Approximation Algorithms for Traveling Salesman Problems

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Vera Traub

aus

Freiburg im Breisgau

Bonn, November 2019

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Erstgutachter:	Prof. Dr. Jens Vygen
Zweitgutachter:	Prof. Dr. Stephan Held
Tag der Promotion:	14.02.2020
Erscheinungsjahr:	2020

Acknowledgements

First and foremost, I would like to thank my advisor Prof. Dr. Jens Vygen for his encouragement and support. He was always happy to spend time on discussing new ideas and I learned a lot from working with him during the past years.

Besides Prof. Dr. Jens Vygen, I wish to thank my other collaborators Anna Köhne and Prof. Dr. Rico Zenklusen — it was a great pleasure to work with them.

Furthermore, I would like to express my gratitude to Prof. Dr. Dr. h.c. Bernhard Korte and Prof. Dr. Jens Vygen for providing excellent working conditions at the Research Institute for Discrete Mathematics.

During the past years, I attended several workshops and visited other universities. I am very grateful for this opportunity and I want to thank all people who made this possible and invited me to workshops or kindly hosted me at their institutions. Some of the research presented in this thesis results from a collaboration that was initiated during my visit at ETH Zurich.

Finally, I would like to thank my present and former colleagues at the Research Institute for Discrete Mathematics. I appreciated the friendly working atmosphere, many conversations about mathematics and other topics, and our collaboration on more applied projects. Although not directly related to this thesis, it made my time as a doctoral student much more enjoyable.

Contents

Chapter 1

Introduction

The traveling salesman problem is the probably most famous problem in combinatorial optimization. Given a set of cities, the task is to find a shortest tour visiting all of them. While this problem has various practical applications from logistics to genome sequencing, our main interest comes from its important role in combinatorial optimization. It has connections to many fundamental concepts such as T-joins, matroids, and network flows.

Let us now give a more formal definition of the traveling salesman problem. An instance consists of a (directed or undirected) graph $G = (V, E)$ and nonnegative edge cost $c : E \to \mathbb{R}_{\geq 0}$. The task is to find a closed walk in G that visits every vertex at least once and has minimum cost. When G is an undirected graph, we call this problem the symmetric traveling salesman problem, often abbreviated as TSP. When G is a directed graph, we call it the asymmetric traveling salesman problem, or in short ATSP.

An often useful equivalent definition is the following. Again, we are given a graph $G = (V, E)$ and edge cost $c : E \to \mathbb{R}_{\geq 0}$. Instead of a walk, we are now looking for a multi-set F of edges such that (V, F) is connected and Eulerian. An undirected graph (V, F) is Eulerian if every vertex has an even number of incident edges. A directed graph (V, F) is Eulerian if for every vertex v the number $|\delta_F^-(v)|$ of incoming edges equals the number $|\delta_F^+(v)|$ of outgoing edges. Via Euler's theorem one can easily see that this formulation is equivalent to our first definition of the traveling salesman problem.

Therefore, we can view the traveling salesman problem as a combination of connectivity constraints and additional constraints on the vertex degrees. If we have only one of these two types of constraints, the problem becomes easy: the problem of finding a minimum cost connected subgraph of G is the minimum cost spanning tree problem. It can be solved easily in polynomial time. We can also handle the second type of constraints on its own, both in the symmetric and the asymmetric case. In an undirected graph, we can compute in polynomial time a set of edges such that every vertex degree has a prescribed parity. This is the minimum cost T-join problem. In a directed graph, we can compute in polynomial time a minimum cost multi-set F of edges such that for every vertex v, the number $|\delta_F^+(v)| - |\delta_F^-(v)|$ matches a prescribed value. This is a minimum cost flow problem.

However, combining the connectivity constraints with the constraints on the vertex degrees makes the problem difficult. Both the symmetric and the asymmetric version of the traveling salesman problem are well-known to be *NP*-hard [Kar72].

For this reason, we will focus on finding approximation algorithms. An *α-approximation algorithm* computes a solution of cost at most α times the cost of an optimum tour and has polynomial runtime. We also say that the algorithm has *approximation ratio* α. For the symmetric TSP, Christofides' algorithm from the 1970's is a $\frac{3}{2}$-approximation algorithm [Chr76, Ser78]. Despite decades of research on TSP, this approximation ratio has not been improved. For ATSP, there is a classical $\log_2(n)$-approximation algorithm by Frieze, Galbiati, and Maffioli [FGM82], where $n := |V|$ denotes the number of vertices. The best-known approximation ratio remained $O(\log(n))$ until 2010, when Asadpour, Goemans, Mądry, Oveis Gharan, and Saberi gave an $O(\log(n)/\log(\log(n)))$-approximation algorithm [AGM+17]. Only very recently Svensson, Tarnawski, and Végh obtained the first constant-factor approximation algorithm for ATSP [STV18a]. They first showed an approximation ratio of 5500, which they later optimized to 506 [STV19].

Besides finding better approximation algorithms, another important question in the study of the traveling salesman problem is to determine the integrality ratio of the classical linear programming relaxations. These relaxations have been introduced by Dantzig, Fulkerson, and Johnson [DFJ54] and played a major role in the development of both approximation algorithms and algorithms that solve large instances exactly in practice [ABCC06]. The best known lower bounds on the integrality ratio can be proven by a family of unit-weight instances, i.e. instances with $c(e) = 1$ for every edge $e \in E$. This is one important motivation for studying these natural special cases with $c \equiv 1$, called graph TSP (in undirected graphs) and graph ATSP (in directed graphs). For these special cases, we know better approximation ratios and better upper bounds on the integrality ratios than for the general case.

