.4 for its definition. Work on matchings has contributed to the development of many core concepts of modern computer science, including linear-algebraic, probabilistic and parallel algorithms. However, we still do not have full understanding of the deterministic parallel complexity of matching. Namely, we do not know whether matching has an algorithm that runs in polylogarithmic time on polynomially many processors, i.e., whether it is in the class NC (see Section 7.1). We know that such algorithms exist if we allow randomization (see Section 7.2). In one such algorithm, due to Mulmuley, Vazirani and Vazirani [MVV87], the only randomized component is the celebrated Isolation Lemma (see also Section 7.3):

Lemma 8.1 (*Isolation Lemma for matchings*)

Consider a graph $G = (V, E)$. Suppose we define a weight function $w : E \to \{1, 2, ..., 2|E|\}$ by selecting each $w(e)$ for $e \in E$ independently and uniformly at random. Then with probability at least $1/2$, there is a unique minimum-weight matching, i.e., w is isolating.

If one were able to derandomize Lemma 8.1 by computing an isolating weight function in NC (that has polynomially bounded values), this would imply that matching is in NC. However, derandomizing the Isolation Lemma turns out to be a challenging open question. It has been done for certain classes of graphs: strongly chordal [DK98], planar bipartite [DKR10, TV12], or graphs with a small number of perfect matchings [GK87, AHT07]. More generally, there has been much interest in obtaining NC algorithms for the perfect matching problem on restricted graph classes (not necessarily using the Isolation Lemma), e.g.: regular bipartite [LPV81], planar bipartite [MN89, MV00], planar [AV18, San18], P_4-tidy [Par98], dense [DHK93], convex bipartite [DS84], claw-free [CNN89], incomparability graphs [KVV85]. The general set-family setting of the Isolation Lemma (Lemma 7.11) is also related to circuit lower bounds and polynomial identity testing [AM08].

Recently, in a major development, Fenner, Gurjar and Thierauf [FGT16] have almost derandomized the Isolation Lemma for bipartite graphs. Namely, they define a family of weight

functions that can be computed obliviously (only using the number n of vertices) and prove that for any bipartite graph, one of these functions is isolating (see Theorem 7.12). Because their family has quasi-polynomial size and the weights are quasi-polynomially large, this has placed the perfect bipartite matching problem in the class quasi-NC (see Corollary 7.13).

Nevertheless, the general-graph setting of the derandomization question (either using the Isolation Lemma or not) remained open. In general, the best known upper bound on the size of uniform circuits with polylogarithmic depth was exponential.

The main result of this part of the thesis is the following:

Theorem 7.14 (*[ST17]*)

For any number n, we can in quasi-NC *construct $n^{O(\log^2 n)}$ weight functions on $\{1, 2, ..., \binom{n}{2}\}$ with weights bounded by $n^{O(\log^2 n)}$ such that for any graph on n vertices, one of these weight functions is isolating.*

Theorem 7.14 implies that matching in general graphs is in the class quasi-NC (see Corollary 7.15). The present chapter is devoted to the proof of Theorem 7.14. We remark that the implied algorithm is very simple. The complexity lies in the analysis, i.e., proving that one of the weight functions is isolating (see Theorem 8.17).

In what follows, we first give an overview of the framework in [FGT16] for bipartite graphs. We then explain how we extend the framework to general graphs. Due to the more complex structure of perfect matchings in general graphs, we need several new ideas. In particular, we exploit the structural properties of the perfect matching polytope that we have developed in Section 2.6.

See Section 8.1.4 for the outline of the rest of this chapter. Conclusions and remarks on future work can be found in Chapter 9.

8.1.1 Isolation in bipartite graphs

In this section we shortly discuss the elegant framework introduced by Fenner, Gurjar and Thierauf [FGT16], which we extend to obtain our result.

If a weight function w is *not* isolating, then there exist two minimum-weight perfect matchings, and their symmetric difference consists of alternating cycles. In each such cycle, the total weight of edges from the first matching must be equal to the total weight of edges from the second matching (as otherwise we could obtain another matching of lower weight). The difference between these two total weights is called the *circulation* of the cycle.[14] By the above, if all cycles have nonzero circulation, then w is isolating. It is known how to obtain weight functions which satisfy a polynomial number of such non-equalities (see Lemma 8.5). However, a graph may have an exponential number of cycles.

A key idea of [FGT16] is to build the weight function in $\log n$ rounds. In the first round, we find a weight function with the property that each cycle *of length 4* has nonzero circulation. This is possible since there are at most n^4 such cycles. We apply this function and from now on consider only those edges which belong to a minimum-weight perfect matching. Crucially, it turns out that in the subgraph obtained this way, all cycles of length 4 have disappeared – this follows from the simple structure of the bipartite perfect matching polytope (a face is simply the bipartite matching polytope of a subgraph) and fails to hold for general graphs. In the second round, we start from this subgraph and apply another weight function which ensures that all

[14] This has no relation to the term *circulation* from Part I.

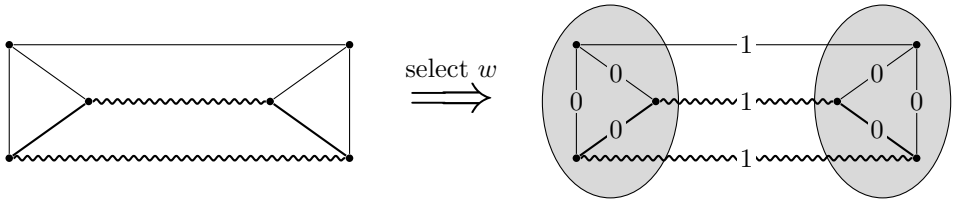


Figure 8.1: The main difficulty of derandomizing the Isolation Lemma for general graphs as compared to bipartite graphs. In trying to remove the bold cycle, we select a weight function w such that the circulation of the cycle is $1-0+1-0 \neq 0$. By minimizing over w we obtain a new, smaller subface – the convex hull of perfect matchings of weight 1 – but every edge of the cycle is still present in one of these matchings. The cycle has only been eliminated in the following sense: it can no longer be obtained in the symmetric difference of two matchings in the new face (since none of them select both swirly edges). The vertex sets drawn in gray represent the new tight odd-set constraints that describe the new face (indeed: for a matching to have weight 1, it must take only one edge from the boundary of a gray set). We will say that the cycle does not *respect* the gray vertex sets (see Section 8.2).

even cycles of length up to 8 have nonzero circulation (one proves that there are again $\leq n^4$ many since the graph contains no 4-cycles). Again, these cycles disappear from the next subgraph, and so on. After $\log n$ rounds, the current subgraph has no cycles, i.e., it is a perfect matching. The final weight function is obtained by combining the $\log n$ polynomial-sized weight functions. To get a parallel algorithm, we need to simultaneously try each such possible combination, of which there are quasi-polynomially many.

This result has later been generalized by Gurjar and Thierauf [GT17] to the linear matroid intersection problem – a natural extension of bipartite matching. From the work of Narayanan, Saran and Vazirani [NSV94], who gave an RNC algorithm for that problem (also based on computing a determinant), it again follows that derandomizing the Isolation Lemma implies a quasi-NC algorithm.

8.1.2 Challenges of non-bipartite graphs

We find it useful to look at the method explained in the previous section from a polyhedral perspective (also used by [GT17]). We begin from the set of all perfect matchings, of which we take the convex hull: the perfect matching polytope. After applying the first weight function, we want to consider only those perfect matchings which minimize the weight; this is exactly the definition of a face of the polytope. In the bipartite case, any face was characterized by just taking a subset of edges (i.e., making certain constraints $x_e \geq 0$ tight), so we could simply think about recursing on a smaller subgraph. This was used to show that any cycle whose circulation has been made nonzero will not retain all of its edges in the next subgraph. The progress we made in the bipartite case could be measured by the girth (the minimum length of a cycle) of the current subgraph, which doubled as we moved from face to subface.

Unfortunately, in the non-bipartite case, the description of the perfect matching polytope is more involved (see Theorem 2.4). Namely, moving to a new subface may also cause new tight *odd-set constraints* to appear. These, also referred to as odd cut constraints, require that, for an odd set $S \subseteq V$ of vertices, exactly one edge of a matching should cross the cut defined by S.

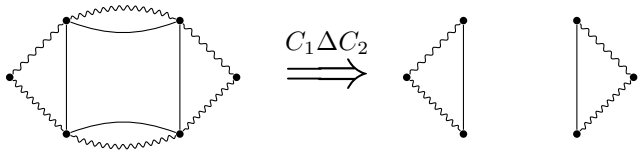


Figure 8.2: The minor difficulty of derandomizing the Isolation Lemma for general graphs as compared to bipartite graphs: two even cycles whose symmetric difference contains no even cycle.

This complicates our task, as depicted in Figure 8.1 (the same example was given by [FGT16] to demonstrate the difficulty of the general-graph case). Now a face is described by not only a subset of edges, but also a family of tight odd-set constraints. Thus we can no longer guarantee that any cycle whose circulation has been made nonzero will disappear from the support of the new face, i.e., the set of edges that appear in at least one perfect matching in this face. Our idea of what it means to remove a cycle thus needs to be refined (see Section 8.2), as well as the measure of progress we use to prove that a single matching is isolated after $\log n$ rounds (see Section 8.3). We need several new ideas, which we outline in Section 8.1.3.

Another difficulty, of a more technical nature, concerns the counting argument used to prove that a graph with no cycles of length at most λ contains only polynomially many cycles of length at most 2λ. In the bipartite case, the symmetric difference of two even cycles contains a simple cycle, which is also even. In addition, one can show that if the two cycles share many vertices, then the symmetric difference must contain one such even cycle that is short (of length at most λ) and thus should not exist. This enables a simple checkpointing argument to bound the number of cycles of length at most 2λ, assuming that no cycle of length at most λ exists. Now, in the general case we are still only interested in removing *even* cycles, but the symmetric difference of two even cycles may not contain an even simple cycle (see Figure 8.2). This forces us to remove not only even simple cycles, but all even walks, which may contain repeated edges (we call these *alternating circuits* – see Definition 8.2), and to rework the counting scheme, obtaining a bound of n^{17} rather than n^4 (see Lemma 8.21). Moreover, instead of simple graphs, we work on node-weighted multigraphs, which arise by contracting certain tight odd-sets.

8.1.3 Our approach

This section is a high-level, idealized explanation of how to deal with the main difficulty (see Figure 8.1); we ignore the more technical one in this description.

Removing cycles which do not cross a tight odd-set As discussed in Section 8.1.2, when moving from face to subface we cannot guarantee that, for each even cycle whose circulation we make nonzero, one of its edges will be absent from the support of the new face. However, this will at least be true for cycles that do not cross any odd-set tight for the new face. This is because if there are no tight odd-set constraints, then our faces behave as in the bipartite case. So, intuitively, if we only consider those cycles which do not cross any tight set, then we can remove them using the same arguments as in that case. This implies, by the same argument as in Section 8.1.1, that if we apply $\log n$ weight functions in succession, then the resulting face will not contain in its support any even cycle that crosses no tight odd-set. This is less than we need, but a good first step. If, at this point, there were no tight sets, then we would be done, as we

would have removed all cycles. However, in general there will still be cycles crossing tight sets, which make our task more difficult.

Decomposition into two subinstances To deal with the tight odd-sets, we will make use of two crucial properties. The first property is easy to see: once we fix the single edge e in the matching which crosses a tight set S, the instance breaks up into two *independent* subinstances. That is, every perfect matching which contains e is the union of: the edge e, a perfect matching on the vertex set S (ignoring the S-endpoint of e), and a perfect matching on the vertex set $V \setminus S$ (ignoring the other endpoint of e).

This will allow us to employ a divide-and-conquer strategy: to isolate a matching in the entire graph, we will take care of both subinstances and of the cut separating them. We formulate the task of dealing with such a subinstance (a subgraph induced on an odd-cardinality vertex set) as follows: we want that, once the (only) edge of a matching which lies on the boundary of the tight odd-set is fixed, the entire matching inside the set is uniquely determined. We will then call this set *contractible* (see Definition 8.7).[15] This can be seen as a generalization of our isolation objective to subgraphs with an odd number of vertices. If we can get that for the tight set and for its complement, then each edge from the cut separating them induces a unique perfect matching in the graph. Therefore there are at most n^2 perfect matchings left in the current face. Now, in order to isolate the entire graph, we only need a weight function w which assigns different weights to all these matchings. This demand can be written as a system of n^4 linear non-equalities on w, and we can generate a weight function w satisfying all of them (see Lemma 8.5).

While it is not clear how to continue this scheme beyond the first level or why we could hope to have a low depth of recursion, we will soon explain how we utilize this basic strategy.

Laminarity The second crucial property that we utilize is that the family of odd-set constraints tight for a face exhibits good structural properties. Namely, it is known that a *laminar* family of odd sets is enough to describe any face (see Lemma 2.15). Recall that a family of sets is laminar if any two sets in the family are either disjoint or one is a subset of the other (see Figure 8.4 for an example). This enables a scheme where we use this family to make progress in a bottom-up fashion. This is still challenging as the family does not stay fixed as we move from face to face. The good news is that it can only increase: whenever a new tight odd-set constraint appears which is not spanned by the previous ones, we can always add an odd-set to our laminar family.

Chain case To get started, let us first discuss the special case where the laminar family of tight constraints is a chain, i.e., an increasing sequence of odd-sets $S_1 \subsetneq S_2 \subsetneq ... \subsetneq S_\ell$. We remark that this will be an informal and simplified description of the proof of Lemma 8.23. For this introduction, assume $\ell = 8$ as depicted in Figure 8.3. Denote by $U_1, ..., U_8$ the *layers* of this chain, i.e., $U_1 = S_1$ and $U_p = S_p \setminus S_{p-1}$ for $p = 2, 3, ..., 8$. Suppose this chain describes the face that was obtained by applying the $\log n$ weight functions as above that remove all even cycles that do not cross a tight set. Then there is no cycle that lies inside a single layer U_p.

Notice that every layer U_p is of even size and it touches two boundaries of tight odd-sets: S_{p-1} and S_p (that is, $\delta(U_p) \subseteq \delta(S_{p-1}) \cup \delta(S_p)$). Any perfect matching in the current face will have one edge from $\delta(S_{p-1})$ and one edge from $\delta(S_p)$ (possibly the same edge), therefore U_p will have two (or zero) boundary edges in the matching. An exception is U_1, which is odd, only touches S_1 and will have one boundary edge in the matching. This motivates us to generalize

[15] This has no relation to the term *contractible* from Part I.

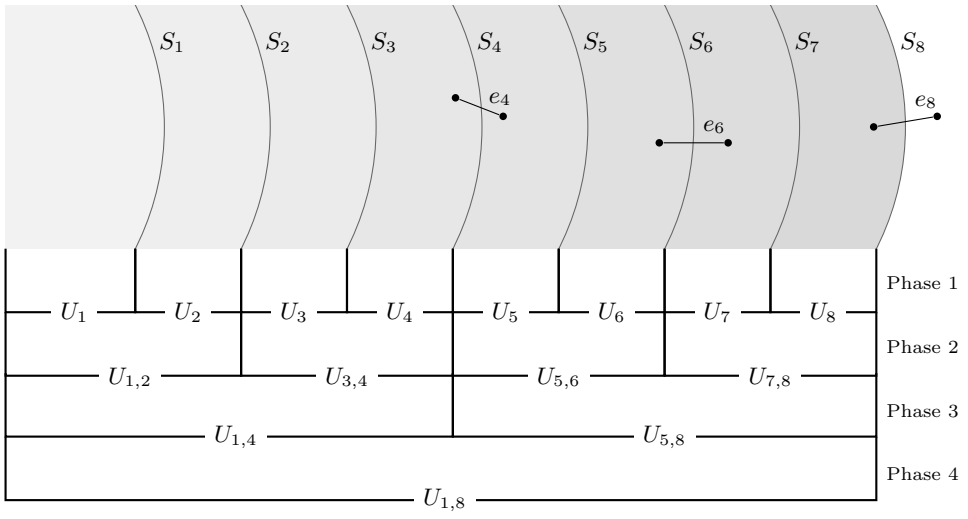


Figure 8.3: Example of a chain consisting of 8 tight sets, and our divide-and-conquer argument.

our isolation objective to layers as follows: we say that a layer U_p is *contractible* if choosing an edge from $\delta(S_{p-1})$ and an edge from $\delta(S_p)$ induces a unique matching inside U_p. This definition naturally extends to layers of the form $S_r \setminus S_{p-1} = U_p \cup U_{p+1} \cup ... \cup U_r$, which we will denote by $U_{p,r}$.

Recall that we have ensured that there is no cycle that lies inside a single layer $U_p = U_{p,p}$. It follows that these layers are contractible. This is because two different matchings (but with the same boundary edges) in the current face would induce an alternating cycle in their symmetric difference.

Let us say that this was the first phase of our approach (see Figure 8.3). In the second phase, we want to ensure contractibility for double layers: $U_{1,2}$, $U_{3,4}$, $U_{5,6}$ and $U_{7,8}$. In general, we double our progress in each phase: in the third one we deal with the quadruple layers $U_{1,4}$ and $U_{5,8}$, and in the fourth phase we deal with the octuple layer $U_{1,8}$.

Let us now describe a single phase. Take e.g. the layer $U_{5,8}$ and two boundary edges $e_4 \in \delta(S_4)$ and $e_8 \in \delta(S_8)$ (see Figure 8.3); we want to have only a unique matching in $U_{5,8}$ including these edges. Now we realize our divide-and-conquer approach. Note that the layers $U_{5,6}$ and $U_{7,8}$ have already been dealt with (made contractible) in the previous phase. Therefore, for each choice of boundary edge $e_6 \in \delta(S_6)$ for the matching, there is a unique matching inside both of these layers. Just like previously, this implies that there are only n^2 matchings using e_4 and e_8 in the layer $U_{5,8}$, and we can select a weight function that isolates one of them. We actually select only one function per phase, which works simultaneously for all layers $U_{p,r}$ in this phase (here: $U_{1,4}$ and $U_{5,8}$) and all pairs of boundary edges e_{p-1} and e_r.

By generalizing this strategy (from $\ell = 8$ to arbitrary ℓ) in the natural way, we can deal with any chain in $\log \ell \leq \log n$ phases, even if it consists of $\Omega(n)$ tight sets. We remark that, in the general proof, we do not quite use a binary tree structure like in the example. Instead, in the t-th phase, we deal with all layers $U_{p,r}$ having $1 \leq p \leq r \leq \ell$ with $r - p \leq 2^{t-1} - 1$. This makes our proof simpler if ℓ is not a power of two.

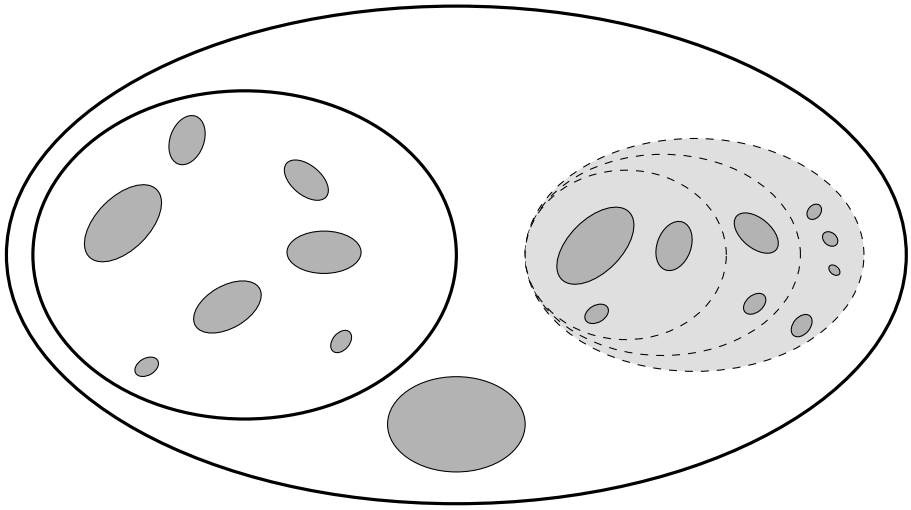


Figure 8.4: Example of a general laminar family.
Dark-gray sets are of size at most λ and thus contractible.
Dashed sets are of size more than λ but at most 2λ; they must form chains (due to the cardinality constraints). We make them contractible in the first step. Then we contract them (so now all light-gray and dark-gray sets are contracted).
Thick sets are of size more than 2λ. For the second step, we erase the edges on their boundaries. Then we remove cycles of length up to 2λ from the resulting instance (the *contraction*), which has no tight odd-sets (and no cycles of length up to λ).

General case Of course, there is no reason to expect that the laminar family of tight cuts we obtain after applying the initial $\log n$ weight functions will be a chain. It also does not seem easy to directly generalize our inductive scheme from a chain to an arbitrary family. Therefore we put forth a different progress measure, which allows us to make headway even in the absence of such a favorable odd-set structure.

Since a laminar family can be represented as a tree, we might think about a bottom-up strategy based on it; however, we cannot deal with its nodes level-by-level, since it may have height $\Omega(n)$ and we can only afford $\text{poly}(\log n)$ many phases. Instead, we will first deal with all tight odd-sets of size up to 4, then up to 8, then up to 16 and so on, by making them contractible. At the same time, we also remove all even cycles of length up to 4, then up to 8 and so on.[16] These two components of our progress measure, which we call λ-goodness, are mutually beneficial, as we will see below.

Making odd-sets contractible enables us not only to achieve progress, but also to simplify our setting. A contractible tight set can be, for our purposes, thought of as a single vertex – much like a blossom in Edmonds' algorithm. This is because such a set has exactly one boundary edge in a perfect matching (as does a vertex), and choosing that edge determines the matching in the interior. As the name suggests, we will contract such sets.

Suppose that our current face is already λ-good. Roughly, this means that we have made odd-sets of size up to λ (which we will call small) contractible and removed cycles of length up

[16] As discussed in Section 8.1.2 and Figure 8.1, the meaning of the term *remove* needs to be refined, as we cannot hope to always delete an edge of the cycle from the support of the current face.

to λ. Now we want to obtain a face which is 2λ-good.

The first step is to make odd-sets of size up to 2λ contractible. Let us zoom in on one such odd-set – a maximal set of size at most 2λ (see the largest dashed set in Figure 8.4). Once we have contracted all the small sets into single vertices, all interesting sets are now of size more than λ but at most 2λ, and any laminar family consisting of such sets must be a chain, since a set of such size cannot have two disjoint subsets of such size (see Figure 8.4). But this is the chain case that we have already solved!

Having made odd-sets of size up to 2λ contractible, we can contract them. The second step is now to remove cycles of length up to 2λ. However, here we do not need to care about those cycles which cross an odd-set S of size larger than 2λ – the reason being, roughly, that in our technical arguments we define the length of a cycle based on the sizes of sets that it crosses, and thus such a cycle actually becomes longer than 2λ. In other words, we can think about removing cycles of length up to 2λ from a version of the input graph where all odd-sets of size up to 2λ have been contracted and all larger ones have had their boundaries erased (see Figure 8.4). We call this version the *contraction* (see Definition 8.11). Our λ-goodness progress measure (see Definition 8.13) is actually defined in terms of cycles in the contraction.

Now the second step is easy: we just need to remove all cycles of length up to 2λ from the contraction, which has no tight odd-sets and no cycles of length up to λ – a simple scenario, already known from the bipartite case. Applying one weight function is enough to do this.

Finally, what does it mean for us to remove a cycle? When we make a cycle's circulation nonzero, it is then eliminated from the new face in the following sense: either one of its edges disappears from the support of the face (recall that this is what always happened in the bipartite case), or a new tight odd-set appears, with the following property: the cycle crosses the set with fewer (or more) even-indexed edges than odd-indexed edges (see the example in Figure 8.1). In short, we say that the cycle does not *respect* the new face (see Section 8.2). This notion of removal makes sense when viewed in tandem with the contraction, because once a cycle crosses a set in the laminar family, there are two possibilities in each phase: either this set is large – then its boundary is not present in the contraction, which cancels the cycle, or it is small – then it is contracted and the cycle also disappears (for somewhat more technical reasons).

To reiterate, our strategy is to simultaneously remove cycles up to a given length and make odd-sets up to a given size contractible. We can do this in $\log n$ phases. In each such phase we need to apply a sequence of $\log n$ weight functions in order to deal with a chain of tight odd-sets (as outlined above). In all, we are able to isolate a perfect matching in the entire graph using a sequence of $O(\log^2 n)$ weight functions with polynomially bounded weights.

8.1.4 Outline

The rest of this chapter is organized as follows. In Section 8.2 we define alternating circuits (our generalization of alternating cycles), discuss what it means for such a circuit to respect a face, and develop our tools for circuit removal. In Section 8.3 we introduce our measure of progress (λ-goodness), contractible sets and the contraction multigraph. We also state Theorem 8.17, which implies our main result. Finally, in Section 8.4 we prove our key technical theorem: that applying $\log_2 n + 1$ weight functions allows us to make progress from λ-good to 2λ-good. See Chapter 9 for conclusions and remarks on future work.

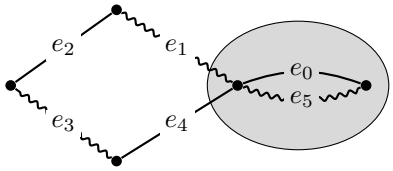


Figure 8.5: An example of an alternating circuit C of length 6 with indicator vector $(\pm\mathbb{1})_C = \sum_{i=0}^{5}(-1)^i \mathbb{1}_{e_i} = -\mathbb{1}_{e_1} + \mathbb{1}_{e_2} - \mathbb{1}_{e_3} + \mathbb{1}_{e_4}$ (since $+\mathbb{1}_{e_0}$ and $-\mathbb{1}_{e_5}$ cancel out). Also note that $\langle (\pm\mathbb{1})_C, \mathbb{1}_{\delta(S)} \rangle = 0$ for the tight set S depicted in gray.

8.2 Alternating circuits and respecting a face

In this section we introduce two notions which are vital for our approach. Before giving the formal definitions, we give an informal motivation.

Our argument is centered around *removing* even cycles. As discussed in Section 8.1.2 and Figure 8.1, the meaning of this term in the non-bipartite case needs to be more subtle than just "removing an edge of the cycle".

In order to deal with a cycle, we find a weight function w which assigns it a nonzero circulation. Formally, given an even cycle C with edges numbered in order, define a vector $(\pm\mathbb{1})_C \in \{-1, 0, 1\}^E$ as having 1 on even-numbered edges of C, -1 on odd-numbered edges of C, and 0 elsewhere. Then, nonzero circulation means that $\langle (\pm\mathbb{1})_C, w \rangle \neq 0$. Now, in the bipartite case, if such a cycle survived in the new face $F[w]$ (recall that this is the face obtained by minimizing the weight function w), that is, $C \subseteq E(F[w])$, then the vector $(\pm\mathbb{1})_C$ could be used to obtain a point in the face F with lower w-weight than the points in $F[w]$, a contradiction. This argument is possible because of the simple structure of the bipartite perfect matching polytope.

In the non-bipartite case, it is not enough that $C \subseteq E(F[w])$ in order to obtain such a point (and a contradiction). It is also required that, if the cycle C enters a tight odd-set S on an even-numbered edge, it exits it on an odd-numbered edge (and vice versa). This makes intuitive sense: if C were obtained from the symmetric difference of two perfect matchings which both have exactly one edge crossing S, then C would have this property. Formally, we require that $\langle (\pm\mathbb{1})_C, \mathbb{1}_{\delta(S)} \rangle = 0$ for each $S \in \mathcal{S}(F[w])$. If C satisfies these two conditions, i.e., that $C \subseteq E(F[w])$ and that $\langle (\pm\mathbb{1})_C, \mathbb{1}_{\delta(S)} \rangle = 0$ for every $S \in \mathcal{S}(F[w])$, then we say that C *respects* the face $F[w]$. The notion of respecting a face exactly formalizes what is required to obtain a contradictory point as above (see the proof of Lemma 8.4).

In other words, if we assign a nonzero circulation to a cycle, then it will not respect the new face, and this is what is now meant by removing a cycle.

To deal with the second, more technical difficulty discussed in Section 8.1.2, we need to remove not only simple cycles of even length, but also walks with repeated edges. However, we would run into problems if we allowed all such walks (up to a given length). Consider for example a walk C of length 2; such a walk traverses an edge back and forth. It is impossible to assign a nonzero circulation to C, because its vector $(\pm\mathbb{1})_C$ is zero. We overcome this technicality by defining alternating circuits to be those even walks whose vector $(\pm\mathbb{1})_C$ is nonzero (see Figure 8.5 for an example). For generality, we also formulate the definition of respect in terms of the vector $(\pm\mathbb{1})_C$.

Definition 8.2

Let $C = (e_0, ..., e_{k-1})$ be a nonempty cyclic walk of even length k.

- *We define the* alternating indicator vector $(\pm\mathbb{1})_C$ *of C to be $(\pm\mathbb{1})_C = \sum_{i=0}^{k-1}(-1)^i \mathbb{1}_{e_i}$, where $\mathbb{1}_e \in \mathbb{R}^E$ is the indicator vector having 1 on position e and 0 elsewhere.*
- *We say that C is an* alternating circuit *if its alternating indicator vector is nonzero. We also refer to C as an* alternating (simple) cycle *if it is an alternating circuit that visits every vertex at most once.*
- *When talking about a graph with node-weights, the node-weight of an alternating circuit is the sum of all node-weights of visited vertices (with multiplicities if visited multiple times).*

We remark that $(\pm\mathbb{1})_C$ does not need to have all entries -1, 0 or 1 since edges can repeat in C.

Definition 8.3

We say that a vector $y \in \mathbb{Z}^E$ respects *a face F if:*

- $\mathrm{supp}(y) \subseteq E(F)$, *and*
- *for each $S \in \mathcal{S}(F)$ we have $\langle y, \mathbb{1}_{\delta(S)} \rangle = 0$.*

Furthermore, we say that an alternating circuit C respects a face F if its alternating indicator vector $(\pm\mathbb{1})_C$ respects F.

Clearly, if $F' \subseteq F$ are faces and a vector respects F', then it also respects F.

Now we argue that we can remove an alternating circuit by assigning it a nonzero circulation. The proof of this lemma (which generalizes Lemma 3.2 of [FGT16]) motivates Definition 8.3.

Lemma 8.4

Let $y \in \mathbb{Z}^E$ be a vector and F a face. If $w : E \to \mathbb{R}$ is such that $\langle y, w \rangle \neq 0$, then y does not respect the face $F' = F[w]$.

Proof.

Suppose towards a contradiction that y respects F'. Assume that $\langle w, y \rangle < 0$ (otherwise use $-y$ in place of y). We pick $x \in F'$ to be the average of all extreme points of F', so that the constraints of PM which are tight for x are exactly those which are tight for F'. Select $\varepsilon > 0$ very small. Then $\langle x + \varepsilon y, w \rangle < \langle x, w \rangle$, which will contradict the definition of $F' = \mathrm{argmin}\{\langle w, x \rangle : x \in F\}$ once we show that $x + \varepsilon y \in F$. We show that $x + \varepsilon y \in F' \subseteq F$ by verifying:

- If $e \in E(F')$ (i.e., e is an edge with $x_e > 0$), then $(x + \varepsilon y)_e = x_e + \varepsilon y_e \geq 0$ if ε is chosen small enough.
- If $e \in E \backslash E(F')$ (i.e., e is an edge with $x_e = 0$), then from y respecting F' we get $e \notin \mathrm{supp}(y)$ and so $(x + \varepsilon y)_e = 0$.
- If $S \notin \mathcal{S}(F')$ is an odd set not tight for F', i.e., $\langle x, \mathbb{1}_{\delta(S)} \rangle > 1$, then $\langle x + \varepsilon y, \mathbb{1}_{\delta(S)} \rangle = \langle x, \mathbb{1}_{\delta(S)} \rangle + \varepsilon \langle y, \mathbb{1}_{\delta(S)} \rangle \geq 1$ if ε is chosen small enough.

- If $S \in \mathcal{S}(F')$ is an odd set tight for F' (this includes all singleton sets), then from y respecting F' we get $\langle y, \mathbb{1}_{\delta(S)} \rangle = 0$ and thus $\langle x + \varepsilon y, \mathbb{1}_{\delta(S)} \rangle = \langle x, \mathbb{1}_{\delta(S)} \rangle = 1$.

□

The following lemma says that we can assign nonzero circulation to many vectors at once using an oblivious choice of weight function from $\mathcal{W}$ (see Section 7.5 for the definition of $\mathcal{W}$). It is a minor generalization of Lemma 2.3 of [FGT16] and the proof remains similar. We give it for completeness.

Lemma 8.5

For any number s and for any set of s vectors $y_1, ..., y_s \in \mathbb{Z}^E \setminus \{0\}$ with the boundedness property $\|y_i\|_1 \leq 4n^2$, there exists $w \in \mathcal{W}(n^3 s)$ with $\langle y_i, w \rangle \neq 0$ for each $i = 1, ..., s$.

We usually invoke Lemma 8.5 with vectors y_i being the alternating indicator vectors of alternating circuits. Then the quantities $\langle y_i, w \rangle$ are the circulations of these circuits.

Proof.
Let $w' : E \to \mathbb{Z}$ be given by $w'(e_j) = (4n^2 + 1)^j$ for $j = 1, ..., |E|$. Then we have $\langle y_i, w' \rangle \neq 0$ for each i because the highest nonzero coefficient dominates the expression. Formally, let j' be the maximum index with $y_i(e_{j'}) \neq 0$ and suppose $y_i(e_{j'}) > 0$ (the other case is analogous). Then, because $\|y_i\|_\infty \leq \|y_i\|_1 \leq 4n^2$, we have

$$\langle y_i, w' \rangle = y_i(e_{j'})(4n^2 + 1)^{j'} + \sum_{j<j'} y_i(e_j)(4n^2 + 1)^j > (4n^2 + 1)^{j'} + \sum_{j=-\infty}^{j'-1} (-4n^2)(4n^2 + 1)^j = 0\,.$$

Let $t = n^3 s$. We want to show that there exists $k \in \{2, ..., t\}$ such that for all $i = 1, ..., s$, $\langle y_i, w_k \rangle \neq 0$. Recalling the definition of w_k (see Definition 7.16), $\langle y_i, w_k \rangle \neq 0$ is equivalent to $|\langle y_i, w' \rangle| \neq 0 \mod k$.

This will be implied if there exists $k \in \{2, ..., t\}$ such that $\prod_i |\langle y_i, w' \rangle| \neq 0 \mod k$. So there should be some $k \in \{2, ..., t\}$ not dividing $\prod_i |\langle y_i, w' \rangle|$ – equivalently, $\operatorname{lcm}(2, ..., t)$ should not divide $\prod_i |\langle y_i, w' \rangle|$. Knowing that $\prod_i |\langle y_i, w' \rangle| \neq 0$, this will follow if we have $\prod_i |\langle y_i, w' \rangle| < \operatorname{lcm}(2, ..., t)$. This is true because

$$\prod_{i=1}^{s} |\langle y_i, w' \rangle| < \left((4n^2 + 1)^{|E|+1}\right)^s < (4n^2 + 1)^{n^2 s} = 2^{n^2 s \log(4n^2+1)} < 2^{n^3 s} = 2^t < \operatorname{lcm}(2, ..., t)$$

where we used that $\operatorname{lcm}(2, ..., t) > 2^t$ for $t \geq 7$ [Nai82]. □

Lemmas 8.4 and 8.5 together imply the following:

Corollary 8.6

Let F be a face. For any finite set of vectors $\mathcal{Y} \subseteq \mathbb{Z}^E \setminus \{0\}$ with the boundedness property $\|y\|_1 \leq 4n^2$ for every $y \in \mathcal{Y}$, there exists $w \in \mathcal{W}(n^3 \cdot |\mathcal{Y}|)$ such that each $y \in \mathcal{Y}$ does not respect the face $F' = F[w]$. □

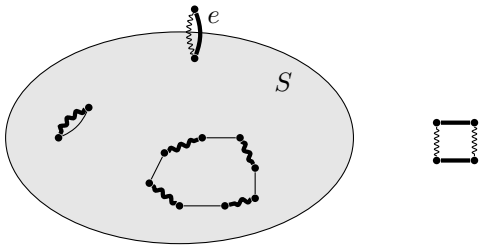


Figure 8.6: Illustration of the matching M_{12} constructed in the proof of Lemma 8.9. Straight and swirly edges denote M_1 and M_2 respectively. The thick edges denote M_{12}, which agrees with M_1 outside S and with M_2 inside S.