One of the best-studied generalizations of TSP and ATSP is their path version. Here the start and end of the tour are given, but might be distinct. The special case of identical start and end is the classical traveling salesman problem that we have considered so far. At first glance, one might think that the generalization to the path version does not change much. However, there are several important differences. For example, Christofides' algorithm yields only a $\frac{5}{3}$-approximation when generalized to the path version [Hoo91]. For the symmetric unit-weight case, the integrality ratios of the classical linear programming relaxations are known to be different [SV14]. Also in many other cases, the best known approximation ratios and bounds on the integrality ratio differ (see Section 1.2). In contrast to TSP, for the path version we know better approximation algorithms than Christofides' algorithm. The first such algorithm was given in 2011 by An, Kleinberg, and Shmoys [AKS15].

Many other variants and special cases of the traveling salesman problem have been studied: TSP in the Euclidean plane [Aro98, Mit99, RS93, BG13], TSP in planar or, more generally, H-minor free graphs [Kle08, DHM10, DHK11, LWN17], TSP in cubic or subcubic graphs [GLS05, BSvdSS14, CLS15, DL18, DKM17], orienteering [BCK+07, BBCM04, CKP12, NR07], and many more. We will focus on the most basic variants of the traveling salesman problem and their path versions: we will only consider general (directed or undirected) graphs with either general nonnegative edge weights or unit weights $c \equiv 1$. Our main goal is to achieve better approximation algorithms and to obtain a better understanding of the classical linear programming relaxations and their integrality ratios.

1.1 Outline and contributions of this thesis

In Chapter 2 we introduce some basic notation and briefly summarize some well-known facts from combinatorial optimization that we will use in later parts of this thesis. After these preliminaries, the first main part of this thesis deals with the asymmetric traveling salesman problem and its path version s-t-path ATSP:

Chapter 3. This chapter provides an introduction to ATSP and its path version. First, we give an overview of prior work on this topic and introduce the classical linear programming relaxations. In the second part of the chapter we study the structure of dual solutions for the linear programs. We recall some known statements, but also prove the following new result.

It is well known that there always exists a dual solution with support corresponding to a laminar family $\mathcal{L}$ of vertex sets. This structure was exploited by Svensson, Tarnawski, and Végh in their recent constant-factor approximation algorithm for ATSP [STV18a]. We show that one can additionally achieve the following property: the elements of the laminar family $\mathcal{L}$ induce strongly connected subgraphs in the support graph of an optimum primal solution.

We will use this structure of dual solutions in the subsequent chapters to obtain approximation algorithms and upper bounds on the integrality ratio.

Chapter 4. In a recent breakthrough Svensson, Tarnawski, and Végh [STV18a] gave the first constant-factor approximation algorithm for ATSP and also showed that its classical LP relaxation has constant integrality ratio. We present a simpler algorithm that yields a much better approximation ratio.

While our new algorithm follows the framework of Svensson, Tarnawski, and Végh [STV19], we make several improvements throughout the different steps of the algorithm.

Using the structure of dual solutions obtained in Chapter 3, the first step is a reduction to so-called *vertebrate pairs* via a recursive algorithm. Vertebrate pairs are instances where we are already given a sub-tour (visiting some, but not all vertices) with particular properties. Svensson, Tarnawski, and Végh first reduce to what they call an *irreducible instance* as an intermediate step before reducing to vertebrate pairs. We show that this intermediate step is not necessary. Our new reduction is not only much simpler, but also leads to a much better approximation ratio.

To solve vertebrate pairs, Svensson, Tarnawski, and Végh [STV19] combine Svensson's algorithm [Sve15] with an algorithm for the *Subtour Cover problem* (see Section 4.3). Their Subtour Cover algorithm computes a minimum cost circulation in a certain flow network. One important step in the construction of the flow network is the computation of a so-called *witness flow*. We show that one can obtain a better approximation ratio by choosing the witness flow with a certain minimality property.

Svensson's algorithm [Sve15] (adapted to vertebrate pairs in [STV19]) uses a potential fuction to measure progress. In [STV19] two different potential functions were considered. One potential function yields a polynomial-time algorithm for

ATSP with approximation ratio 506. The other potential function yields an algorithm that may have exponential runtime but computes a better solution: it implies an upper bound of 319 on the integrality ratio. We present a new potential function that leads to a polynomial runtime and simultaneously implies the best known upper bound on the integrality ratio. This yields an improved approximation ratio that matches the upper bound on the integrality ratio up to an arbitrarily small $\varepsilon > 0$.

Overall, we obtain for every $\varepsilon > 0$ a $(22 + \varepsilon)$-approximation algorithm for ATSP, improving on the 506-approximation algorithm from [STV19]. Moreover, we give an upper bound of 22 on the integrality ratio of the classical LP relaxation, improving on the upper bound of 319 from [STV19].

For graph ATSP we give a $(13 + \varepsilon)$-approximation algorithm for any fixed $\varepsilon > 0$, improving on the $(27 + \varepsilon)$-approximation algorithm from [Sve15].

Chapter 5. In this chapter we study the path version of ATSP. We prove the first constant upper bound on the integrality ratio of its classical LP relaxation: we show an upper bound of 43 for s-t-path ATSP and an upper bound of 25 for its unit-weight special case. Moreover, for any $\varepsilon > 0$ we give approximation algorithms with approximation ratio $43 + \varepsilon$ for s-t-path ATSP and $25 + \varepsilon$ for its unit-weight special case.

Feige and Singh [FS07] gave the following black-box reduction from s-t-path ATSP to ATSP: if there is an α-approximation algorithm for ATSP, then for any fixed $\varepsilon > 0$ there is a $(2\alpha + \varepsilon)$-approximation algorithm for its path version. We give a similar black-box result for the integrality ratios of the classical LP relaxations: if ρ is the integrality ratio for ATSP, then the integrality ratio for s-t-path ATSP is at most $4\rho - 3$. For unit-weight instances we obtain a better bound: if ρ is the integrality ratio for graph ATSP, then the integrality ratio for s-t-path graph ATSP is at most $2\rho - 1$.

The main difficulty when generalizing the constant upper bound on the integrality ratio from ATSP to its path version is the following. For ATSP one can eliminate the dual LP variables corresponding to vertices, crucially using that ATSP solutions are Eulerian. This step fails for the s-t-path ATSP. In order to apply similar techniques as for ATSP in Chapter 4, one needs to upper bound the difference $a_s - a_t$ of the dual variables corresponding to the start s and the end t of the tour. In general, this term is unbounded. However, we show that there always exists an optimum dual solution, where $a_s - a_t$ is at most the optimum LP value. Using results from Chapter 4, we can then derive our constant upper bound on the integrality ratio. To prove our black-box reduction from s-t-path ATSP to ATSP, we combine our new bound on $a_s - a_t$ also with techniques from Feige and Singh [FS07].

In the second part of the thesis we consider the symmetric TSP and its path version s-t-path TSP:

Chapter 6. This chapter provides an introduction to approximation algorithms for TSP and its path version. We introduce the classical LP relaxations and give an overview of previous work and known results.

Chapter 7. In this chapter we prove a new upper bound on the integrality ratio of the classical LP relaxation for the s-t-path TSP. To show this new bound, we give an improved analysis of the Best-of-many Christofides algorithm with lonely edge deletion, which was proposed by Sebő and van Zuylen [SvZ19].

Like the earlier work [AKS15, Seb13, Vyg16, GV18], Sebő and van Zuylen start by writing an optimum LP solution as a convex combination of incidence vectors of spanning trees. This distribution of spanning trees is the starting point for the tour computation: from each of the trees they compute a tour and return the cheapest of them. Instead of bounding the cost of the cheapest tour directly, in the analysis one bounds the weighted average of the cost of the tours; in [AKS15, Seb13, Vyg16, GV18, SvZ19] the weights are the same as the weights of the spanning trees in the convex combination. In our improved analysis of the algorithm by Sebő and van Zuylen [SvZ19] we choose the weights differently. We show that this yields a better bound on the approximation ratio of the algorithm and the integrality ratio of the linear programming relaxation.

Chapter 8. In contrast to many other variants of the traveling salesman problem, for the s-t-path graph TSP the integrality ratio $\frac{3}{2}$ is known exactly. There is a well-known family of instances proving that the integrality ratio is at least $\frac{3}{2}$. In this chapter, we show that these classical examples are essentially the only instances with this property (up to small local differences).

We also derive a 1.497-approximation algorithm for the s-t-path graph TSP, thus achieving for the first time an approximation ratio below the integrality ratio of the classical LP relaxation.

Like the current best approximation algorithm for graph TSP by Sebő and Vygen [SV14], our approach uses ear-decompositions and ear-induction. The first step of our algorithm is to compute an ear-decomposition with certain structural properties. We introduce a new type of ear-decompositions that is optimized using matroid intersection. A similar step was used in [SV14], but we optimize larger parts of the ear-decomposition using laminar matroids instead of partition matroids. From the dual of this matroid intersection problem we derive a new lower bound for the length of an optimum s-t-tour.

Based on our optimized ear-decomposition, we apply a new kind of ear-induction that reveals a connection to matroid union. Applying also the removable pairings technique due to Mömke and Svensson [MS16], we obtain the following: if the distance of the start s and the end t of the tour is relatively small, we get a tour of length at most 1.497 times the value of an optimum LP solution. The smaller the distance between s and t is, the better is our bound. We use this to prove that all instances with integrality ratio close to $\frac{3}{2}$ are very similar to the classical examples with this property.

Combining our algorithm for the special case of small s-t-distance with a dynamic program, we obtain a 1.497-approximation algorithm for the s-t-path graph TSP. This dynamic programming approach will be further developed in the next chapter, where we prove an even better approximation ratio.

Chapter 9. In this chapter we prove that the s-t-path TSP is not much harder to approximate than its special case TSP. More precisely, we give the following black-

box reduction from s-t-path TSP to TSP: if there exists an α-approximation algorithm for TSP for some $\alpha > 1$, then for any $\varepsilon > 0$ there exists an $(\alpha + \varepsilon)$-approximation algorithm for its path version.

Our reduction also applies to the unit-weight case: any α-approximation algorithm for graph TSP implies an $(\alpha + \varepsilon)$-approximation algorithm for its path version. By applying this result to the $\frac{7}{5}$-approximation algorithm for graph TSP by Sebő and Vygen [SV14], we obtain a $(\frac{7}{5} + \varepsilon)$-approximation algorithm for the s-t-path graph TSP. This improves on the 1.497-approximation algorithm from Chapter 8.

To prove our reduction from the path version to TSP, we proceed as follows. We introduce a new generalization of s-t-path TSP, called Φ-TSP. For Φ-TSP we give a 4-approximation algorithm using Jain's iterative rounding framework [Jai01]. By assumption we are given an α-approximation algorithm for TSP for some $\alpha > 1$. Using this, we iteratively strengthen our Φ-TSP algorithm such that the approximation ratio improves, while the runtime increases. After a constant number of improvement steps, we obtain an approximation ratio of $(\alpha + \varepsilon)$. For a special case of Φ-TSP that includes the s-t-path TSP the overall runtime stays polynomial.