8.3 Contractible sets and λ-goodness

We will make progress by ensuring that larger and larger parts of the graph are "isolated" in our current face F. By "parts of the graph" we mean sets S which are tight for F. As discussed in Section 8.1.3, for such a set S, the following isolation property is desirable: once the (only) edge of a matching which lies on the boundary of S is fixed, the entire matching inside S is uniquely determined. This motivates the following definition:

Definition 8.7

*Let F be a face and let $S \in \mathcal{S}(F)$ be a tight set for F. We say that S is F-*contractible *if for every $e \in \delta(S)$ there are no two perfect matchings in F which both contain e and are different inside S.*

Note that, in the above definition, there could be no such perfect matching for certain edges $e \in \delta(S)$ (this is the case if and only if $e \notin E(F)$). Intuitively, a contractible set can be thought of as a single vertex with respect to the structure of the current face of the perfect matching polytope. The notion of contractibility enjoys the following two natural monotonicity properties:

Fact 8.8

Let $F' \subseteq F$ be two faces. If S is F-contractible, then it is also F'-contractible. □

Lemma 8.9

Let F be a face and $S \subseteq T$ two sets tight for F, i.e., $S, T \in \mathcal{S}(F)$. If T is F-contractible, then so is S.

Proof.
Let $e \in \delta(S)$. Suppose that M_1 and M_2 are two perfect matchings in F which contain e but are different inside S. We will argue that in that case there also exist two perfect matchings M_1 and M_{12} in F which contain e, are different inside S, and are equal outside of S.

Once we have that, we conclude as follows. Let f be the (only) edge in $\delta(T) \cap M_1$ (perhaps $f = e$); then also $f \in M_{12}$. Then M_1 and M_{12} are two perfect matchings in F which contain $f \in \delta(T)$ but are different inside T, contradicting that T is F-contractible.

To get the outstanding claim, we define

$$M_{12} = (M_1 \setminus E(S)) \cup (M_2 \cap E(S))$$

to be the perfect matching that agrees with M_1 on all edges not in $E(S)$ and agrees with M_2 on all edges in $E(S)$ (see Figure 8.6). To see that M_{12} is a perfect matching, notice that both M_1 and M_2 are in F and contain e. Furthermore, as $e \in \delta(S)$ for the tight set $S \in \mathcal{S}(F)$, we have that $M_1 \cap E(S)$ and $M_2 \cap E(S)$ are both perfect matchings on the vertex set S where we ignore the vertex incident to e. We can thus "replace" $M_1 \cap E(S)$ by $M_2 \cap E(S)$ to obtain the perfect matching M_{12}.

We now show that M_{12} is in the face F. Suppose the contrary. Since M_1 and M_2 are both in F, we have $M_{12} \subseteq E(F)$. Therefore, if M_{12} is not in F, we must have $|\delta(R) \cap M_{12}| > 1$ for some tight set $R \in \mathcal{S}(F)$. Since $|R|$ is odd, also $|\delta(R) \cap M|$ is odd for any perfect matching M. In particular, $|\delta(R) \cap M_{12}| \geq 3$, which contradicts

$$|\delta(R) \cap M_{12}| \leq |\delta(R) \cap M_1| + |\delta(R) \cap M_2| = 2\,,$$

where the equality holds because M_1 and M_2 are perfect matchings in F and $R \in \mathcal{S}(F)$ is a tight set. □

In our proof, we will be working with faces and laminar families which are compatible in the following sense:

Definition 8.10

Let F be a face and $\mathcal{L}$ a laminar family. If $\mathcal{L} \subseteq \mathcal{S}(F)$, i.e., all sets $S \in \mathcal{L}$ are tight for F, then we say that $(F, \mathcal{L})$ is a face-laminar pair.

Given a face-laminar pair $(F, \mathcal{L})$, we will often work with a multigraph obtained from G by contracting all small sets, i.e., those with size being at most some parameter λ (which is a measure of our progress). This multigraph will be called the *contraction* (see Figure 8.7 for an example).

In the contraction, we will also remove all boundaries of larger sets (i.e., those with size larger than λ). This is done to simulate working inside each such large set independently, because the contraction then decomposes into a collection of disconnected components, one per each large set. Because, in the contraction, each set in $\mathcal{L}$ has either been contracted or has had its boundary removed, our task is reduced to dealing with instances having no laminar sets.

Moreover, we only include those edges which are still in the support of the current face F, i.e., the set $E(F)$.

Definition 8.11

Given a face-laminar pair $(F, \mathcal{L})$ and a parameter λ (with $1 \leq \lambda \leq 2n$), we define the $(F, \mathcal{L}, \lambda)$-contraction *of G as a node-weighted multigraph as follows:*

- *the node set is the set of maximal sets of size (cardinality) at most λ in $\mathcal{L}$,*
- *each node has a node-weight equal to the size of the corresponding set,*
- *the edge set is obtained from $E(F) \setminus \bigcup_{T \in \mathcal{L}: |T| > \lambda} \delta(T)$ by contracting each of these maximal sets. That is, an edge of G maps to an edge of the contraction if it is in $E(F)$, it is not inside any of the contracted sets and it does not cross any cut defined by a set $T \in \mathcal{L} : |T| > \lambda$. Sometimes we identify edges of the contraction with their preimages in G.*

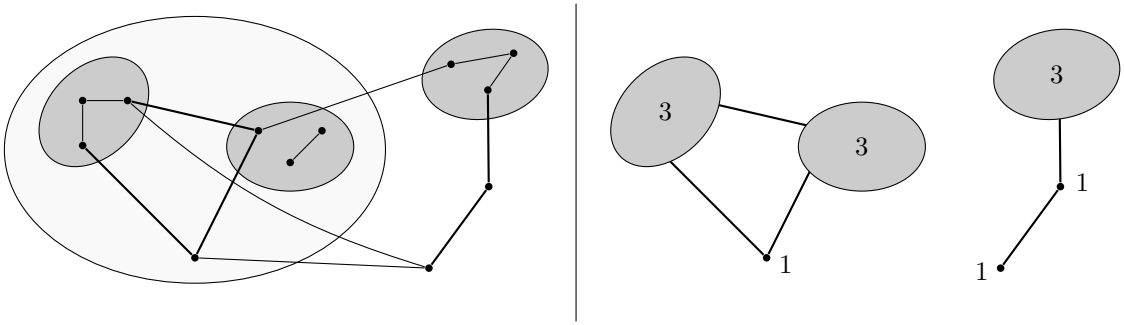


Figure 8.7: An example of the $(F, \mathcal{L}, \lambda)$-contraction of G.
On the left: a graph G and a laminar family $\mathcal{L}$. We only draw the edges in $E(F)$. We also do not draw ellipses for the singleton sets in $\mathcal{L}$. The dark-gray sets are F-contractible.
On the right: the $(F, \mathcal{L}, 4)$-contraction of G. Its vertices are labeled by their node-weights.

In the $(F, \mathcal{L}, \lambda)$-contractions arising in our arguments, we will always only contract sets $S \in \mathcal{L}$ which are F-contractible (i.e., the vertices of a contraction will always correspond to F-contractible sets). Then, a very useful property is that alternating circuits in the contraction can be lifted to alternating circuits in the entire graph G in a canonical way. This is done in the proofs of Lemmas 8.20 and 8.25.

Finally, we need the following extension of Definition 8.3 for vectors defined on the contraction.

Definition 8.12

Denote the $(F, \mathcal{L}, \lambda)$-contraction of G as H, and let $z \in \mathbb{Z}^{E(H)}$ be a vector on the edges of H. We say that z respects a subface $F' \subseteq F$ if [17]

- $\mathrm{supp}(z) \subseteq E(F')$, *and*
- *for each $S \in \mathcal{S}(F')$ which is a union of sets corresponding to vertices in $V(H)$,* [18] *we have $\langle z, \mathbb{1}_{\delta(S)} \rangle = 0$.*

As before, we say that an alternating circuit C in H respects a subface F' if its alternating indicator vector $(\pm\mathbb{1})_C \in \mathbb{Z}^{E(H)}$ respects F'.

Now we are able to define our measure of progress. On one hand, we want to make larger and larger laminar sets contractible. On the other hand, there could very well be no laminar sets, so we also proceed as in the bipartite case: remove longer and longer alternating circuits.

[17] In the following conditions we abuse notation and think of z as a vector in $\mathbb{Z}^E$ obtained by identifying each edge of H with its preimage in G and letting those edges of G without a preimage in H have value 0.
[18] That is, the maximal sets of size at most λ in $\mathcal{L}$.

Definition 8.13

Let $(F, \mathcal{L})$ be a face-laminar pair and λ a parameter (with $1 \leq \lambda \leq 2n$). We say that $(F, \mathcal{L})$ is λ-good if $\mathcal{L}$ is a maximal laminar subset of $\mathcal{S}(F)$ and:

(i) each $S \in \mathcal{L}$ with $|S| \leq \lambda$ is F-contractible,

(ii) in the $(F, \mathcal{L}, \lambda)$-contraction of G, there is no alternating circuit of node-weight at most λ.

We begin with $\lambda = 1$, which is trivial, and then show that by concatenating enough weight functions we can obtain face-laminar families which are 2-good, 4-good, 8-good, and so on. We are done once we have a λ-good family with $\lambda \geq n$. The components of this proof strategy are given in the following three claims. The first step is clear:

Fact 8.14

Let $\mathcal{L}_0$ be a maximal laminar subset of $\mathcal{S}(PM)$. Then the face-laminar pair $(\mathrm{PM}, \mathcal{L}_0)$ is 1-good. □

Note that $\mathcal{L}_0$ contains all singleton sets, i.e., $\{\{v\} : v \in V\} \subseteq \mathcal{L}_0$.

We then proceed iteratively in $\log_2 n$ rounds using the following theorem. Its proof, which constitutes the bulk of our argument, is given in Section 8.4.

Theorem 8.15

Let $(F, \mathcal{L})$ be a λ-good face-laminar pair. Then there exists a weight function $w \in \mathcal{W}^{\log_2 n+1}$ and a laminar family $\mathcal{L}' \supseteq \mathcal{L}$ such that $(F[w], \mathcal{L}')$ is a 2λ-good face-laminar pair.

We are done once λ exceeds n:

Lemma 8.16

Suppose $(F, \mathcal{L})$ is λ-good for some $\lambda \geq n$. Then $|F| = 1$.

We think that the proof of this lemma is instructive. It serves to understand and motivate Definition 8.13, and more involved versions of this argument appear in the sequel.

Proof.
Let H be the $(F, \mathcal{L}, \lambda)$-contraction of G. Also let $S_1, S_2, ..., S_k$ be all maximal sets in $\mathcal{L}$. As $\mathcal{L}$ contains all singletons, their disjoint union is V and we have $V(H) = \{S_1, ..., S_k\}$ and $E(H) = \bigcup_{i=1}^{k} \delta(S_i)$. Since $(F, \mathcal{L})$ is λ-good with $\lambda \geq n$, each set $S \in \mathcal{L}$ is F-contractible, and H contains no alternating circuit of node-weight at most λ – in particular, H contains no alternating simple cycle. Indeed, an upper bound on the node-weight of any alternating simple cycle is $|S_1| + ... + |S_k| = n \leq \lambda$.

Now we show that there is only one perfect matching in F. One direction is easy: since F is a face, it is nonempty by definition. For the other direction, let M_1 and M_2 be two perfect matchings in F. We show that $M_1 = M_2$.

Because the sets $S_1, ..., S_k$ are tight for F and no edge can possibly cross a tight odd-set of cardinality at least $\lambda \geq n$, any perfect matching (in the face F) in G induces a perfect matching in H. If the matchings induced by M_1 and M_2 were different, then their symmetric difference would contain an alternating simple cycle in H, which is impossible. So the induced matchings must be equal, i.e., $M_1 \cap \bigcup_i \delta(S_i) = M_2 \cap \bigcup_i \delta(S_i)$. Moreover, the sets $S_1, ..., S_k$ are F-contractible,

which means that, given the boundary edges, there is a unique perfect matching in F inside each S_i. This yields $M_1 = M_2$. □

Before we proceed to the proof of Theorem 8.15, let us see how Fact 8.14, Theorem 8.15, and Lemma 8.16 together give our desired result:

Theorem 8.17

There exists an isolating weight function $w \in \mathcal{W}^{(\log_2 n+1)\log_2 n}$, i.e., one with $|\mathrm{PM}[w]| = 1$.

Proof.
Let $\ell = \log_2 n$. We iteratively construct a sequence of face-laminar pairs $(F_i, \mathcal{L}_i)$ for $i = 0, 1, ..., \ell$ such that $(F_i, \mathcal{L}_i)$ is 2^i-good and $F_i = F_{i-1}[w_i]$ for some weight function $w_i \in \mathcal{W}^{\ell+1}$. We begin by setting $F_0 = \mathrm{PM}$ and $\mathcal{L}_0$ to be a maximal laminar subset of $\mathcal{S}(PM)$. By Fact 8.14, $(F_0, \mathcal{L}_0)$ is 1-good. Then for $i = 1, ..., \ell$ we use Theorem 8.15 to obtain the wanted weight function w_i along with a laminar family $\mathcal{L}_i \supseteq \mathcal{L}_{i-1}$. Finally, we have that $(F_\ell, \mathcal{L}_\ell)$ is 2^ℓ-good, so that by Lemma 8.16, $|F_\ell| = 1$.

It remains to argue that $F_\ell = \mathrm{PM}[w]$ for some $w \in \mathcal{W}^{(\ell+1)\ell}$. To do this, we proceed as in Section 7.5: define the concatenation $w' \bullet w'' := n^{21(\ell+1)}w' + w''$ for two weight functions w' and w'', where $w'' \in \mathcal{W}^{\ell+1}$. (We need to use a padding term $n^{21(\ell+1)}$ that is larger than the n^{21} of Definition 7.18 because the right-hand weight functions are now from $\mathcal{W}^{\ell+1}$ rather than from $\mathcal{W}$.) By the same reasoning as for Fact 7.19 we get that $F_\ell = \mathrm{PM}[w_1][w_2]...[w_\ell] = \mathrm{PM}[w_1 \bullet w_2 \bullet ... \bullet w_\ell]$. We put $w = w_1 \bullet w_2 \bullet ... \bullet w_\ell \in \mathcal{W}^{(\ell+1)\ell}$. □

Theorem 8.17 implies Theorem 7.14 because we have $|\mathcal{W}^{(\log_2 n+1)\log_2 n}| = |\mathcal{W}|^{(\log_2 n+1)\log_2 n} \leq n^{20(\log_2 n+1)\log_2 n}$, the values of any $w \in \mathcal{W}^{(\log_2 n+1)\log_2 n}$ are bounded by $n^{21(\log_2 n+1)\log_2 n}$, and the functions $w \in \mathcal{W}$ can be generated obliviously using only the number of vertices n.

8.4 Proof of the key Theorem 8.15: from λ-good to 2λ-good

In this section we show how to make progress (measured by the λ parameter of λ-goodness) by applying a new weight function to the current face. Our objective is to make larger sets contractible (by doubling the size threshold from λ to 2λ) and to ensure that in the new contracted graph, alternating circuits of an increased node-weight are not present. We do this by moving from the current face-laminar pair, which we call $(F_{\mathsf{in}}, \mathcal{L}_{\mathsf{in}})$, to a new face-laminar pair $(F_{\mathsf{out}}, \mathcal{L}_{\mathsf{out}})$. Both pairs have the property that the laminar family is a maximal laminar family of sets tight for the face. The new family extends the previous, i.e., $\mathcal{L}_{\mathsf{out}} \supseteq \mathcal{L}_{\mathsf{in}}$.

Our main technical tools are Theorem 8.18 and Lemma 8.23. Theorem 8.18 is used to ensure that certain alternating circuits are not present in the new contraction. It says that if our current contraction has no alternating circuits of at most some node-weight, then a single weight function $w \in \mathcal{W}$ is enough to guarantee that all alternating circuits of at most twice that node-weight do not respect the new face obtained by applying w. We call this *removing* these circuits. Lemma 8.23 is used to make sure that sets in our laminar family that are of size at most 2λ become contractible. Later, new sets will be added to the laminar family in Lemma 8.25, in such a way that these properties are maintained and that the removed alternating circuits indeed do not survive in the new contraction.

The formal structure of the proof is as follows. We begin from a λ-good face-laminar pair $(F_{\text{in}}, \mathcal{L}_{\text{in}})$. Then, using our technical tools Theorem 8.18 and Lemma 8.23, we show in Theorem 8.22 the existence of a weight function $w_{\text{out}} \in \mathcal{W}^{\log_2(n)+1}$ such that the face $F_{\text{out}} = F_{\text{in}}[w_{\text{out}}]$ satisfies two conditions which make progress on conditions (i) and (ii) of λ-goodness:

(i)' For each $S \in \mathcal{L}_{\text{in}}$ with $|S| \leq 2\lambda$, S is F_{out}-contractible.

(ii)' In the $(F_{\text{out}}, \mathcal{L}_{\text{in}}, 2\lambda)$-contraction of G, there is no F_{out}-respecting alternating circuit of node-weight at most 2λ.

This gives us the wanted face F_{out} and weight function w_{out}. Finally, in Lemma 8.25 we show that extending the laminar family $\mathcal{L}_{\text{in}}$ to a maximal laminar family $\mathcal{L}_{\text{out}}$ (of sets tight for the new face) yields a 2λ-good pair $(F_{\text{out}}, \mathcal{L}_{\text{out}})$. This finishes the proof of Theorem 8.15.

8.4.1 Removing alternating circuits

This section is devoted to the proof of Theorem 8.18, which is a technical tool we use to remove alternating circuits of size between λ and 2λ from the contraction.

Theorem 8.18

Consider a face-laminar pair $(F, \mathcal{L})$ such that each $S \in \mathcal{L}$ with $|S| \leq \beta$ is F-contractible (for a parameter β). Denote by H the $(F, \mathcal{L}, \beta)$-contraction of G. If H has no alternating circuit of node-weight at most λ, then there exists $w \in \mathcal{W}$ such that H has no $F[w]$-respecting alternating circuit of node-weight at most 2λ.