To perform an improvement step we distinguish two cases. In one case, we use a careful guessing of edges to reduce the problem to TSP. In the other case, we apply dynamic programming over a laminar family of cuts. We obtain this laminar family from the dual of a linear program for a minimum cost T-join problem. The dynamic program allows us to guess some parts of the optimum solution and partition the instance into several smaller instances of Φ-TSP. To those subinstances we then recursively apply our current approximation algorithm for Φ-TSP.

Finally, Chapter 10 contains some concluding remarks and open questions.

Chapter 7, Chapter 8, and parts of Chapter 4 are based on joint work with Jens Vygen. Moreover, Chapter 9 is based on joint work with Jens Vygen and Rico Zenklusen and parts of Chapter 5 are based on joint work with Anna Köhne and Jens Vygen.

1.2 State of the art

In this section we summarize the known results on the approximability and the integrality ratios of the classical linear programming relaxations. Here we only state the best known bounds. A more detailed discussion of the history and prior results will be given in Chapter 3 for ATSP and its path version and in Chapter 6 for the symmetric TSP and its path version.

1.2.1 Asymmetric traveling salesman problem

The following table summarizes the best upper bounds that we know on the integrality ratio of the classical LP relaxation and the approximation ratio for (graph) ATSP and its path version s-t-path (graph) ATSP. New bounds that we prove in this thesis are marked in red.

upper bounds	integrality ratio[1]		approximation ratio	
ATSP	22	[Corollary 4.40]	$22+\varepsilon$	[Theorem 4.39]
s-t-path ATSP	43	[Corollary 5.10]	$43+\varepsilon$	[Theorem 5.9]
graph ATSP	13	[Sve15]	$13+\varepsilon$	[Theorem 4.43]
s-t-path graph ATSP	25	[Corollary 5.15]	$25+\varepsilon$	[Corollary 5.15]

Moreover, we have the following black-box reductions from the path version of (graph) ATSP to (graph) ATSP.

- Let $\alpha > 1$ and $\varepsilon > 0$. If there is an α-approximation algorithm for ATSP, then there is a $(2\alpha+\varepsilon)$-approximation algorithm for its path version [FS07].
- Let $\varepsilon > 0$ and let ρ be the integrality ratio of the classical LP relaxation for ATSP. Then the integrality ratio of the classical LP relaxation of its path version is at most $4\rho - 3$ [Theorem 5.13].
- Let $\varepsilon > 0$ and let ρ be the integrality ratio of the classical LP relaxation for graph ATSP. Then the integrality ratio of the classical LP relaxation of its path version is at most $2\rho - 1$ [Theorem 5.14].

The first of the above reductions is due to Feige and Singh [FS07]. We will prove the other two reductions in Chapter 5. The following table shows the known lower bounds on the integrality ratio of the classical LP relaxation and lower bounds on the best possible approximation ratio; the approximability lower bounds assume $P \neq NP$.

lower bounds	integrality ratio[1]		approximation ratio	
ATSP	2	[CGK06]	$\frac{75}{74}$	[KLS15]
s-t-path ATSP	2	[CGK06]	$\frac{75}{74}$	[KLS15]
graph ATSP	2	[Corollary 4.46]	$\frac{685}{684}$	[KLS15]
s-t-path graph ATSP	2	[Corollary 4.46]	$\frac{685}{684}$	[KLS15]

[1] The bounds on the integrality ratio refer to the classical LP relaxations (ATSP LP) for ATSP and (ATSPP LP) for s-t-path ATSP that are defined in Chapter 3.

1.2.2 Symmetric traveling salesman problem

Now we consider the symmetric case. The best upper bounds that we know on the integrality ratio of the classical LP relaxation and the approximation ratio for (graph) TSP and its path version s-t-path (graph) TSP are shown in the following table, where again new bounds that we prove in this thesis are marked in red.

upper bounds	integrality ratio[2]		approximation ratio	
TSP	$\frac{3}{2}$	[Wol80]	$\frac{3}{2}$	[Chr76, Ser78]
s-t-path TSP	1.5284	[Theorem 7.6]	$\frac{3}{2}$	[Zen19]
graph TSP	$\frac{7}{5}$	[SV14]	$\frac{7}{5}$	[SV14]
s-t-path graph TSP	$\frac{3}{2}$	[SV14]	$\frac{7}{5}+\varepsilon$	[Corollary 9.4]

In Chapter 9 we will prove the following black-box reductions from s-t-path (graph) TSP to (graph) TSP.

- Let $\alpha > 1$ and $\varepsilon > 0$. If there is an α-approximation algorithm for TSP, then there is an $(\alpha+\varepsilon)$-approximation algorithm for s-t-path TSP [Corollary 9.2].
- Let $\alpha > 1$ and $\varepsilon > 0$. If there is an α-approximation algorithm for graph TSP, then there is an $(\alpha+\varepsilon)$-approximation algorithm for s-t-path graph TSP [Corollary 9.3].

The best known lower bounds on the integrality ratio of the classical LP relaxation and achievable approximation ratios are summarized in the following table; the lower bounds on the approximability assume $P \neq NP$.

lower bounds	integrality ratio[2]	approximation ratio	
TSP	$\frac{4}{3}$	$\frac{123}{122}$	[KLS15]
s-t-path TSP	$\frac{3}{2}$	$\frac{123}{122}$	[KLS15]
graph TSP	$\frac{4}{3}$	$\frac{685}{684}$	[KLS15]
s-t-path graph TSP	$\frac{3}{2}$	$\frac{685}{684}$	[KLS15]

Concerning the approximability, the lower bounds are quite far away from the upper bounds. The known bounds on the integrality ratio are much better, but the only case in which the integrality ratio is known exactly is the s-t-path graph TSP. This is also the only variant for which we know an algorithm with approximation ratio better than the integrality ratio (see Chapter 8).

[2]The bounds on the integrality ratio refer to the classical LP relaxations (TSP LP) for TSP and (TSPP LP) for s-t-path TSP that are defined in Chapter 6.

Chapter 2

Preliminaries

This chapter contains preliminaries on some basic concepts of combinatorial optimization. We give a brief summary of some well-known results and basic terminology that we will use in later parts of this thesis. For a more in-depth discussion of these topics we refer to textbooks on combinatorial optimization, e.g. [KV18] or [Sch03].

2.1 Notation

A *weighted graph* is a tuple (V, E, c), where V is the vertex set, E is the edge set, and $c : E \to \mathbb{R}$ denotes the edge *cost*. We will sometimes also call $c(e)$ the *weight* or the *length* of the edge e. Often we abbreviate $c(\{v, w\})$ or $c((v, w))$ by $c(v, w)$. For a (multi-)subset $F \subseteq E$ we define $c(F) := \sum_{e \in F} c(e)$. Similarly, for a vector $x \in \mathbb{R}^E$ we write $x(F) := \sum_{e \in F} x_e$ and $c(x) := \sum_{e \in E} c(e) \cdot x_e$.

We denote by $V(G)$ the vertex set of a graph G and by $E(G)$ its edge set. For a directed graph $G = (V, E)$ and $U \subseteq V$ we define

$$\delta^+(U) := \{(u, v) \in E : u \in U,\, v \in V \setminus U\}$$

to be the set of outgoing edges of U and $\delta^-(U) := \delta^+(V \setminus U)$ to be the set of incoming edges of U. For a vertex v we also write $\delta^+(v) := \delta^+(\{v\})$ and $\delta^-(v) := \delta^-(\{v\})$. For any directed or undirected graph $G = (V, E)$ and $U \subseteq V$ we write

$$\delta(U) := \{e \in E : \text{exactly one endpoint of } e \text{ is contained in } U\}.$$

We also write $\delta(v) := \delta(\{v\})$ for $v \in V$. If the graph G is not clear from the context, we sometimes write δ_G, δ_G^+, or δ_G^- instead of δ, δ^+, or δ^-. Similarly, for a (multi-)set F of edges we also write $\delta_F(U) := \{e \in F : \text{exactly one endpoint of } e \text{ is contained in } U\}$.

Let $G = (V, E)$ be a (directed or undirected) graph. For a vertex set U and an edge set F, we denote by $F[U]$ the set of edges in F for which both endpoints are contained in U. Moreover, $G[U]$ denotes the induced subgraph with vertex set U and edge set $E[U]$. For a set $U \subseteq V$ and a graph G, we denote by G/U the graph resulting from G by contracting the vertex set U, where $G/\emptyset := G$.

For an undirected graph $G = (V, E)$ and a multi-set F of edges, we define

$$\text{odd}(F) := \{v \in V : |\delta(v) \cap F| \text{ is odd}\}.$$

For an edge set $F \subseteq E$ we define its incidence vector $\chi^F \in \mathbb{R}^E$ to be the vector with $\chi^F_e = 1$ if $e \in F$ and $\chi^F_e = 0$ if $e \in E \setminus F$. For a multi edge set F, the component χ^F_e of the incidence vector χ^F denotes the number of copies of e that are contained in F.

Finally, for two sets A and B we denote by $A \dot\cup B$ their disjoint union and by $A \triangle B$ their symmetric difference.

2.2 Eulerian walks

An *Eulerian walk* in a (directed or undirected) graph G is a walk that contains every edge of G exactly once and every vertex at least once. A directed (multi-)graph G is called *Eulerian* if $|\delta^-(v)| = |\delta^+(v)|$ for all vertices $v \in V(G)$. We also call a multi-set F of directed edges *Eulerian* if $(V(G), F)$ is Eulerian.

Theorem 2.1 (Euler's theorem for directed graphs)**.** *Let G be a directed graph. There exists a closed Eulerian walk in G if and only if G is connected and Eulerian.*

Let $s, t \in V(G)$ with $s \neq t$. Then there exists an Eulerian s-t-walk in G if and only if G is connected, $|\delta^-(v)| = |\delta^+(v)|$ for all vertices $v \in V(G) \setminus \{s, t\}$ and $|\delta^-(s)| + 1 = |\delta^+(s)|$.

An undirected graph G is called *Eulerian* if $|\delta(v)|$ is even for all vertices $v \in V(G)$, i.e. $\mathrm{odd}(E(G)) = \emptyset$. We also call a multi-set F of undirected edges *Eulerian* if $(V(G), F)$ is Eulerian.

Theorem 2.2 (Euler's theorem for undirected graphs)**.** *Let G be an undirected graph. There exists a closed Eulerian walk in G if and only if G is connected and Eulerian.*

Let $s, t \in V(G)$ with $s \neq t$. Then there exists an Eulerian s-t-walk in G if and only if G is connected and $\mathrm{odd}(E(G)) = \{s, t\}$.

Walks as in Theorem 2.1 and Theorem 2.2 can be computed in linear time.

2.3 Ear-decompositions

Let $G = (V, E)$ be an undirected graph. Consider a sequence $P_0, P_1, \ldots, P_k$ of subgraphs of G, where P_0 contains a single vertex, and P_i for $i > 0$ is either

- a path such that $V(P_i) \cap \big(V(P_1) \cup \cdots \cup V(P_{i-1})\big)$ is the set of endpoints of P_i, or
- a cycle such that $\big|V(P_i) \cap \big(V(P_0) \cup \cdots \cup V(P_{i-1})\big)\big| = 1$.