We begin with a simple technical fact: to verify that a vector respects a face, it is enough to check this for a maximal laminar family of tight constraints.

Lemma 8.19

Consider a face F and a vector $y \in \mathbb{Z}^E$. Let $\mathcal{L}$ be a maximal laminar subset of $\mathcal{S}(F)$. If for each $S \in \mathcal{L}$ we have $\langle y, \mathbb{1}_{\delta(S)} \rangle = 0$, then the same holds for all $S \in \mathcal{S}(F)$.

Proof.
The proof is similar to that of Lemma 2.16, but even simpler. As $\mathcal{L}$ is a maximal laminar subset of $\mathcal{S}(F)$, Lemma 2.15 says that $\operatorname{span}(\mathcal{L}) = \operatorname{span}(\mathcal{S}(F))$. In other words, for any $S \in \mathcal{S}(F)$ we can write $\mathbb{1}_{\delta(S)}$ as a linear combination $\sum_{L \in \mathcal{L}} \mu_L \mathbb{1}_{\delta(L)}$ for some coefficients $(\mu_L)_{L \in \mathcal{L}}$. Hence

$$\langle y, \mathbb{1}_{\delta(S)} \rangle = \left\langle y, \sum_{L \in \mathcal{L}} \mu_L \mathbb{1}_{\delta(L)} \right\rangle = \sum_{L \in \mathcal{L}} \mu_L \langle y, \mathbb{1}_{\delta(L)} \rangle = 0.$$

□

Now we prove a lemma which reduces the task of removing an alternating circuit in H to that of removing a vector defined on the edges of G, which we can do using Corollary 8.6. Throughout this section, F, $\mathcal{L}$ and H are as in the statement of Theorem 8.18. Recall that the vertices of H are elements of $\mathcal{L}$, i.e., sets of vertices, and so by $S \in V(H)$ we mean the set $S \in \mathcal{L}$ that corresponds to a vertex in H.

Lemma 8.20

Let $z \in \mathbb{Z}^{E(H)}$ be a nonzero vector on the edges of H satisfying $\langle z, \mathbb{1}_{\delta(S)} \rangle = 0$ for each $S \in V(H)$. Then there exists a nonzero vector $y \in \mathbb{Z}^E$ such that for any face $F' \subseteq F$ we have: if z respects F', then y respects F'. We also have $\|y\|_1 \leq n\|z\|_1$.

We remark that y does not depend on F'.

Proof.
We consider z as a vector $z \in \mathbb{Z}^E$ by identifying each edge of H with its preimage in G and letting those edges of G without a preimage in H have value 0. We may assume $\text{supp}(z) \subseteq E(F)$; otherwise z cannot respect F' (see Definition 8.12) and thus we are done by outputting any y.

The proof idea is to extend z to a vector $y \in \mathbb{Z}^E$ which resembles an alternating indicator vector. We do this in a canonical way so that if this extension does not respect F', then it must be because z itself does not respect F'.

To this end, we do the following for each $S \in V(H)$: pair up the boundary edges $e \in \delta(S)$ which have $z_e > 0$ with boundary edges e which have $z_e < 0$, respecting their multiplicities as given by z. For example, if we had $\delta(S) = \{e_1, e_2, e_3\}$ with $z(e_1) = 3$, $z(e_2) = -2$ and $z(e_3) = -1$, we would get the pairs $\{(e_1, e_2), (e_1, e_2), (e_1, e_3)\}$. Such a pairing is always possible because $\langle z, \mathbb{1}_{\delta(S)} \rangle = 0$. Let $\left\{(e_i^+, e_i^-)\right\}_i$ be the multiset of pairs of edges obtained in this way across all $S \in V(H)$, and let $S_i \in V(H)$ be the set for which the pair (e_i^+, e_i^-) has been introduced. Also denote by v_i^+, v_i^- the S_i-endpoints of edges e_i^+, e_i^-.

Now, for each i we have $e_i^+, e_i^- \in \text{supp}(z) \subseteq E(F)$, and S is F-contractible, so there is a unique perfect matching M_i^+ on the vertex-induced subgraph $(S_i \setminus \{v_i^+\}, E(S_i \setminus \{v_i^+\}))$ in F (more precisely, M_i^+ is the unique perfect matching on that subgraph that extends to a matching in F), as well as a unique perfect matching M_i^- on $(S_i \setminus \{v_i^-\}, E(S_i \setminus \{v_i^-\}))$ in F. We let

$$y := z + \sum_i \left(\mathbb{1}_{M_i^+} - \mathbb{1}_{M_i^-} \right).$$

What remains now is to prove the following claim:

Claim 12

Let $F' \subseteq F$ be such that z respects F'. Then y respects F'.

Proof.
Since z respects F' (see Definition 8.12), we have $\text{supp}(z) \subseteq E(F')$. This implies that for each i, $M_i^+ \subseteq E(F')$. Indeed, since $e_i^+ \in \text{supp}(z) \subseteq E(F')$, there is a perfect matching on $(S_i \setminus \{v_i^+\}, E(S_i \setminus \{v_i^+\}))$ in F'. However, S_i is F-contractible and thus M_i^+ is the *only* such matching in F (thus also in F'). Therefore $M_i^+ \subseteq E(F')$ and analogously $M_i^- \subseteq E(F')$.

Now we check the conditions for y to respect F' (see Definition 8.3):

- We have $\text{supp}(y) = \text{supp}(z) \cup \bigcup_i \left(M_i^+ \cup M_i^-\right) \subseteq E(F')$.
- Let $T \in \mathcal{S}(F')$. We need to verify that $\langle y, \mathbb{1}_{\delta(T)} \rangle = 0$. Let $\mathcal{L}'$ be a maximal laminar subfamily of $\mathcal{S}(F')$ extending $\mathcal{L}$, i.e., $\mathcal{L} \subseteq \mathcal{L}' \subseteq \mathcal{S}(F')$. By Lemma 8.19, it is enough to verify that $\langle y, \mathbb{1}_{\delta(T)} \rangle = 0$ for $T \in \mathcal{L}'$. For a set T belonging to a laminar family which extends $\mathcal{L}$, it is not hard to see that there are two possibilities: either $T \subsetneq S$ for some $S \in V(H)$, or T is a union of sets in $V(H)$ (for recall that $V(H)$ is a partitioning of V that

consists of sets in $\mathcal{L}$). In the latter case, $\langle y, \mathbb{1}_{\delta(T)} \rangle = \langle z, \mathbb{1}_{\delta(T)} \rangle = 0$ because y equals z on edges crossing sets in $V(H)$ and because z respects F'. In the former case, we have

$$\begin{aligned}\langle y, \mathbb{1}_{\delta(T)} \rangle &= \left\langle \sum_{i:S_i=S} \left(\mathbb{1}_{e_i^+} - \mathbb{1}_{e_i^-} + \mathbb{1}_{M_i^+} - \mathbb{1}_{M_i^-} \right), \mathbb{1}_{\delta(T)} \right\rangle \\ &= \sum_{i:S_i=S} \left\langle \mathbb{1}_{e_i^+} - \mathbb{1}_{e_i^-} + \mathbb{1}_{M_i^+} - \mathbb{1}_{M_i^-}, \mathbb{1}_{\delta(T)} \right\rangle\end{aligned}$$

because these are the only edges in y's support that have an endpoint in S (other edges cannot possibly cross $T \subseteq S$). Now it is enough to show that each summand is 0.

For this, we know that $M_i^+ \cup \{e_i^+\}$ and $M_i^- \cup \{e_i^-\}$ are (partial) matchings in F' and that T is tight for F'. Therefore we have $|\delta(T) \cap \left(M_i^+ \cup \{e_i^+\}\right)| = 1$,[19] and the same holds for $M_i^- \cup \{e_i^-\}$. Therefore $\left\langle \mathbb{1}_{M_i^+ \cup \{e_i^+\}}, \mathbb{1}_{\delta(T)} \right\rangle = 1 = \left\langle \mathbb{1}_{M_i^- \cup \{e_i^-\}}, \mathbb{1}_{\delta(T)} \right\rangle$.

□

Regarding the norm: every edge (with multiplicity) in z causes less than $n/2$ new edges (a partial matching) to appear in y. Therefore $\|y\|_1 \leq (n/2+1)\|z\|_1 \leq n\|z\|_1$. □

Our second lemma gives a bound on the number of alternating circuits we need to remove. Its proof resembles that of Lemma 3.4 in [FGT16], but it is somewhat more complex, as we are dealing with a node-weighted multigraph, as well as with alternating circuits instead of simple cycles (see Section 8.2). We have made no attempt to minimize the exponent 17.

Lemma 8.21

There are polynomially many alternating circuits of node-weight at most 2λ in H, up to identifying circuits with equal alternating indicator vectors. More precisely, the cardinality of the set

$$\{(\pm\mathbb{1})_C : C \text{ is an alternating circuit in } H \text{ of node-weight at most } 2\lambda\}$$

is at most n^{17}.

Proof.
We will associate a small *signature* with each alternating circuit in H of node-weight at most 2λ, with the property that alternating circuits with different alternating indicator vectors are assigned different signatures. This will prove that the considered cardinality is at most the number of possible signatures, which is polynomially bounded.

Let $C = (e_0, e_1, ..., e_{k-1})$ be an alternating circuit in H of node-weight at most 2λ. We want to define its signature $\sigma(C)$. To streamline notation, we let v_i be the tail of e_i for $i = 0, \ldots, k-1$. Thus C is of the form

$$v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{k-2}} v_{k-1} \xrightarrow{e_{k-1}} v_0\,.$$

[19] Formally, consider a perfect matching M^+ (on G) in F' which is a superset of $M_i^+ \cup \{e_i^+\}$. Then we have $|\delta(T) \cap M^+| = 1$. But $\delta(T) \cap \left(M_i^+ \cup \{e_i^+\}\right) = \delta(T) \cap M^+$ because $T \subseteq S$.

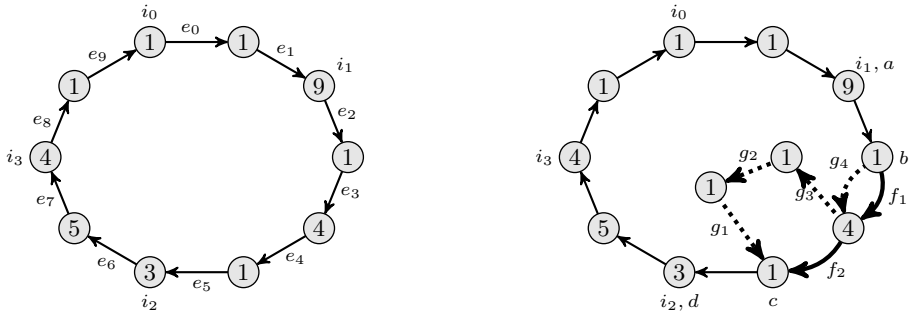


Figure 8.8: Intuition of the signature vector definition and the proof of Lemma 8.21.
On the left, each vertex is labeled by its node-weight, and the corresponding selection of i_0, i_1, i_2, i_3 is shown for $\lambda = 16$. Notice that the selected vertices partition the alternating circuit into paths; the total node-weight of internal vertices on each path is at most $\lambda/2$.
On the right we see two different alternating circuits with the same signature. They differ in that one uses f_2 and the other uses g_3, g_2, g_1. The thick edges illustrate the alternating circuit $B = (f_1, f_2, g_1, g_2, g_3, g_4)$ of node-weight at most λ which leads to the contradiction. We walk the dashed path (P_D) in reverse.

(See also Figure 8.8 for an example.) We also let $\mathrm{NW}(v_i)$ denote the node-weight of vertex v_i, and $\mathrm{in}(v_i) = e_{(i-1) \bmod k}$ and $\mathrm{out}(v_i) = e_i$ be the incoming and outgoing edges of v_i in C.[20] We now define the signature $\sigma(C)$ as the output of the following procedure:

- Let $i_0 = 0$ be the index of the first vertex in C.
- For $j = 1, 2, 3$, select $i_j \leq k$ to be the largest index satisfying $\sum_{i=i_{j-1}+1}^{i_j - 1} \mathrm{NW}(v_i) \leq \lambda/2$.
- Let $t = \max\{j : i_j < k\}$ and output the signature $\sigma(C) = ((-1)^{i_j}, \mathrm{in}(v_{i_j}), \mathrm{out}(v_{i_j}))_{j=0,1,\ldots,t}$. (Note that $t \leq 3$.)

The intuition of the signature is as follows (see also the left part of Figure 8.8). The procedure starts at the first vertex $v_{i_0} = v_0$. It then selects the farthest (according to C) vertex v_{i_1} while guaranteeing that the total node-weight of the vertices visited in-between v_{i_0} and v_{i_1} is at most $\lambda/2$. Similarly, v_{i_2} is selected to be the farthest vertex such that the total node-weight of the vertices $v_{i_1+1}, \ldots, v_{i_2-1}$ is at most $\lambda/2$, and i_3 is selected in the same fashion. The indices $i_0, i_1, \ldots, i_t$ thus partition C into paths

$$
\begin{aligned}
C_0 &= v_{i_0} \xrightarrow{e_{i_0}} v_{i_0+1} \xrightarrow{e_{i_0+1}} \ldots \xrightarrow{e_{i_1-2}} v_{i_1-1} \xrightarrow{e_{i_1-1}} v_{i_1} \\
C_1 &= v_{i_1} \xrightarrow{e_{i_1}} v_{i_1+1} \xrightarrow{e_{i_1+1}} \ldots \xrightarrow{e_{i_2-2}} v_{i_2-1} \xrightarrow{e_{i_2-1}} v_{i_2} \\
&\vdots \\
C_t &= v_{i_t} \xrightarrow{e_{i_t}} v_{i_t+1} \xrightarrow{e_{i_t+1}} \ldots \xrightarrow{e_{i_0-2}} v_{i_0-1} \xrightarrow{e_{i_0-1}} v_{i_0}
\end{aligned}
$$

[20] The functions $\mathrm{in}(v)$ and $\mathrm{out}(v)$ are not formally well-defined since they depend on the considered alternating circuit C and on which occurrence of v in the circuit we are considering, but their values will be clear from the context.

so that the total node-weight of the internal vertices on each path is at most $\lambda/2$. Indeed, for C_j with $j < 3$ this follows from the selection of i_j. For C_3 (in the case $t = 3$), by maximality of i_1, i_2 and i_3 we have

$$\underbrace{\sum_{i=i_0+1}^{i_1} \mathrm{NW}(v_i)}_{\geq \lambda/2} + \underbrace{\sum_{i=i_1+1}^{i_2} \mathrm{NW}(v_i)}_{\geq \lambda/2} + \underbrace{\sum_{i=i_2+1}^{i_3} \mathrm{NW}(v_i)}_{\geq \lambda/2} \geq \frac{3}{2}\lambda$$

and so the internal vertices of C_3 can have node-weight at most $\lambda/2$ (the total node-weight of C being at most 2λ).

We now count the number of possible signature vectors. As for each j there are at most n^2 ways of choosing the incoming edge, at most n^2 ways of choosing the outgoing edge, and i_j can have two different parities, the number of possible signatures is (summing over the choices of $t = 0, 1, 2, 3$) at most $\left(2n^2 \cdot n^2\right) + \left(2n^2 \cdot n^2\right)^2 + \left(2n^2 \cdot n^2\right)^3 + \left(2n^2 \cdot n^2\right)^4 < n^{17}$.

It remains to be shown that any two alternating circuits C and D in H of node-weight at most 2λ have different signatures if $(\pm\mathbb{1})_C \neq (\pm\mathbb{1})_D$. Suppose that $(\pm\mathbb{1})_C \neq (\pm\mathbb{1})_D$ but $\sigma(C) = \sigma(D)$. We would like to derive a contradiction with the assumption (in Theorem 8.18) that H contains no alternating circuit of node-weight at most λ. This will finish the proof.

As described above, C can be partitioned into disjoint paths $C_0, \ldots, C_t$ using its indices $i_0, \ldots, i_t$. Similarly we partition D into $D_0, \ldots, D_t$. Since these are disjoint unions, $(\pm\mathbb{1})_C \neq (\pm\mathbb{1})_D$ implies that at least one of the four subpaths must be different between C and D, in the sense that the part of the alternating indicator vector arising from that subpath is different. More formally, let b_j denote the parity (i.e., the first element) of the j-th tuple in the signatures $\sigma(C) = \sigma(D)$. Then we have $(\pm\mathbb{1})_C = \sum_{j=0}^{t} b_j \cdot (\pm\mathbb{1})_{C_j}$ and $(\pm\mathbb{1})_D = \sum_{j=0}^{t} b_j \cdot (\pm\mathbb{1})_{D_j}$. Therefore, as $(\pm\mathbb{1})_C \neq (\pm\mathbb{1})_D$, there must be a $j \in \{0, \ldots, t\}$ such that $(\pm\mathbb{1})_{C_j} \neq (\pm\mathbb{1})_{D_j}$. We will "glue" together the paths C_j and D_j to obtain another alternating circuit B.