We say that $P_0, P_1, \ldots, P_k$ is an *ear-decomposition* of G if

$$G \;=\; \big(V(P_0) \cup \cdots \cup V(P_k),\; E(P_1) \cup \cdots \cup E(P_k)\big).$$

We call the graphs $P_1, \ldots, P_k$ *ears*. See Figure 2.1.

Theorem 2.3 (Whitney [Whi32])**.** *An undirected graph has an ear-decomposition if and only if it is 2-edge-connected.*

An ear is called *open* if it is P_1 or it is a path; otherwise it is called closed. An ear-decomposition is called *open* if all its ears are open.

Theorem 2.4 (Whitney [Whi32])**.** *An undirected graph has an open ear-decomposition if and only if it is 2-vertex-connected.*

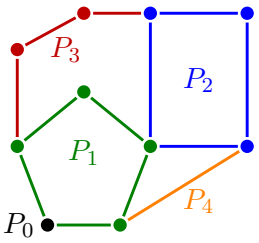

Figure 2.1: An ear-decomposition with four ears $P_1, \dots, P_4$. The graph P_0 consists of a single vertex (black). The edges of the first ear P_1 are shown in green. The ear P_2 (blue) is closed, P_3 (red) is open, and P_4 (orange) is an open ear with exactly one edge.

2.4 Laminar families

Definition 2.5. *Let V be a finite set and let $\mathcal{L}$ be a family of subsets of V. Then we call $\mathcal{L}$ a* laminar *family if for every two sets $X, Y \in \mathcal{L}$, we have $X \subseteq Y$ or $Y \subseteq X$ or $X \cap Y = \emptyset$.*

Let $v \in V$ such that v is contained in at least one element of $\mathcal{L}$. If $\mathcal{L}$ is a laminar family, then there is a unique minimal set $X \in \mathcal{L}$ containing v and a unique maximal set $Y \in \mathcal{L}$ containing v.

We will also often use the following simple observation.

Lemma 2.6. *Let V be a finite set and let $\mathcal{L}$ be a laminar family of non-empty subsets of V. Then $|\mathcal{L}| \leq 2|V| - 1$.*

2.5 Linear programming

In *linear programming* we are given a matrix $A \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. The task is the following:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & c^T x \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0. \end{aligned} \tag{2.1}$$

We call (2.1) a *linear program* or *LP*. A linear program might also be of a different but equivalent form; we always optimize a linear objective subject to some linear inequalities. Here, we stated the form that we will use most often in the context of TSP. The polyhedron

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

is the set of feasible solutions of the linear program (2.1). If A and b are rational, we also call the polyhedron P *rational*. If $P \neq \emptyset$, we call the LP (2.1) *feasible*. If for every number $C \in \mathbb{R}$, there exists some $x \in P$ with $c^T x < C$, we call (2.1) *unbounded*. Otherwise, we call it *bounded*.

Khachiyan [Kha79] proved that linear programming can be solved in polynomial time for rational input A, b, c. This result requires the matrix A, the right-hand side b, and the cost vector c to be given explicitly. Later Grötschel, Lovász, and Schrijver [GLS81] showed that it is sufficient if the set P of feasible solutions is given only

implicitly via a *separation oracle*. A *separation oracle* for a rational polytope P receives a vector $x \in \mathbb{Q}^n$ as an input; then it either certifies $x \in P$ or returns a vector $a \in \mathbb{Q}^n$ such that $a^T x > a^T x'$ for all $x' \in P$. Given such a separation oracle for rational P and the cost vector $c \in \mathbb{Q}^n$, one can solve the linear program (2.1) using only a polynomial number of oracle calls and further operations taking polynomial time.

LP duality

The *dual* of the linear program (2.1) is

$$\begin{aligned} \max_{y \in \mathbb{R}^m} \; & b^T y \\ \text{s.t.} \quad & A^T y \leq c \\ & y \geq 0. \end{aligned} \tag{2.2}$$

We have the following relation between the primal LP (2.1) and its dual LP (2.2).

Theorem 2.7 (strong duality)**.** *Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. If* (2.1) *is feasible and bounded, then also* (2.2) *is feasible and bounded. Moreover,*

$$\min \left\{ c^T x \;:\; Ax \geq b,\; x \geq 0,\; x \in \mathbb{R}^n \right\} = \max \left\{ b^T y \;:\; A^T y \leq c,\; y \geq 0,\; y \in \mathbb{R}^m \right\}.$$

We will also use the following properties of optimum primal and dual solutions.

Theorem 2.8 (complementary slackness)**.** *Let $x \in \mathbb{R}^n$ be a feasible solution of* (2.1) *and let $y \in \mathbb{R}^m$ be a feasible solution of* (2.2)*. Then x and y are both optimal solutions if and only if the following two conditions hold:*

- $x^T(c - A^T y) = 0$
- $y^T(Ax - b) = 0$.

LP relaxations and integrality ratios

Suppose we consider an optimization problem in which we want to find a minimum cost (multi-)set of edges subject to some constraints. Then we call a linear program a *relaxation* if the incidence vectors of feasible solutions to the optimization problem are feasible solutions to the linear program of the same cost. The value of the linear program yields a lower bound on the value of an optimum solution to our optimization problem. For a fixed optimization problem and a fixed LP relaxation, we denote by $\mathrm{OPT}(\mathcal{I})$ the cost of an optimum solution to the optimization problem on an instance $\mathcal{I}$ and we denote by $\mathrm{LP}(\mathcal{I})$ the optimum value of the LP relaxation. If the instance is clear from the context, we sometimes simply write OPT or LP. We define the *integrality ratio* to be the supremum of $\frac{\mathrm{OPT}(\mathcal{I})}{\mathrm{LP}(\mathcal{I})}$ over all instances $\mathcal{I}$ where the $\mathrm{OPT}(\mathcal{I})$ is non-zero; if for some instance $\mathcal{I}$ we have $\mathrm{OPT}(\mathcal{I}) > 0$ and $\mathrm{LP}(\mathcal{I}) = 0$, the integrality ratio is defined to be ∞. Sometimes the integrality ratio is also called *integrality gap*. We remark that there are also more general definitions of LP relaxations and integrality ratios in other contexts, but the above ones are sufficient for our purpose.

2.6 Matroids

Definition 2.9. *A* matroid $\mathcal{M} = (E, \mathcal{F})$ *consists of a finite ground set* E *and a family* $\mathcal{F}$ *of subsets of* E *such that the following three conditions hold:*

- $\emptyset \in \mathcal{F}$.
- *If* $B \in \mathcal{F}$, *then* $A \in \mathcal{F}$ *for all* $A \subseteq B$.
- *If* $A, B \in \mathcal{F}$ *and* $|B| > |A|$, *then there exists some* $e \in B$ *such that* $A \cup \{e\} \in \mathcal{F}$.

A set $I \in \mathcal{F}$ *is called* independent. *An (inclusionwise) maximal independent set is a* basis *of the matroid.*

One important example of matroids are *graphic matroids*. A graphic matroid arises from an undirected graph $G = (V, E)$. The ground set of the matroid is the edge set of the graph. A set F of edges is independent if and only if it does not contain the edge set of a cycle – in other words, (V, F) is a forest. If the graph G is connected, the bases of this matroid correspond to the spanning trees of G.

Another important example of matroids are *partition matroids*. Let E be some finite ground set and $\mathcal{P} = \{E_1, \dots, E_k\}$ a partition of E. Moreover, let $a_1, \dots, a_k \in \mathbb{Z}_{\geq 0}$ and

$$\mathcal{F} := \{A \subseteq E : |A \cap E_i| \leq a_i \text{ for all } i \in \{1, \dots, k\}\}.$$

Then $(E, \mathcal{F})$ is a *partition matroid*.

A generalization of partition matroids are *laminar matroids*. Let again E be some finite ground set. Now let $\mathcal{L} := \{E_1, \dots, E_k\}$ be a laminar family of subsets of E. Moreover, let again $a_1, \dots, a_k \in \mathbb{Z}_{\geq 0}$ and

$$\mathcal{F} := \{U \subseteq E : |U \cap E_i| \leq a_i \text{ for all } i \in \{1, \dots, k\}\}.$$

Clearly, $\mathcal{F}$ fulfills the first two conditions of Definition 2.9. To prove the third condition, consider sets $A, B \in \mathcal{F}$ with $|B| > |A|$. We call a set $E_i \in \mathcal{L}$ *tight* if $|A \cap E_i| = a_i$. Suppose there exists no $e \in B$ such that $A \cup \{e\} \in \mathcal{F}$. Then every element of B is contained in a tight set. We consider the family $\mathcal{L}'$ of maximal tight sets in $\mathcal{L}$. Since $\mathcal{L}$ is a laminar family, the elements of $\mathcal{L}'$ are disjoint. Therefore,

$$|A| \;\geq\; \sum_{E_i \in \mathcal{L}'} |A \cap E_i| \;=\; \sum_{i : E_i \in \mathcal{L}'} a_i \;\geq\; \sum_{E_i \in \mathcal{L}'} |B \cap E_i| \;=\; |B|,$$

a contradiction. Hence, $(E, \mathcal{F})$ is indeed a matroid. We call such a matroid a *laminar matroid*.

Definition 2.10. *Let* $\mathcal{M} = (E, \mathcal{F})$ *be a matroid. The* rank function $r : 2^E \to \mathbb{Z}_{\geq 0}$ *of the matroid* $\mathcal{M}$ *is given by*

$$r(F) \;:=\; \max\{|F'| : F' \subseteq F,\; F' \in \mathcal{F}\}.$$

Note that a set $U \subseteq E$ is independent if and only if $r(U) = |U|$. The rank function of any matroid is *submodular*, i.e. for any $A, B \subseteq E$ we have $r(A \cap B) + r(A \cup B) \leq r(A) + r(B)$.

Spanning trees and bases of matroids

Given a cost function $c : E \to \mathbb{R}$, a minimum cost basis of a matroid can be computed by the matroid greedy algorithm. In particular, we can compute a minimum cost spanning tree of a weighted graph $G = (V, E, c)$ in polynomial time. In fact, we can even compute a minimum cost spanning tree in time $O(|E| + |V| \cdot \log(|V|))$.

The convex hull of incidence vectors of bases of a matroid is called the *base polytope* of the matroid. It admits the following description.

Theorem 2.11 (Edmonds [Edm70])**.** *Let $\mathcal{M}$ be a matroid with ground set E and rank function r. Then*

$$\left\{x \in \mathbb{R}^E_{\geq 0} : x(E) = r(E),\ x(F) \leq r(F) \text{ for all } F \subseteq E\right\} \tag{2.3}$$

is the convex hull of incidence vectors of bases of $\mathcal{M}$.

A special case of this is the convex hull of incidence vectors of spanning trees of a graph G, also called the *spanning tree polytope*.

Corollary 2.12. *Let $G = (V, E)$ be an undirected graph. Then*

$$\left\{x \in \mathbb{R}^E_{\geq 0} : x(E) = |V| - 1,\ x(E[U]) \leq |U| - 1 \text{ for all } U \subseteq V\right\} \tag{2.4}$$

is the convex hull of incidence vectors of spanning trees of G.