First notice that both C_j and D_j are paths of the form $v_{i_j} = a \to b \to \cdots \to c \to d = v_{i_{(j+1) \bmod t}}$, where the segment from b to c differs between them.[21] This follows from the assumption that $\sigma(C) = \sigma(D)$. Let P_C denote the path from b to c in C_j and let P_D denote the path from b to c in D_j. As the parity fields of the signatures agree, we have that $|P_C| + |P_D|$ is even. Now let B be the cyclic walk of even length obtained by walking from b to c along the path P_C and back from c to b along the path P_D (in reverse). That is, B is of the form (see also the right part of Figure 8.8)

$$b \xrightarrow{f_1} \ldots \xrightarrow{f_{|P_C|}} c \xrightarrow{g_1} \ldots \xrightarrow{g_{|P_D|}} b\,,$$

where we let $f_1, \ldots, f_{|P_C|}$ denote the edges of the path P_C and $g_1, \ldots, g_{|P_D|}$ denote the edges of the reversed path P_D. To verify that B is an alternating circuit we need to show that its

[21] Here again we slightly abuse notation since i_j might differ between C and D; however, the vertex v_{i_j} does not, because it is the tail of e_{i_j}, which is part of the signature $\sigma(C) = \sigma(D)$. The same applies to $v_{i_{j+1 \bmod t}}$.

alternating indicator vector is nonzero:

$$
\begin{aligned}
-(\pm\mathbb{1})_B &= \sum_{i=1}^{|P_C|}(-1)^i \mathbb{1}_{f_i} + \sum_{i=1}^{|P_D|}(-1)^{|P_C|+i}\mathbb{1}_{g_i} \\
&= \underbrace{\left((-1)^0\mathbb{1}_{\mathrm{out}(a)} + \sum_{i=1}^{|P_C|}(-1)^i\mathbb{1}_{f_i} + (-1)^{|P_C|+1}\mathbb{1}_{\mathrm{in}(d)}\right)}_{=(\pm\mathbb{1})_{C_j}} \\
&\qquad + \underbrace{\left((-1)^{|P_C|+2}\mathbb{1}_{\mathrm{in}(d)} + \sum_{i=1}^{|P_D|}(-1)^{|P_C|+2+i}\mathbb{1}_{g_i} + (-1)^{|P_C|+|P_D|+3}\mathbb{1}_{\mathrm{out}(a)}\right)}_{=-(\pm\mathbb{1})_{D_j}}.
\end{aligned}
$$

The second equality is easiest to see by mentally extending B from a circuit $b \to ... \to c \to ... \to b$ to $a \to b \to ... \to c \to d \to c \to ... \to b \to a$. Also recall that $|P_C| + |P_D|$ is even. Thus we get $(\pm\mathbb{1})_B = -(\pm\mathbb{1})_{C_j} + (\pm\mathbb{1})_{D_j}$, which is nonzero by the choice of j. Finally, the node-weight of B is at most the node-weight of the internal nodes of path C_j plus the node-weight of the internal nodes of path D_j and thus at most $\lambda/2 + \lambda/2 = \lambda$.

We have thus shown that B is a nonempty cyclic walk of even length whose alternating indicator vector is nonzero – thus an alternating circuit – and whose node-weight is at most λ. This contradicts our assumption on H. □

Now we have all the tools needed to prove the main result of this section.

Proof of Theorem 8.18.
Let us fix some $w \in \mathcal{W}$. We want to articulate conditions on w which will make sure that the statement is satisfied. Then we show that some $w \in \mathcal{W}$ satisfies these conditions.

Let C be any alternating circuit in H of node-weight at most 2λ. Our condition on w will be that all such circuits C should not respect $F[w]$, i.e., that all vectors from the set

$$\mathcal{Z} := \{(\pm\mathbb{1})_C : C \text{ is an alternating circuit in } H \text{ of node-weight at most } 2\lambda\}$$

should not respect $F[w]$. We use Lemma 8.20 to transform each $z \in \mathcal{Z}$ ($z \in \mathbb{Z}^{E(H)}$) to a vector $y = y(z) \in \mathbb{Z}^E$ such that if $y(z)$ does not respect $F[w]$, then z does not respect $F[w]$. Let $\mathcal{Y} = \{y(z) : z \in \mathcal{Z}\}$. Clearly $|\mathcal{Y}| \leq |\mathcal{Z}|$ (actually $|\mathcal{Y}| = |\mathcal{Z}|$ since the mapping $z \mapsto y(z)$ is one-to-one), and $|\mathcal{Z}| \leq n^{17}$ by Lemma 8.21. Moreover, since the alternating circuits C were of node-weight at most $2\lambda \leq 4n$, we have $\|z\|_1 \leq 4n$ for $z \in \mathcal{Z}$ and $\|y\|_1 \leq 4n^2$ for $y \in \mathcal{Y}$. Now it is enough to apply Corollary 8.6 to obtain that there exists $w \in \mathcal{W}(n^3 \cdot n^{17}) = \mathcal{W}$ such that each $y \in \mathcal{Y}$ does not respect the face $F[w]$, and thus each $z \in \mathcal{Z}$ does not respect $F[w]$. □

8.4.2 The existence of a good weight function

In this section, we use Theorem 8.18 to prove the existence of a weight function defining a face F_{out} with the desired properties (so as to be the face in our 2λ-good face-laminar pair), namely:

Theorem 8.22

Let $(F_{in}, \mathcal{L}_{in})$ be a λ-good face-laminar pair. Then there exists a weight function $w_{out} \in \mathcal{W}^{\log_2(n)+1}$ such that the face $F_{out} = F_{in}[w_{out}]$ satisfies:

(i)' For each $S \in \mathcal{L}_{in}$ with $|S| \leq 2\lambda$, S is F_{out}-contractible.

(ii)' In the $(F_{out}, \mathcal{L}_{in}, 2\lambda)$-contraction of G, there is no F_{out}-respecting alternating circuit of node-weight at most 2λ.

Throughout this section, F_{in} and $\mathcal{L}_{\mathsf{in}}$ are as in the statement of Theorem 8.22. The proof of Theorem 8.22 is based on the following technical lemma.

Lemma 8.23

There exists a weight function $w_{mid} \in \mathcal{W}^{\log_2 n}$ such that the face $F_{mid} = F_{in}[w_{mid}]$ satisfies:

(i)' For each $S \in \mathcal{L}_{in}$ with $|S| \leq 2\lambda$, S is F_{mid}-contractible.

Before giving the proof of Lemma 8.23 let us see how it, together with Theorem 8.18, readily implies Theorem 8.22. In short, once we have made sets of size up to 2λ contractible using Lemma 8.23, we are only left with removing alternating circuits of node-weight between λ and 2λ. One application of Theorem 8.18 is enough to achieve this.

Proof of Theorem 8.22.
Lemma 8.23 says that there is a weight function $w_{\mathsf{mid}} \in \mathcal{W}^{\log_2(n)}$ such that the face $F_{\mathsf{mid}} = F_{\mathsf{in}}[w_{\mathsf{mid}}]$ satisfies that every $S \in \mathcal{L}_{\mathsf{in}}$ with $|S| \leq 2\lambda$ is F_{mid}-contractible. As we will obtain F_{out} as a subface of F_{mid}, we have thus proved point $(i)'$ of Theorem 8.22, as any set that is F_{mid}-contractible will remain contractible in any subface of F_{mid} (by Fact 8.8).

By the above, every vertex in the $(F_{\mathsf{mid}}, \mathcal{L}_{\mathsf{in}}, 2\lambda)$-contraction of G corresponds to an F_{mid}-contractible set. Moreover, by the assumption that the face-laminar pair $(F_{\mathsf{in}}, \mathcal{L}_{\mathsf{in}})$ is λ-good, the $(F_{\mathsf{in}}, \mathcal{L}_{\mathsf{in}}, \lambda)$-contraction of G does not have any alternating circuits of node-weight at most λ. This implies that the $(F_{\mathsf{mid}}, \mathcal{L}_{\mathsf{in}}, 2\lambda)$-contraction of G does not have any such alternating circuits. For suppose C were one. Let $S_1, ..., S_k$ be maximal sets of size at most 2λ in $\mathcal{L}_{\mathsf{in}}$, i.e., the vertices of the $(F_{\mathsf{mid}}, \mathcal{L}_{\mathsf{in}}, 2\lambda)$-contraction of G. Note that C cannot cross a set S_i with $|S_i| > \lambda$, because then its node-weight would be larger than λ. Therefore C only crosses sets S_i with $|S_i| \leq \lambda$. Thus C also appears in the $(F_{\mathsf{mid}}, \mathcal{L}_{\mathsf{in}}, \lambda)$-contraction of G, with the same node-weight, and in the $(F_{\mathsf{in}}, \mathcal{L}_{\mathsf{in}}, \lambda)$-contraction (of which the $(F_{\mathsf{mid}}, \mathcal{L}_{\mathsf{in}}, \lambda)$-contraction is a subgraph) as well. This is a contradiction.

We can thus apply Theorem 8.18 with $\beta = 2\lambda$ to the face-laminar pair $(F_{\mathsf{mid}}, \mathcal{L}_{\mathsf{in}})$. We get a weight function $w \in \mathcal{W}$ such that the $(F_{\mathsf{mid}}, \mathcal{L}_{\mathsf{in}}, 2\lambda)$-contraction has no $F_{\mathsf{mid}}[w]$-respecting alternating circuit of node-weight at most 2λ. Therefore, as the $(F_{\mathsf{mid}}[w], \mathcal{L}_{\mathsf{in}}, 2\lambda)$-contraction is a subgraph of the $(F_{\mathsf{mid}}, \mathcal{L}_{\mathsf{in}}, 2\lambda)$-contraction, the face $F_{\mathsf{out}} = F_{\mathsf{mid}}[w]$ satisfies $(ii)'$. Selecting $w_{\mathsf{out}} = w_{\mathsf{mid}} \circ w \in \mathcal{W}^{\log_2(n)+1}$ completes the proof (by Fact 7.19). □

The rest of this section is devoted to the proof of Lemma 8.23. Recall that we need to prove the existence of a weight function $w_{\mathsf{mid}} \in \mathcal{W}^{\log_2(n)}$ satisfying $(i)'$, i.e., that

$$\text{for each } S \in \mathcal{L}_{\mathsf{in}} \text{ with } |S| \leq 2\lambda,\ S \text{ is } F_{\mathsf{mid}}\text{-contractible},$$

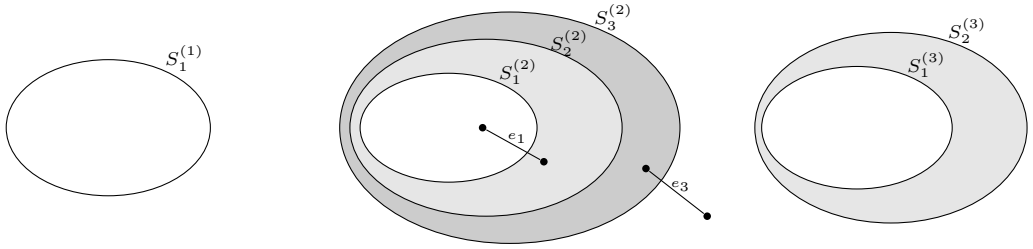


Figure 8.9: An example of the laminar family $\mathcal{L}$, which consists of disjoint chains. The different shades of gray depict the sets $U_p^{(i)}$.

where $F_{\mathsf{mid}} = F_{\mathsf{in}}[w_{\mathsf{mid}}]$. First note that the statement will be true for every $S \in \mathcal{L}_{\mathsf{in}}$ with $|S| \leq \lambda$, regardless of the choice of the weight function w_{mid}. Indeed, by assumption $(F_{\mathsf{in}}, \mathcal{L}_{\mathsf{in}})$ is λ-good and so S is F_{in}-contractible. Thus, by Fact 8.8, S remains F_{mid}-contractible for any subface $F_{\mathsf{mid}} \subseteq F_{\mathsf{in}}$.

It remains to deal with the sets $S \in \mathcal{L}_{\mathsf{in}}$ with $\lambda < |S| \leq 2\lambda$. Let $\mathcal{L} = \{S \in \mathcal{L}_{\mathsf{in}} : \lambda < |S| \leq 2\lambda\}$ be the laminar family $\mathcal{L}_{\mathsf{in}}$ restricted to these sets. Notice that any set in $\mathcal{L}$ can have at most one child in $\mathcal{L}$ due to the cardinality constraints. In other words, $\mathcal{L}$ consists of disjoint chains, as depicted in Figure 8.9.

Notation. We refer to the sets in the i-th chain of $\mathcal{L}$ by $\mathcal{L}_i$. Let $\ell_i = |\mathcal{L}_i|$ and index the sets of the chain $\mathcal{L}_i = \{S_1^{(i)}, S_2^{(i)}, \ldots, S_{\ell_i}^{(i)}\}$ so that $S_1^{(i)} \subseteq S_2^{(i)} \subseteq \cdots \subseteq S_{\ell_i}^{(i)}$. Let $U_1^{(i)} = S_1^{(i)}$ and $U_p^{(i)} = S_p^{(i)} \setminus S_{p-1}^{(i)}$ for $p = 2, 3, \ldots, \ell_i$. Also define $U_{p,r}^{(i)} = U_p^{(i)} \cup U_{p+1}^{(i)} \cup \cdots \cup U_r^{(i)}$.

Recall that a set $U_{1,r}^{(i)} = S_r^{(i)}$ is defined to be F-contractible if for every $e_r \in \delta(U_{1,r}^{(i)})$ there are no two perfect matchings in F which both contain e_r and are different inside $U_{1,r}^{(i)}$ (Definition 8.7). For the proof of Lemma 8.23, we generalize this definition to also include sets $U_{p,r}^{(i)}$ with $p \geq 2$.

Definition 8.24

Consider a face F. We say that a set $U_{p,r}^{(i)}$ with $2 \leq p \leq r \leq \ell_i$ is F-contractible *if for every $e_{p-1} \in \delta(S_{p-1}^{(i)})$ and $e_r \in \delta(S_r^{(i)})$ there are no two perfect matchings in F which both contain e_{p-1} and e_r and are different inside $U_{p,r}^{(i)}$. (It is possible that $e_{p-1} = e_r$, in which case neither endpoint of this edge lies in $U_{p,r}^{(i)}$.)*

The intuition of this definition is similar to that of Definition 8.7. Consider the second chain in Figure 8.9 for an example. If we restrict our attention to perfect matchings that contain edges $e_1 \in \delta(S_1^{(2)})$ and $e_3 \in \delta(S_3^{(2)})$, then, as $S_1^{(2)}$ are $S_3^{(2)}$ are tight sets, the task of selecting such a matching decomposes into two *independent* problems: the problem of selecting a perfect matching in $U_{2,3}^{(2)}$ (ignoring the vertices incident to e_1 and e_3) and the problem of selecting a perfect matching in $V \setminus U_{2,3}^{(2)}$ (again ignoring the vertices incident to e_1 and e_3).

The proof now proceeds iteratively as follows.

- First we select $w_1 \in \mathcal{W}$ such that $F_1 = F_{\mathsf{in}}[w_1]$ satisfies:

$$U_p^{(i)} \text{ is } F_1\text{-contractible for all chains } i \text{ and } 1 \leq p \leq \ell_i. \tag{8.1}$$

- For $t = 2, 3, \ldots, \log_2(n)$ we select $w_t \in \mathcal{W}$ such that $F_t = F_{t-1}[w_t]$ satisfies:

$$U^{(i)}_{p,r} \text{ is } F_t\text{-contractible for all chains } i \text{ and } 1 \leq p \leq r \leq \ell_i \text{ with } r - p \leq 2^{t-1} - 1. \quad (8.2)$$

We remark that, having selected $w_1, w_2, \ldots, w_{\log_2(n)}$ as above, if we let $w_{\mathsf{mid}} = w_1 \circ w_2 \circ \ldots \circ w_{\log_2(n)} \in \mathcal{W}^{\log_2(n)}$, then the face $F_{\mathsf{mid}} = F_{\mathsf{in}}[w_{\mathsf{mid}}]$ equals $F_{\log_2(n)}$ (by Fact 7.19). To see that this completes the proof of Lemma 8.23, note that $\ell_i < n/2$ for any chain i since $|S^{(i)}_1| > \lambda = (2\lambda)/2 \geq |S^{(i)}_{\ell_i}|/2$ and $|S^{(i)}_{\ell_i}| \leq n$. Therefore any set $S^{(i)}_r \in \mathcal{L}$ has $r \leq n/2$ and so, by (8.2), $S^{(i)}_r = U^{(i)}_{1,r}$ is F_{mid}-contractible.

In what follows, we complete the proof of Lemma 8.23 with a description of how to select w_1, followed by the selection of w_t, $t = 2, 3, \ldots, \log_2(n)$, in the iterative case.

The selection of w_1

The following claim allows us to use Theorem 8.18 to show the existence of a weight function w_1 satisfying (8.1).

Claim 13

If the $(F_{in}, \mathcal{L}_{in}, \lambda)$-contraction of G has no F_1-respecting alternating circuit of node-weight at most 2λ, then every $U^{(i)}_p$ is F_1-contractible.

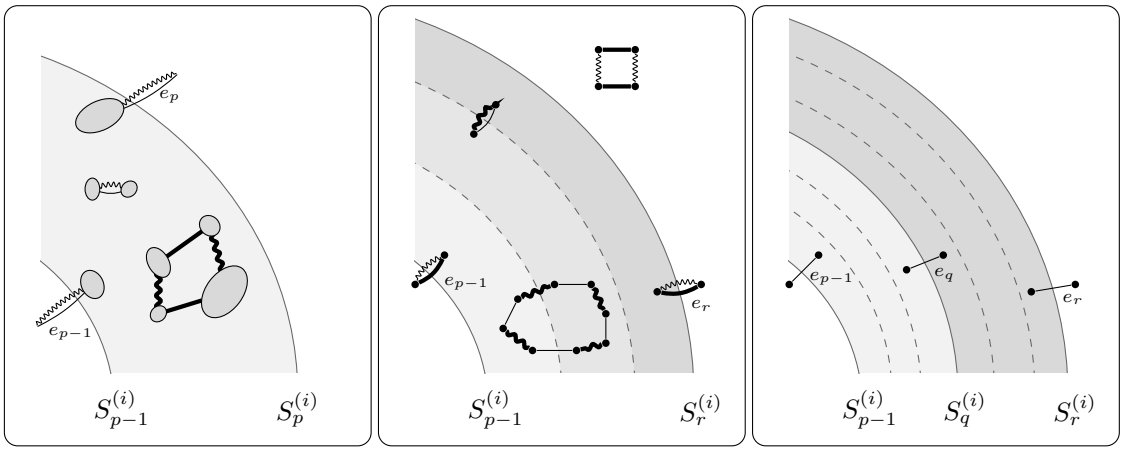
The claim together with Theorem 8.18 completes the selection of w_1 as follows. Since $(F_{\mathsf{in}}, \mathcal{L}_{\mathsf{in}})$ is λ-good, we can apply Theorem 8.18 with $\beta = \lambda$ to obtain the existence of a weight function $w_1 \in \mathcal{W}$ such that in the $(F_{\mathsf{in}}, \mathcal{L}_{\mathsf{in}}, \lambda)$-contraction of G there is no F_1-respecting alternating circuit of node-weight at most 2λ, where $F_1 = F_{\mathsf{in}}[w_1]$. Hence, by the above claim, every $U^{(i)}_p$ is F_1-contractible as required.

Proof of Claim 13.
This proof resembles that of Lemma 8.16. Fix $U^{(i)}_p$ and let $e_{p-1} \in \delta(S^{(i)}_{p-1})$ and $e_p \in \delta(S^{(i)}_p)$.[22] Suppose that M_1 and M_2 are two perfect matchings in F_1 that both contain e_{p-1} and e_p. We want to show that M_1 and M_2 are equal inside $U^{(i)}_p$.

Let $S_1, ..., S_k$ be all maximal sets $S \in \mathcal{L}_{\mathsf{in}}$ with $S \subseteq U^{(i)}_p$. They are those vertices of the $(F_{\mathsf{in}}, \mathcal{L}_{\mathsf{in}}, \lambda)$-contraction which lie in $U^{(i)}_p$, and we have $S_1 \cup ... \cup S_k = U^{(i)}_p$. Because these sets, as well as $S^{(i)}_{p-1}$ and $S^{(i)}_p$, are tight for F_1, any perfect matching in F_1 containing e_{p-1} and e_p induces an almost-perfect matching on $S_1, ..., S_k$, that is, one where only the (up to two) sets S_i containing endpoints of e_{p-1} and e_p are unmatched (see the left part of Figure 8.10).

If the matchings induced by M_1 and M_2 were different, then their symmetric difference would induce an alternating simple cycle C in the graph obtained from $E(F_{\mathsf{in}})$ by contracting $S_1, ..., S_k$. The cycle C would also be present in the $(F_{\mathsf{in}}, \mathcal{L}_{\mathsf{in}}, \lambda)$-contraction. This is because $S_1, ..., S_k$ are maximal sets of size at most λ in $\mathcal{L}_{\mathsf{in}}$, so they are indeed vertices of the contraction, and because the edges between these vertices are not on the boundary $\delta(T)$ of any $T \in \mathcal{L}_{\mathsf{in}}$ with $|T| > \lambda$ (in which case they would be missing from the contraction), which in turn follows by laminarity of $\mathcal{L}_{\mathsf{in}}$ and the definition of $U^{(i)}_p$.

[22] Here and in Section 8.4.2, we abuse notation and assume $p \geq 2$. The only difference is that, given a set $U^{(i)}_{p,r}$ with $p = 1$ (in this section $p = r$), we consider matchings containing one edge $e_r \in \delta(S^{(i)}_r)$ instead of matchings containing two edges $e_{p-1} \in \delta(S^{(i)}_{p-1})$ and $e_r \in \delta(S^{(i)}_r)$, since if $p = 1$, the set $S^{(i)}_{p-1}$ is not defined.



Claim 13 Claim 14 Claim 15

Figure 8.10: An illustration of the claims used in the proof of Lemma 8.23.
Claim 13: Straight and swirly edges denote M_1 and M_2 respectively. The thick edges denote the alternating cycle. The dark-gray sets are $S_1, ..., S_k$.
Claim 14: Straight and swirly edges denote M_1 and M_2 respectively. The thick edges denote M_{12}, which agrees with M_1 outside $U_{p,r}^{(i)}$ and with M_2 inside $U_{p,r}^{(i)}$.
Claim 15: The divide-and-conquer argument is illustrated (only edges e_{p-1}, e_q, and e_r are depicted). After fixing e_{p-1} and e_q, the matching in the light-gray area is unique in the face F_{t-1}. Similarly, after fixing e_q and e_r, the matching in the dark-gray area is unique in the face F_{t-1}. Thus, for each choice of e_{p-1} and e_r, there can be at most one matching inside $U_{p,r}^{(i)}$ for each possible way of fixing e_q. It follows that there are at most n^2 matchings inside $U_{p,r}^{(i)}$ that contain e_{p-1} and e_r.

Since C arises from two matchings in F_1, it respects F_1. This follows by a similar argument as the proof of Claim 12, which we repeat here for completeness. Clearly, $\text{supp}(C) \subseteq M_1 \cup M_2 \subseteq E(F_1)$. Let $T \in \mathcal{S}(F_1)$ be a set tight for F_1 which is a union of the vertices of the contraction; we want to show that $\left\langle (\pm\mathbb{1})_C, \mathbb{1}_{\delta(T)} \right\rangle = 0$. Because C is a cycle in the contraction and $|M_1 \cap \delta(T)| = |M_2 \cap \delta(T)| = 1$, either C has no edge in $\delta(T)$ or it has two, one from M_1 and one from M_2 (and they cancel out).

Moreover, since C is a simple cycle inside $U_p^{(i)}$, its node-weight is at most $|S_1| + \ldots + |S_k| = |U_p^{(i)}| \leq 2\lambda$. This contradicts our assumption.

Therefore, the induced matchings must be equal. Moreover, the sets $S_1, ..., S_k$ are F_1-contractible, since they are vertices of the $(F_{\text{in}}, \mathcal{L}_{\text{in}}, \lambda)$-contraction, $(F_{\text{in}}, \mathcal{L}_{\text{in}})$ is λ-good, and $F_1 \subseteq F_{\text{in}}$. This means that, given the boundary edges (i.e., the induced matching plus e_{p-1} and e_p), there is a unique perfect matching in F_1 inside each S_i. It follows that M_1 and M_2 are equal inside $U_p^{(i)}$. □

The selection of w_t for $t = 2, 3, \ldots, \log_2(n)$

In this section we show the existence of a weight function $w_t \in \mathcal{W}$ satisfying (8.2), i.e.,

$$U^{(i)}_{p,r} \text{ is } F_t\text{-contractible for all chains } i \text{ and } 1 \le p \le r \le \ell_i \text{ with } r - p \le 2^{t-1} - 1,$$

where $F_t = F_{t-1}[w_t]$.

The proof outline is as follows. First, in Claim 14, we give sufficient conditions on w_t for $U^{(i)}_{p,r}$ to be F_t-contractible. They are given as a system of linear non-equalities with coefficients in $\{-1, 0, 1\}$. Then, in Claim 15, we upper-bound the number of these non-equalities by n^{11}. This allows us to deduce the existence of $w_t \in \mathcal{W}$ by applying Lemma 8.5.

The following claim gives sufficient linear non-equalities on w_t for every $U^{(i)}_{p,r}$ to be F_t-contractible (one non-equality for each choice of $U^{(i)}_{p,r}$, e_{p-1}, e_r, M_1 and M_2).

Claim 14

Let $U = U^{(i)}_{p,r}$ for some chain i and $1 \le p \le r \le \ell_i$. Suppose that for every two edges $e_{p-1} \in \delta(S^{(i)}_{p-1})$ and $e_r \in \delta(S^{(i)}_r)$ defining a face $F = \{x \in F_{t-1} : x_{e_{p-1}} = 1, x_{e_r} = 1\}$ we have:

$$w_t(M_1 \cap E(U)) \neq w_t(M_2 \cap E(U)) \quad \text{for any two matchings } M_1, M_2 \text{ in } F \text{ that differ inside } U.$$

Then U is F_t-contractible.

Proof.
We prove the contrapositive. Suppose that U is not F_t-contractible. Then, by definition, there must be $e_{p-1} \in \delta(S^{(i)}_{p-1})$ and $e_r \in \delta(S^{(i)}_r)$ that define a face $F' = \{x \in F_t : x_{e_{p-1}} = 1, x_{e_r} = 1\}$ such that there are two matchings M_1 and M_2 in F' that differ inside U. Notice that $F' \subseteq F = \{x \in F_{t-1} : x_{e_{p-1}} = 1, x_{e_r} = 1\}$. Therefore M_1 and M_2 are also two matchings in F that differ inside U.

We complete the proof of the claim by showing that

$$w_t(M_1 \cap E(U)) = w_t(M_2 \cap E(U)) \,. \tag{8.3}$$

Define

$$M_{12} = (M_1 \setminus E(U)) \cup (M_2 \cap E(U))$$

to be the perfect matching that agrees with M_1 on all edges not in $E(U)$ and agrees with M_2 on all edges in $E(U)$ (see the central part of Figure 8.10 for an example). By the same argument as in the proof of Lemma 8.9, M_{12} is a perfect matching in F'. It differs from M_1 inside U and agrees with M_1 outside U.

We now use that M_1 and M_{12} are perfect matchings in F' to prove (8.3). As $F_t = F_{t-1}[w_t]$ is the convex-hull of matchings in F_{t-1} that minimize the objective function w_t, all matchings M in F_t and in its subface F' have the same weight $w_t(M)$. In particular,

$$w_t(M_1 \setminus E(U)) + w_t(M_1 \cap E(U)) = w_t(M_1) = w_t(M_{12}) = w_t(M_1 \setminus E(U)) + w_t(M_2 \cap E(U))$$

and thus $w_t(M_1 \cap E(U)) = w_t(M_2 \cap E(U))$ as required. □

The above claim says that it is sufficient to write down a non-equality for each choice of $U^{(i)}_{p,r}$, e_{p-1}, e_r, M_1, and M_2. It is easy to upper-bound the number of ways of choosing i, p, r, e_{p-1}, and e_r. The following claim bounds the number of ways of choosing M_1 and M_2. Its proof is based on a divide-and-conquer strategy (see the right part of Figure 8.10). It uses the inductive assumption that $U^{(i)}_{p,r}$ is F_{t-1}-contractible for all chains i and $1 \le p \le r \le \ell_i$ with $r-p \le 2^{t-2}-1$.

Claim 15

Let $U = U^{(i)}_{p,r}$ with $r-p \le 2^{t-1}-1$ and define $q = \lfloor (p+r)/2 \rfloor$. For any two edges $e_p \in \delta(S^{(i)}_{p-1})$ and $e_r \in \delta(S^{(i)}_r)$ defining a face $F = \{x \in F_{t-1} : x_{e_{p-1}} = 1, x_{e_r} = 1\}$ we have

$$|\{M \cap E(U) : M \text{ is a matching in } F\}| \le |\delta(S^{(i)}_q) \cap E(F)| \le n^2 \,.$$

We remark that the first inequality holds with equality, but we only need the inequality.

Proof.
The second inequality in the statement is trivial. We prove the first.

As we have $S^{(i)}_q \in \mathcal{L}_{\mathsf{in}} \subseteq \mathcal{S}(F_{\mathsf{in}})$ and $F \subseteq F_{t-1} \subseteq F_{\mathsf{in}}$, the set $S^{(i)}_q$ is tight for F. Thus any matching M in F must satisfy $M \cap \delta(S^{(i)}_q) = \{e_q\}$ for some edge $e_q \in \delta(S^{(i)}_q) \cap E(F)$.

We prove the statement by showing that for every choice of e_q, any matching M in the face $F_{e_q} = \{x \in F : x_{e_q} = 1\}$ matches the nodes in $U^{(i)}_{p,r}$ in a unique way. In other words, we show that $|\{M \cap E(U) : M \text{ is a matching in } F_{e_q}\}| \le 1$ for every $e_q \in \delta(S^{(i)}_q) \cap E(F)$, which implies

$$\begin{aligned} |\{M \cap E(U) : M \text{ is a matching in } F\}| &\le \sum_{e_q \in \delta(S^{(i)}_q) \cap E(F)} |\{M \cap E(U) : M \text{ is a matching in } F_{e_q}\}| \\ &\le |\delta(S^{(i)}_q) \cap E(F)| \,. \end{aligned}$$

To prove that $|\{M \cap E(U) : M \text{ is a matching in } F_{e_q}\}| \le 1$, suppose the contrary, i.e., that $|\{M \cap E(U) : M \text{ is a matching in } F_{e_q}\}| \ge 2$. Take two such matchings M_1 and M_2 that differ inside U. By the definition of F_{e_q} we have $M_1 \cap \delta(S^{(i)}_q) = M_2 \cap \delta(S^{(i)}_q) = \{e_q\}$ and so M_1 and M_2 must differ inside $U^{(i)}_{p,q}$ or inside $U^{(i)}_{q+1,r}$; assume the former (the argument for the other case is the same). Notice that M_1 and M_2 are two matchings in $F_{e_q} \subseteq F_{t-1}$ which both contain e_{p-1} and e_q but differ inside $U^{(i)}_{p,q}$, which contradicts that $U^{(i)}_{p,q}$ is F_{t-1}-contractible. (Note that $q - p \le (r-p)/2 \le 2^{t-2} - 1/2$, which implies that $q - p \le 2^{t-2} - 1$.) □

We now have all the needed tools to show the existence of a weight function $w_t \in \mathcal{W}$ such that the face $F_t = F_{t-1}[w_t]$ satisfies (8.2), i.e., that for all chains i and $1 \le p \le r \le \ell_i$ with $r - p \le 2^{t-1} - 1$, $U^{(i)}_{p,r}$ is F_t-contractible. By Claim 14, this holds if for any $U = U^{(i)}_{p,r}$ with $r-p \le 2^{t-1}-1$ and for any $e_{p-1} \in \delta(S^{(i)}_{p-1})$ and $e_r \in \delta(S^{(i)}_r)$ defining a face $F = \{x \in F_{t-1} : x_{e_{p-1}} = 1, x_{e_r} = 1\}$ we have the following:

$$w_t(M_1 \cap E(U)) - w_t(M_2 \cap E(U)) \neq 0 \quad \text{for any two matchings } M_1, M_2 \text{ in } F \text{ that differ inside } U.$$

There are at most n ways of choosing i, n ways of choosing p, n ways of choosing r, n^2 ways of choosing e_{p-1}, n^2 ways of choosing e_r, and by Claim 15 there are at most n^4 ways of choosing M_1 and M_2. In total, we can write the sufficient conditions on the weight function

w_t as a system of at most n^{11} linear non-equalities with coefficients in $\{-1, 0, 1\}$. It follows by Lemma 8.5 that there is a weight function $w_t \in \mathcal{W}(n^{14}) \subseteq \mathcal{W}(n^{20}) = \mathcal{W}$ satisfying these conditions. This completes the selection of w_t and the proof of Lemma 8.23.

8.4.3 A maximal laminar family completes the proof

In Theorem 8.22 we have demonstrated the existence of a weight function w_{out} that defines a face F_{out} with properties $(i)'$ and $(ii)'$. We now show that extending $\mathcal{L}_{\text{in}}$ to a maximal laminar family $\mathcal{L}_{\text{out}}$ of $\mathcal{S}(F_{\text{out}})$ yields a 2λ-good face-laminar pair. Such an extension is possible because $\mathcal{L}_{\text{in}}$ consists of sets that are tight for F_{in} and thus also for F_{out}. As explained in the beginning of Section 8.4, this will complete the proof of Theorem 8.15.

Why a *maximal* laminar family? Part of our argument so far was about removing certain alternating circuits C. In other words, we have made C not respect the new face F_{out}. This means either not having some edge from $\text{supp}(C)$ in the support $E(F_{\text{out}})$ of F_{out}, or introducing a new odd-set S which is tight for F_{out} and such that $\left\langle (\pm\mathbb{1})_C, \mathbb{1}_{\delta(S)} \right\rangle \neq 0$. In the latter case, we want to have an odd-set with this property also in the new laminar family $\mathcal{L}_{\text{out}}$, so that the removal of C is reflected in the new contraction (which is based on $\mathcal{L}_{\text{out}}$). Lemma 8.19 guarantees that this will happen if we choose $\mathcal{L}_{\text{out}}$ to be a maximal laminar subset of $\mathcal{S}(F_{\text{out}})$.

Lemma 8.25

Let $(F_{in}, \mathcal{L}_{in})$ be a λ-good face-laminar pair and $F_{out} \subseteq F_{in}$ be the face guaranteed by Theorem 8.22. Then $(F_{out}, \mathcal{L}_{out})$ is a 2λ-good face-laminar pair, where $\mathcal{L}_{out}$ is any maximal laminar family with $\mathcal{L}_{in} \subseteq \mathcal{L}_{out} \subseteq \mathcal{S}(F_{out})$.

Proof of Lemma 8.25.

Recall that Theorem 8.22 guarantees that:

(i)' each $S \in \mathcal{L}_{\text{in}}$ with $|S| \leq 2\lambda$ is F_{out}-contractible,

(ii)' in the $(F_{\text{out}}, \mathcal{L}_{\text{in}}, 2\lambda)$-contraction of G, there is no alternating circuit of node-weight at most 2λ which respects F_{out}.

We want to show that the pair $(F_{\text{out}}, \mathcal{L}_{\text{out}})$ satisfies Definition 8.13, that is,

(i) each $S \in \mathcal{L}_{\text{out}}$ with $|S| \leq 2\lambda$ is F_{out}-contractible,

(ii) in the $(F_{\text{out}}, \mathcal{L}_{\text{out}}, 2\lambda)$-contraction of G, there is no alternating circuit of node-weight at most 2λ.

Property (i). Fix a set $S \in \mathcal{L}_{\text{out}}$ with $|S| \leq 2\lambda$. Let $S_1, ..., S_k$ be all maximal subsets of S in $\mathcal{L}_{\text{in}}$ (we have $S = S_1 \cup ... \cup S_k$). If S is contained in a set from $\mathcal{L}_{\text{in}}$ of size at most 2λ, then that set is F_{out}-contractible by $(i)'$, and thus S is F_{out}-contractible by Lemma 8.9. So assume that is not the case; therefore, by laminarity, each S_i is a maximal set of size at most 2λ in $\mathcal{L}_{\text{in}}$, that is, a vertex of the $(F_{\text{out}}, \mathcal{L}_{\text{in}}, 2\lambda)$-contraction of G. By $(i)'$, each S_i is F_{out}-contractible.

Now the proof proceeds as in Claim 13. We present it for completeness. Let M_1 and M_2 be two perfect matchings in F_{out} which both contain an edge $e \in \delta(S)$. We want to show that M_1 and M_2 are equal inside S. Because S and $S_1, ..., S_k$ are tight for F_{out}, any perfect matching (on

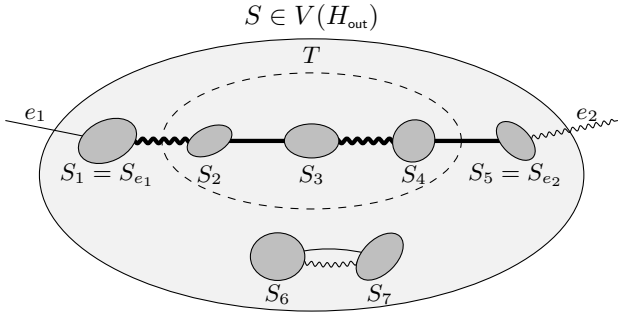


Figure 8.11: The construction of the path $P_{e_1e_2}$ inside a set $S \in V(H_{\text{out}})$ in the proof of Property (ii). The dark-gray sets correspond to vertices of H_{in} that are subsets of S. The straight and swirly edges depict matchings M_1 and M_2, respectively. The path $P_{e_1e_2}$ is depicted by fat edges.

G) in F_{out} containing e induces an almost-perfect matching on $S_1, ..., S_k$, that is, one where only the set S_i containing the S-endpoint of e is unmatched.

If the matchings induced by M_1 and M_2 were different, then their symmetric difference would contain an alternating simple cycle in the $(F_{\text{out}}, \mathcal{L}_{\text{in}}, 2\lambda)$-contraction. Since this cycle arises from two matchings in F_{out}, it respects F_{out}. Moreover, since it is a simple cycle inside S, its node-weight is at most $|S_1| + \ldots + |S_k| = |S| \leq 2\lambda$. This would contradict our assumption (ii)'.

Therefore the induced matchings must be equal. Moreover, the sets $S_1, ..., S_k$ are F_{out}-contractible, which means that, given the boundary edges (i.e., the induced matching plus e), there is a unique perfect matching in F_{out} inside each S_i. It follows that M_1 and M_2 are equal inside S.

Property (ii). Let H_{out} be the $(F_{\text{out}}, \mathcal{L}_{\text{out}}, 2\lambda)$-contraction of G and let H_{in} be the $(F_{\text{out}}, \mathcal{L}_{\text{in}}, 2\lambda)$-contraction of G. Thus H_{out} can also be obtained by further contracting H_{in}, as well as removing all edges that are in the boundaries of sets $S \in \mathcal{L}_{\text{out}} \setminus \mathcal{L}_{\text{in}}$ with $|S| > 2\lambda$. This will be our perspective. Suppose towards a contradiction that there is an alternating circuit C_{out} in H_{out} of node-weight at most 2λ.

To obtain a contradiction, we are going to lift C_{out} back to an F_{out}-respecting alternating circuit C_{in} in H_{in}, which should not exist by $(ii)'$. (This is in the same spirit as the proof of Lemma 8.20.) Namely, whenever C_{out} visits a vertex $S \in V(H_{\text{out}})$, we connect up the dangling endpoints of this visit inside S to obtain a walk in H_{in}. More precisely, let e_1 and e_2 be two consecutive edges of C_{out}, whose common endpoint in H_{out} is S. Between them, we insert a simple path $P_{e_1e_2}$ inside the image of S in H_{in}, which is constructed as follows.

Since $e_1, e_2 \in \text{supp}(C_{\text{out}}) \subseteq E(F_{\text{out}})$, there exist matchings M_1 and M_2 (on G) in F_{out} containing e_1 and e_2, respectively. Let $S_1, ..., S_k$ be all maximal subsets of S in $\mathcal{L}_{\text{in}}$ (S_i are vertices of H_{in} and we have $S = S_1 \cup ... \cup S_k$). Denote by S_{e_1} and S_{e_2} the sets S_i which contain the S-endpoint of e_1 and e_2, respectively. The sets S and $S_1, ..., S_k$ are tight for F_{out}, so M_1 induces a perfect matching on $\{S_1, ..., S_k\} \setminus \{S_{e_1}\}$ (and similarly for M_2 and e_2). The symmetric difference of these two induced matchings contains a simple path $P_{e_1e_2}$ from S_{e_1} to S_{e_2} in H_{in} which has even length (possibly 0). For an example, see Figure 8.11. We obtain C_{in} by inserting such a path $P_{e_1e_2}$ between each two consecutive edges e_1, e_2 in C_{out}.

To obtain a contradiction, we need to prove that C_{in} is an alternating circuit of node-weight at most 2λ which respects F_{out}.

- That C_{in} is an alternating circuit follows by construction because each path $P_{e_1e_2}$ is of even length and the alternating indicator vector of C_{in} is nonzero since it contains the alternating indicator vector of C_{out} (via the natural mapping of edges).
- For the node-weight, note that in C_{out}, the visit to S (on the edge e_1) incurs a node-weight increase of $|S|$, whereas in C_{in}, the visit to a certain subset of $\{S_1, ..., S_k\}$ (on e_1 and $P_{e_1e_2}$) incurs an increase of at most $|S_1|+\ldots+|S_k| = |S|$ because $P_{e_1e_2}$ is a simple path. Therefore the node-weight of C_{in} is at most that of C_{out} – at most 2λ.
- To see that C_{in} respects F_{out}, we use the assumption that $\mathcal{L}_{\text{out}}$ is a maximal laminar subset of $\mathcal{S}(F_{\text{out}})$. This is in similar spirit as the proof of Claim 12. To work around the fact that we are dealing with the contraction, we use the following version of Lemma 8.19:

  Claim 16

  > *Consider a vector $z \in \mathbb{Z}^{E(H_{in})}$. If for each $T \in \mathcal{L}_{out}$ which is a union of sets in $V(H_{in})$ we have $\langle z, \mathbb{1}_{\delta(T)}\rangle = 0$, then the same holds for each $T \in \mathcal{S}(F_{out})$ which is a union of sets in $V(H_{in})$.*

  Let us defer the proof of Claim 16 and first use it to show that C_{in} respects F_{out}. We verify the conditions of Definition 8.12. First, note that $\text{supp}(C_{\text{in}}) \subseteq E(F_{\text{out}})$ by construction. Second, let $T \in \mathcal{S}(F_{\text{out}})$ be a union of vertices of H_{in}; we need to show that $\langle (\pm\mathbb{1})_{C_{\text{in}}}, \mathbb{1}_{\delta(T)}\rangle = 0$. By Claim 16, we may assume that $T \in \mathcal{L}_{\text{out}}$. We consider two cases:

  - If $|T| > 2\lambda$, then all boundary edges of T are absent from H_{out} (see Definition 8.11), so $\text{supp}(C_{\text{out}}) \cap \delta(T) = \emptyset$. In this case T is a union of vertices of H_{out} and so no path $P_{e_1e_2}$ contains any edges from $\delta(T)$ either. Hence $\text{supp}(C_{\text{in}}) \cap \delta(T) = \emptyset$.[23]
  - If $|T| \leq 2\lambda$, then T must be contained in a single set $S \in V(H_{\text{out}})$ (as depicted in Figure 8.11). This is because $T \in \mathcal{L}_{\text{out}}$ and the sets $S \in V(H_{\text{out}})$ are maximal sets $S \in \mathcal{L}_{\text{out}}$ with $|S| \leq 2\lambda$. For every path $P_{e_1e_2}$ inside S, the path $e_1, P_{e_1e_2}, e_2$ is a path from outside of S to outside of S which is part of the symmetric difference of two matchings in F_{out}. If this path enters T, it must also leave T. Suppose it entered T on an edge of the first matching. Then it must exit T on an edge of the second matching, since $T \in \mathcal{L}_{\text{out}} \subseteq \mathcal{S}(F_{\text{out}})$ is tight for F_{out} and both matchings are in F_{out}, and so the corresponding ± 1 terms cancel out. Abusing notation, we have $\langle (\pm\mathbb{1})_{e_1, P_{e_1e_2}, e_2}, \mathbb{1}_{\delta(T)}\rangle = 0$. Since this holds for every path $P_{e_1e_2}$ inside S, we get $\langle (\pm\mathbb{1})_{C_{\text{in}}}, \mathbb{1}_{\delta(T)}\rangle = 0$ as required.

Thus C_{in} is an alternating circuit of node-weight at most 2λ which respects F_{out}. Its existence contradicts $(ii)'$. We conclude with the proof of Claim 16:

Proof of Claim 16.
We wish to reduce our setting to that of Lemma 8.19. Define H'_{in} to be the graph obtained from $(V, E(F_{\text{out}}))$ by contracting all maximal sets $S \in \mathcal{L}_{\text{in}}$ with $|S| \leq 2\lambda$, but not erasing the

[23] Here we take advantage of the fact that the boundaries of large sets are erased in the definition of the contraction. If they were not erased, we would be unable to proceed, as there would be no reason for C_{out} (and thus C_{in}) to respect F_{out}.

boundaries of sets $S \in \mathcal{L}_{\mathsf{in}}$ with $|S| > 2\lambda$. (If we did erase them, we would obtain H_{in}; instead, we have $V(H_{\mathsf{in}}) = V(H'_{\mathsf{in}})$ and $E(H_{\mathsf{in}}) \subseteq E(H'_{\mathsf{in}})$.)

Next, we define F'_{out} to be the image of F_{out} in H'_{in}. More precisely, since the contracted sets $S \in \mathcal{L}_{\mathsf{in}}$ with $|S| \leq 2\lambda$ are tight for F_{out}, each perfect matching on G in F_{out} induces a perfect matching on H'_{in}. We let F'_{out} be the convex hull of the indicator vectors of these induced matchings. Note that there is a one-to-one correspondence between subsets of $V(H_{\mathsf{in}})$ that are tight for F'_{out} and subsets of V that are tight for F_{out} and are unions of sets in $V(H_{\mathsf{in}})$.

Finally, $\mathcal{L}_{\mathsf{out}}$ also naturally maps to a laminar family $\mathcal{L}'_{\mathsf{out}}$ of subsets of vertices of H'_{in} since $\mathcal{L}_{\mathsf{in}} \subseteq \mathcal{L}_{\mathsf{out}}$. Specifically, there is a set in $\mathcal{L}'_{\mathsf{out}}$ corresponding to each set in $\mathcal{L}_{\mathsf{out}}$ that is a union of sets in $V(H'_{\mathsf{in}})$. Note that each set in $\mathcal{L}'_{\mathsf{out}}$ is still tight for F'_{out}, and that $\mathcal{L}'_{\mathsf{out}}$ is still a maximal subset of $\mathcal{S}(F'_{\mathsf{out}})$. Indeed, if it were possible to add any set in $\mathcal{S}(F'_{\mathsf{out}})$ to $\mathcal{L}'_{\mathsf{out}}$ while maintaining laminarity, then that set could be mapped back to a set in $\mathcal{S}(F_{\mathsf{out}})$ and used to enlarge $\mathcal{L}_{\mathsf{out}}$.

Now, by assumption, for each $T \in \mathcal{L}_{\mathsf{out}}$ which is a union of sets in $V(H_{\mathsf{in}})$ we have $\langle z, \mathbb{1}_{\delta(T)} \rangle = 0$. This is equivalent to saying that $\langle z, \mathbb{1}_{\delta(T')} \rangle = 0$ for each $T' \in \mathcal{L}'_{\mathsf{out}}$. By Lemma 8.19 (applied to H'_{in}, F'_{out}, and $\mathcal{L}'_{\mathsf{out}}$), we have the same for all $T' \in \mathcal{S}(F'_{\mathsf{out}})$. Finally, that is equivalent to having $\langle z, \mathbb{1}_{\delta(T)} \rangle = 0$ for each $T \in \mathcal{S}(F_{\mathsf{out}})$ which is a union of sets in $V(H_{\mathsf{in}})$. □

This concludes the proof of Property (ii) and Lemma 8.25. □

Chapter 9

Conclusion

In this thesis we have developed new algorithms for two fundamental graph problems: the traveling salesman problem and perfect matching. In Part I we gave a constant-factor approximation algorithm for the Asymmetric TSP, and in Part II we gave a deterministic algorithm that solves perfect matching in polylogarithmic time while using a quasi-polynomial number of processors. Even though these problems are combinatorial in nature, our solutions in both cases crucially utilize the continuous linear programming formulations for these problems and extract from them a laminar structure of sets on which we then make progress by performing contractions. This serves as strong evidence of the power of polyhedral techniques.

We believe that, in general, an improved understanding of polytopes associated with combinatorial optimization problems can lead to further advances in the design and analysis of algorithms for these problems. Indeed, on the analysis side, for many problems the known LP relaxations outperform the integrality gap upper bounds that we know for them – not just in practice, but on all instances we have tried. This is certainly the case for ATSP and the Held-Karp relaxation, where our upper bound of 319, though constant, is very far from the lower bound of 2 [CGK06].

Open question 1

> *Is the integrality gap of the Held-Karp relaxation for ATSP upper-bounded by* 2*?*

On the algorithmic side, the gap in our understanding is even wider: 506 versus 75/74 [KLS13]. The decrease of the approximation ratio from 5500 [STV18a] to 506 was brought about by a more careful design and analysis of the algorithm, but we believe that in order to obtain another order-of-magnitude improvement, substantially new ideas will be required. We raise this as an important open problem.

Open question 2

> *Is there a* 50*-approximation algorithm for ATSP?*

We hope that the modularity of our framework for ATSP will facilitate new developments. In particular, the reduction of Theorem 3.4 should find wide applicability as it induces no loss in the approximation guarantee. Moreover, it is a significant open problem to improve the approximation ratio for the special case of node-weighted or singleton instances.

Another approach to ATSP that has been hoped to yield a constant-factor approximation algorithm is based on *thin spanning trees*. Given a vector $x \in \mathbb{R}^E$ (here: an optimum solution to

the Held-Karp relaxation), we say that a tree T is α-thin if for every $S \subseteq V$ we have $|T \cap \delta(S)| \leq \alpha \cdot x(\delta(S))$. Asadpour et al. [AGM+10] proved that given an α-thin tree, one can find a tour that is an $O(\alpha)$-approximation with respect to the Held-Karp relaxation. They also showed how to efficiently obtain trees of thinness $O\left(\frac{\log n}{\log \log n}\right)$, and Anari and Oveis Gharan [AG15] proved the existence of poly$(\log \log n)$-thin trees. Our algorithm does not imply any new result in this context, and the following question remains unanswered:

Open question 3

Does every graph have an $O(1)$-thin spanning tree? If so, how to find one efficiently?

A related problem is *Bottleneck TSP*: given a complete graph with metric edge weights (i.e., ones satisfying the triangle inequality), find a Hamiltonian cycle that minimizes the maximum edge weight (rather than the sum). Note that here every vertex must be visited exactly once. The approximability of the problem is well-understood in the symmetric case: there is a 2-approximation algorithm [Fle74, Lau81, PR84] and approximating the problem within a factor smaller than 2 can be easily seen to be NP-hard using the same argument as in the proof of Fact 2.1. For the asymmetric case, the best known approximation algorithm is due to An, Kleinberg and Shmoys [AKS10]. In fact, they too reduce the problem to finding thin spanning trees, which yields an $O\left(\frac{\log n}{\log \log n}\right)$-approximation algorithm, as well as a poly$(\log \log n)$-estimation algorithm [AG15]. No better result is known.

Open question 4

Is there an $O(1)$-approximation algorithm for the Bottleneck ATSP?

Note that an affirmative answer to Open question 3 would imply the same for Open question 4. Thus improving the state of the art for Bottleneck ATSP can be seen as an easier objective than for thin trees.

For the symmetric TSP, a major challenge is to improve upon the $3/2$-approximation algorithm of Christofides [Chr76] and Serdyukov [Ser78].

Open question 5

Is there a ρ-approximation algorithm with $\rho < 3/2$ for the symmetric TSP?

Recently there has been much progress in the case of unweighted graphs [GSS11, MS16, Muc12, SV14]. A compelling question is whether we can translate this progress to more general settings. Here, some natural classes might be node-weighted instances[24] or graphs with two edge weights.

Open question 6

Is there a ρ-approximation algorithm with $\rho < 3/2$ for the symmetric TSP on node-weighted graphs or on graphs with two edge weights?

We also remark that the symmetric Held-Karp relaxation exhibits the same laminar structure as we have used in Theorem 3.4. This gives us hope that the general setting can be in some sense reduced to the unweighted (or node-weighted?) setting.

[24] Note that in the asymmetric case, node-weighted instances are almost equivalent to unweighted ones, as one can replace every weighted node with a directed path of an appropriate length (see also [KTV18, Theorem 16]). We are not aware of such a reduction for the symmetric case.

For the perfect matching problem, we gave a deterministic algorithm that works in polylogarithmic time on a quasi-polynomial number of processors. The most immediate open problem left by our work is to go down to a polynomial number of processors.

Open question 7

> *Is the perfect matching problem in NC for bipartite graphs? In general?*

Making progress on Open question 7 will require new insights, as the framework of Fenner, Gurjar and Thierauf [FGT16] seems to inherently mandate a logarithmic number of weight functions applied in succession. On the other hand, there has been much recent progress that allows us to hope for a resolution of Open question 7. In a subsequent version of the paper they [FGT19] provide two somewhat different approaches to eliminating cycles (that still lead to a quasi-NC algorithm). Furthermore, our techniques have helped in obtaining an NC algorithm to find a perfect matching in planar graphs [AV18]. In general, Anari and Vazirani show that to get an NC algorithm for the search version of the problem, it is enough to give one for the decision version [AV19]. They also obtain a *pseudodeterministic* RNC algorithm for the search version: one that is randomized but with high probability always finds the same matching.

In fact, let us give a candidate algorithm:

Open question 8

> *Is it true that, for all n and for all (bipartite) graphs of size n, some function $w \in \mathcal{W}(n^{3500})$ is isolating?*

An affirmative answer to Open question 8 would yield an NC algorithm (see Lemma 7.10 and Corollary 7.15).

A related problem is Exact Matching [PY82]. We are given a graph, with some subset of edges colored red, and a number k. The task is to determine whether the graph has a perfect matching that uses exactly k red edges. If randomness is allowed, Exact Matching is efficiently solvable even in parallel: it is in RNC [MVV87]. However, in the deterministic setting our understanding is very poor: the problem is not known to be in P, even for bipartite graphs! This intriguing state of affairs seems to be related to the fact that we do not know any linear programming formulation that captures the problem well. Consider for example the natural linear program obtained by adding the constraint $\sum_e x_e = k$ to the perfect matching polytope PM, and a (bipartite) graph consisting of a single Hamiltonian cycle with every other edge red. This is a YES-instance only if $k \in \{0, n/2\}$, yet the LP is feasible for every $k \in \{0, 1, ..., n/2\}$.

Open question 9

> *Is Exact Matching in P for bipartite graphs? In general?*

Roughly speaking, the only efficient algorithms known for this problem follow the randomized linear-algebraic approach. The following is a much relaxed version of Open question 9:

Open question 10

> *Give an RP algorithm for Exact Matching (for bipartite graphs) that is significantly different from the linear-algebraic ones.*

A problem that is somewhat similar in nature is one we call Min-k Matching. We are given an edge-weighted graph and a number k. The objective is to find the cheapest set of k edges that extends to a perfect matching; or, in other words, a perfect matching whose cheapest k edges are as cheap as possible. We do not know a reduction in either direction between Min-k Matching and

Exact Matching, but the problems have the same status: the RNC algorithm [MVV87] extends to Min-k Matching, yet the problem is not known to be in P.

Open question 11

> *Is Min-k Matching in P (for bipartite graphs)?*

One can also associate a natural LP with Min-k Matching, the polytope being $\{(x, y) \in \mathbb{R}^E \times \mathbb{R}^E : x \in \mathrm{PM}, 0 \leq y \leq x, \sum_e y_e = k\}$.

Open question 12

> *What is the integrality gap of this LP?*

As we remarked above, our ability to derandomize the Isolation Lemma seems to depend on the structure of the associated polytope. This motivates the following general question:

Open question 13

> *Which zero-one polytopes admit such a derandomization?*

One class that comes to mind are totally unimodular polyhedra – and indeed, for that class this question has been resolved [GTV18], generalizing a similar result for linear matroid intersection [GT17], which can in turn be seen as a generalization of bipartite matching [FGT16]. Fenner, Gurjar and Thierauf conjecture that their approach works for any family of sets whose corresponding polytopes can be described using 0/1 constraints [FGT19].

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[AG15] Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric TSP. In *IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 20–39, 2015.

[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 *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pages 379–389, 2010.

[AHT07] Manindra Agrawal, Thanh Minh Hoang, and Thomas Thierauf. The polynomially bounded perfect matching problem is in NC^2. In *STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science*, pages 489–499, 2007.

[AKS10] Hyung-Chan An, Robert D. Kleinberg, and David B. Shmoys. Approximation algorithms for the bottleneck asymmetric traveling salesman problem. In *APPROX*, pages 1–11, 2010.

[AKS15] Hyung-Chan An, Robert D. Kleinberg, and David B. Shmoys. Improving Christofides' algorithm for the s-t path TSP. *J. ACM*, 62(5):34:1–34:28, 2015.

[AM08] Vikraman Arvind and Partha Mukhopadhyay. Derandomizing the isolation lemma and lower bounds for circuit size. In *APPROX and RANDOM*, pages 276–289, 2008.

[AV18] Nima Anari and Vijay V. Vazirani. Planar graph perfect matching is in NC. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 2018.

[AV19] Nima Anari and Vijay V. Vazirani. Matching is as easy as the decision problem, in the NC model. *CoRR*, abs/1901.10387, 2019.

[Bar92] D. A. M. Barrington. Quasipolynomial size circuit classes. In *Proceedings of the Seventh Annual Structure in Complexity Theory Conference*, pages 86–93, Jun 1992.

[BCH86] Paul W Beame, Stephen A Cook, and H James Hoover. Log depth circuits for division and related problems. *SIAM J. Comput.*, 15(4):994–1003, November 1986.

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984.

[Blä08] Markus Bläser. A new approximation algorithm for the asymmetric TSP with triangle inequality. *ACM Transactions on Algorithms*, 4(4), 2008.

[CFN85] Gérard Cornuéjols, Jean Fonlupt, and Denis Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, 33(1):1–27, 1985.

[CGK06] Moses Charikar, Michel X. Goemans, and Howard J. Karloff. On the integrality ratio for the asymmetric traveling salesman problem. *Math. Oper. Res.*, 31(2):245–252, 2006.

[Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Graduate School of Industrial Administration, CMU, 1976.

[CNN89] Marek Chrobak, Joseph Naor, and Mark B. Novick. *Using bounded degree spanning trees in the design of efficient algorithms on claw-free graphs*, pages 147–162. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989.

[Csa76] L. Csanky. Fast parallel inversion algorithm. *SIAM Journal of Computing*, 5:618–623, 1976.

[DHK93] E. Dahlhaus, P. Hajnal, and M. Karpinski. On the parallel complexity of Hamiltonian cycle and matching problem on dense graphs. *Journal of Algorithms*, 15(3):367 – 384, 1993.

[DK98] Elias Dahlhaus and Marek Karpinski. Matching and multidimensional matching in chordal and strongly chordal graphs. *Discrete Applied Mathematics*, 84(1-3):79–91, 1998.

[DKR10] Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory Comput. Syst.*, 47(3):737–757, 2010.

[DS84] Eliezer Dekel and Sartaj Sahni. A parallel matching algorithm for convex bipartite graphs and applications to scheduling. *Journal of Parallel and Distributed Computing*, 1(2):185–205, 1984.

[Edm65a] Jack Edmonds. Maximum matching and a polyhedron with $0,1$ vertices. *Journal of Research of the National Bureau of Standards*, 69:125–130, 1965.

[Edm65b] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[FGM82] Alan M. Frieze, Giulia Galbiati, and Francesco Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

[FGT16] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 754–763, 2016.

[FGT19] Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. A deterministic parallel algorithm for bipartite perfect matching. *Commun. ACM*, 62(3):109–115, February 2019.

[Fle74] Herbert Fleischner. The square of every two-connected graph is Hamiltonian. *Journal of Combinatorial Theory, Series B*, 16(1):29 – 34, 1974.

[FS07] Uriel Feige and Mohit Singh. Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007*, pages 104–118, 2007.

[GK87] Dima Grigoriev and Marek Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC. In *28th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 166–172, 1987.

[GS11] Shayan Oveis Gharan and Amin Saberi. The asymmetric traveling salesman problem on graphs with bounded genus. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 967–975, 2011.

[GSS11] Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pages 550–559, 2011.

[GT17] Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-NC. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 821–830, 2017.

[GTV18] Rohit Gurjar, Thomas Thierauf, and Nisheeth K. Vishnoi. Isolating a vertex via lattices: Polytopes with totally unimodular faces. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 74:1–74:14, 2018.

[Har09] Nicholas J. A. Harvey. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput.*, 39(2):679–702, 2009.

[Jac90] Carl Gustav Jacob Jacobi. De investigando ordine systematis aequationum differentialium vulgarium cujuscunque (published posthumously by C. W. Borchardt). *Borchardt Journal für die reine und angewandte Mathematik*, 64:297–320, 1890.

[Kar96] Alexander Karzanov. How to tidy up a symmmetric set-system by use of uncrossing operations. *Theoretical Computer Science*, 157:215–225, 1996.

[Kha79] Leonid G Khachiyan. A polynomial algorithm in linear programming. In *Doklady Akademii Nauk SSSR*, volume 244, pages 1093–1096, 1979.

[KLS13] Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. In *Algorithms and Computation - 24th International Symposium, ISAAC 2013*, pages 568–578, 2013.

[KLSS05] Haim Kaplan, Moshe Lewenstein, Nira Shafrir, and Maxim Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *J. ACM*, 52(4):602–626, 2005.

[KTV18] Anna Köhne, Vera Traub, and Jens Vygen. The asymmetric traveling salesman path LP has constant integrality ratio. *CoRR*, abs/1808.06542, 2018.

[KUW86] Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.

[KV12] Bernhard Korte and Jens Vygen. *Combinatorial optimization*. Springer, 2012.

[KVV85] Dexter Kozen, Umesh V. Vazirani, and Vijay V. Vazirani. NC algorithms for comparability graphs, interval gaphs, and testing for unique perfect matching. In *Proceedings of the Fifth Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 496–503, London, UK, 1985. Springer-Verlag.

[Lau81] H. T. Lau. Finding EPS-graphs. *Monatshefte für Mathematik*, 92(1):37–40, Mar 1981.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979.

[LPV81] G. F. Lev, N. Pippenger, and L. G. Valiant. A fast parallel algorithm for routing in permutation networks. *IEEE Transactions on Computers*, C-30(2):93–100, Feb 1981.

[LRS11] Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*. Cambridge University Press, 2011.

[MN89] G. L. Miller and J. Naor. Flow in planar graphs with multiple sources and sinks. In *30th Annual Symposium on Foundations of Computer Science*, pages 112–117, 1989.

[MS04] Marcin Mucha and Piotr Sankowski. Maximum matchings via Gaussian elimination. In *45th Symposium on Foundations of Computer Science (FOCS)*, pages 248–255, 2004.

[MS16] Tobias Mömke and Ola Svensson. Removing and adding edges for the traveling salesman problem. *J. ACM*, 63(1):2:1–2:28, 2016.

[Muc12] Marcin Mucha. 13/9-approximation for graphic TSP. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012*, pages 30–41, 2012.

[MV97] Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. 1997.

[MV00] Meena Mahajan and Kasturi R. Varadarajan. A new NC-algorithm for finding a perfect matching in bipartite planar and small genus graphs. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing, STOC*, pages 351–357, 2000.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[Nai82] M. Nair. On Chebyshev-type inequalities for primes. *The American Mathematical Monthly*, 89(2):126–129, 1982.

[NSV94] H. Narayanan, Huzur Saran, and Vijay V. Vazirani. Randomized parallel algorithms for matroid union and intersection, with applications to arboresences and edge-disjoint spanning trees. *SIAM J. Comput.*, 23(2):387–397, 1994.

[Par98] I. Parfenoff. An efficient parallel algorithm for maximum matching for some classes of graphs. *Journal of Parallel and Distributed Computing*, 52(1):96 – 108, 1998.

[PR82] Manfred W. Padberg and M. R. Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1):67–80, 1982.

[PR84] R.Gary Parker and Ronald L Rardin. Guaranteed performance heuristics for the bottleneck travelling salesman problem. *Operations Research Letters*, 2(6):269 – 272, 1984.

[PY82] Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM*, 29(2):285–309, April 1982.

[Rot17] Thomas Rothvoss. The matching polytope has exponential extension complexity. *Journal of the ACM*, 64(6):41, 2017.

[San18] Piotr Sankowski. NC algorithms for weighted planar perfect matching and related problems. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 97:1–97:16, 2018.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980.

[Sch03] Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer-Verlag, Berlin, 2003.

[Ser78] Anatoliy I. Serdyukov. On some extremal tours in graphs (in russian). *Upravlyaemye systemy*, 17:76–79, 1978.

[ST17] O. Svensson and J. Tarnawski. The matching problem in general graphs is in Quasi-NC. © 2017 IEEE. Reprinted, with permission. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 696–707, 2017.

[STV18a] Ola Svensson, Jakub Tarnawski, and László A. Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 204–213, New York, NY, USA, 2018.

[STV18b] Ola Svensson, Jakub Tarnawski, and László A. Végh. Constant factor approximation for ATSP with two edge weights. *Mathematical Programming*, 172(1-2):371–397, 2018.

[SV14] András Sebő and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, 34(5):597–629, 2014.

[Sve15] Ola Svensson. Approximating ATSP by relaxing connectivity. In *FOCS 2015: Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, 2015.

[Tut47] W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 22:107–111, 1947.

[Tut65] William Thomas Tutte. Lectures on matroids. *J. Research of the National Bureau of Standards (B)*, 69:1–47, 1965.

[TV12] Raghunath Tewari and N. V. Vinodchandran. Green's theorem and isolation in planar graphs. *Inf. Comput.*, 215:1–7, 2012.

[VY99] Santosh Vempala and Mihalis Yannakakis. A convex relaxation for the asymmetric tsp. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, pages 975–976, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[Wol80] Laurence A. Wolsey. *Heuristic analysis, linear programming and branch and bound*, pages 121–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 1980.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation*, pages 216–226, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg.

Jakub Tarnawski

http://jakub.tarnawski.org *jakub.tarnawski@epfl.ch*

Research interests

- Approximation algorithms, graph algorithms, combinatorial optimization, submodular maximization

Education & research positions

- Microsoft Research, Redmond, USA – Research Intern (05–08.2018), under supervision of Nikhil Devanur and Janardhan Kulkarni.
- École Polytechnique Fédérale de Lausanne, Switzerland – Doctoral Assistant (09.2014–present), Theory of Computation laboratory under supervision of Ola Svensson.
  - Simons Institute for the Theory of Computing – visiting graduate student (11–12.2017).
- École Polytechnique Fédérale de Lausanne, Switzerland – Summer@EPFL internship (07–08.2013), Theory of Computation laboratory under supervision of Aleksander Mądry.
- Faculty of Mathematics and Computer Science, University of Wrocław, Poland. Two majors: Computer Science and Mathematics, 2008–2014, MSc in both. GPA over 4.95/5.00.

Awards

- **Best Paper Award** at STOC 2018
- **Best Paper Award** at FOCS 2017
- Honorable mention in competition for best Masters thesis in computer science (Polish Society of Informatics, 2016)
- Scholarship of Polish Ministry of Education for academic achievements – 2008, 2011, 2013
- EU scholarship for top students – 2009, 2010, 2011
- Multiple scholarships for high GPA from University of Wrocław

Teaching experience

- Teaching Assistant for 8 semesters (Advanced Algorithms 2016–2018, Algorithms 2015–2018, Theory of Computation 2015), EPFL
- gave several lectures in Algorithms (2018) and Advanced Algorithms (2016, 2018), EPFL
- supervised a Master semester project (02–06.2016), EPFL
- taught exercises in Algorithms and Data Structures (graduate level) (02–06.2014), Wrocław
- taught supplementary tutorial in Logic For Computer Science for first-year students (10.2010–02.2011), Wrocław

Invited talks

- University of Bonn, Germany (Apr 2019)
- ETH Zürich, Switzerland (Mar 2019)
- Toyota Technological Institute of Chicago, USA (Feb 2019)
- University of California San Diego, USA (Feb 2019)
- Georgia Institute of Technology, USA (Jan 2019)
- BIRS workshop on TSP, Banff Centre, Canada (Sep 2018)
- Microsoft Research Redmond, USA (Jul 2018)
- Google Research Zürich, Switzerland (Apr 2018)
- Aussois, 22nd Combinatorial Optimization Workshop, France (Jan 2018)
- Stanford University, USA (Nov 2017)
- Georgia Institute of Technology, USA (Oct 2017)
- ETH Zürich, Switzerland (Sep 2017)
- 8th Cargese-Porquerolles Workshop on Combinatorial Optimization, France (Sep 2017)
- NII Shonan Meeting "Current Trends in Combinatorial Optimization", Japan (Apr 2016)
- University of Wrocław, Poland (Feb 2016)

Industry experience

- Facebook – Software Engineering Intern (07–09.2012, Seattle, USA). Traffic Infrastructure team. Performance optimization of the load balancing software that all of Facebook's web traffic passes through. Achieved a 30% gain in efficiency.

Competitive programming

- Onsite finals of Facebook Hacker Cup 2014 and 2015 (top 25, Menlo Park)
- ACM ICPC World Finals 2013 (St. Petersburg, Russia) and 2014 (Ekaterinburg, Russia; 13th place out of over 12000 teams)
- 1st place in IEEEXtreme 10.0 2016 (out of ~2000 teams), 2nd place in 2015, 3rd place in 2018
- 1st place in Wielka Przesmycka 2016 (Wrocław; open individual championship of Poland)

Service

- Reviewer for journals: Journal of the ACM, Theoretical Computer Science, Discrete Optimization
- Reviewer for conferences: SODA 2019, FOCS 2018, ICALP 2018, STOC 2018, STOC 2017, APPROX 2017, SWAT 2016, ESA 2014
- Reviewer of grant proposals: Polish National Science Center

- Head of problemsetting team at Helvetic Coding Contest, an annual programming competition held at EPFL (2015–2018); same for Santa's Programming Challenge (2014–2017)
- Contributed problems to Polish Collegiate Programming Contest AMPPZ (2015–2018)
- Maintainer and main author of open source project Hightail, a tool for competitive programming (> 6000 downloads)

References

- Prof. Ola Svensson (*ola.svensson@epfl.ch*), faculty at EPFL.
  Association: Co-author and PhD supervisor.
- Prof. Aleksander Mądry (*madry@mit.edu*), faculty at MIT.
  Association: Co-author and internship supervisor.
- Dr. Nikhil Devanur (*nikdev@microsoft.com*), researcher at Microsoft.
  Assocation: Internship supervisor.
- Prof. Amin Saberi (*saberi@stanford.edu*), faculty at Stanford.

Publications

- A. Norouzi-Fard, J. Tarnawski, S. Mitrović, A. Zandieh, A. Mousavifar and O. Svensson. Beyond 1/2-Approximation for Submodular Maximization on Massive Data Streams. In *35th International Conference on Machine Learning* (ICML), 2018, long talk.
- O. Svensson, J. Tarnawski and L. Végh. A Constant-Factor Approximation Algorithm for the Asymmetric Traveling Salesman Problem. In *50th Annual ACM Symposium on the Theory of Computing* (STOC), 2018. **Best Paper Award**
- O. Svensson and J. Tarnawski. The Matching Problem in General Graphs is in Quasi-NC. In *58th Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 2017. **Best Paper Award**
- S. Mitrović, I. Bogunović, A. Norouzi-Fard, J. Tarnawski and V. Cevher. Streaming Robust Submodular Maximization: A Partitioned Thresholding Approach. In *Neural Information Processing Systems* (NIPS), 2017.
- A. Mosińska, J. Tarnawski and P. Fua. Active Learning and Proofreading for Delineation of Curvilinear Structures. In *20th International Conference on Medical Image Computing and Computer Assisted Intervention* (MICCAI), 2017, oral presentation.
- C. Kalaitzis, O. Svensson and J. Tarnawski. Unrelated Machine Scheduling of Jobs with Uniform Smith Ratios. In *28th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), 2017.
- O. Svensson, J. Tarnawski and L. Végh. Constant Factor Approximation for ATSP with Two Edge Weights. *Mathematical Programming* 172(1–2), 2018. Previously in *18th Conference on Integer Programming and Combinatorial Optimization* (IPCO), 2016.
- A. Mądry, D. Straszak and J. Tarnawski. Fast Generation of Random Spanning Trees and the Effective Resistance Metric. In *26th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), 2015.

**Ce document a été imprimé au Centre d'impression EPFL,
imprimerie climatiquement neutre, certifiée myClimate.**