In particular, every vector x contained in the polytope (2.4) can be written as a convex combination of incidence vectors of spanning trees.

Matroid intersection

Matroid intersection is the problem of finding a maximum cardinality common independent set in two matroids over the same ground set. More precisely, given two matroids $\mathcal{M}_1 = (E, \mathcal{F}_1)$ and $\mathcal{M}_2 = (E, \mathcal{F}_2)$, find a set $I \in \mathcal{F}_1 \cap \mathcal{F}_2$ maximizing $|I|$. The matroids are given either by some polynomial-time oracle for the rank function or, equivalently, by an independence oracle, i.e. an oracle that decides in polynomial time whether a given set $F \subseteq E$ is independent or not. The matroid intersection problem can be solved in polynomial time. Moreover, we have the following characterization of the optimum solution value.

Theorem 2.13 (Edmonds [Edm70])**.** *Let $\mathcal{M}_1 = (E, \mathcal{F}_1)$ be a matroid with rank function r_1. Let $\mathcal{M}_2 = (E, \mathcal{F}_2)$ be a matroid over the same ground set with rank function r_2. Then*

$$\max\{|I| : I \in \mathcal{F}_1 \cap \mathcal{F}_2\} \ = \ \min\{r_1(Q) + r_2(E \setminus Q) : Q \subseteq E\}.$$

In the *weighted matroid intersection* problem we are given again two matroids $\mathcal{M}_1$ and $\mathcal{M}_2$ with common ground set E, but now in addition a weight function $w : E \to \mathbb{R}_{\geq 0}$. Here, the task is to find a common independent set of the two matroids that has maximum weight. Also this problem can be solved in polynomial time.

The s-t-path ATSP can be described as the problem of finding a maximum weight independent set in three matroids. We may assume that the given directed graph $G = (V, E)$ is a complete graph and the edge cost c fulfill the triangle inequality; then

we can require that a solution to the s-t-path ATSP visits every vertex exactly once; see Chapter 3. In this formulation a solution for the s-t-path ATSP (with $s \neq t$) is a path from s to t that contains every vertex. The edge set of such a path is a common basis of three matroids whose ground set is the edge set of G. The first matroid is the graphic matroid in which a subset of E is independent if it does not contain an undirected cycle. The second matroid is a partition matroid in which a subset of E is independent if it contains at most one incoming edge of every vertex in $V \setminus \{s\}$ and no incoming edge of s. The third matroid is again a partition matroid in which a subset of E is independent if it contains at most one outgoing edge of every vertex in $V \setminus \{t\}$ and no outgoing edge of t. Moreover, with the weight function is $w(e) := 2 \cdot \sum_{f \in E} c(f) - c(e)$ for all $e \in E$, the maximum weight common independent set of these three matroids correspond precisely to the minimum cost solutions of the s-t-path ATSP. However, even finding a maximum cardinality common independent set of three matroids is *NP*-hard.

Matroid union

We will also need the matroid union theorem. Let $\mathcal{M}_1 = (E_1, \mathcal{F}_1)$ and $\mathcal{M}_2 = (E_2, \mathcal{F}_2)$ be two matroids. Then the *union* of $\mathcal{M}_1$ and $\mathcal{M}_2$ is

$$(E_1 \cup E_2,\ \{F_1 \cup F_2 : F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2\})\,.$$

This is again a matroid.

Theorem 2.14 (Nash-Williams [NW67])**.** *Let $\mathcal{M}_1 = (E_1, \mathcal{F}_1)$ be a matroid with rank function r_1 and let $\mathcal{M}_2 = (E_2, \mathcal{F}_2)$ be a matroid with rank function r_2. Then the rank function r of the union of $\mathcal{M}_1$ and $\mathcal{M}_2$ is given by*

$$r(U) \;=\; \min\Big\{|U \setminus Q| + r_1(Q \cap E_1) + r_2(Q \cap E_2) : Q \subseteq U\Big\}.$$

2.7 T-joins

Definition 2.15. *Let $G = (V, E)$ be an undirected graph and $T \subseteq V$ with $|T|$ even. Then a multi-set $J \subseteq E$ is a* T-join *if $T = \mathrm{odd}(J)$.*

An undirected graph G contains a T-join if and only if every connected component of G contains an even number of elements of T. Given such a graph G and cost $c : E \to \mathbb{R}_{\geq 0}$, we can compute in polynomial time a T-join of minimum cost. The convex hull of incidence vectors of T-joins is the *T-join polyhedron*. It can be described as follows.

Theorem 2.16 (Edmonds, Johnson [EJ73])**.** *Let $G = (V, E)$ be an undirected graph and $T \subseteq V$ with $|T|$ even. Then*

$$\Big\{x \in \mathbb{R}^E_{\geq 0} : x(\delta(U)) \geq 1 \text{ for all } U \subseteq V \text{ with } |U \cap T| \text{ odd}\Big\} \tag{2.5}$$

is the convex hull of incidence vectors of T-joins in G.

We call a cut $\delta(U)$ with $|U \cap T|$ odd a *T-cut*. Every vector in the T-join polyhedron dominates a convex combination of simple T-joins: for every vector x contained in (2.5),

there is a vector $x' \leq x$ (componentwise) such that x' is a convex combination of incidence vectors of T-joins without parallel edges. For any $c : E \to \mathbb{R}_{\geq 0}$ we have

$$\min\Big\{c(J) : J \text{ is a } T\text{-join}\Big\} \;=\; \min\Big\{c(x) : x \in \mathbb{R}^E_{\geq 0},\ x(C) \geq 1 \text{ for every } T\text{-cut } C\Big\}.$$

We can also describe the convex hull of incident vectors of simple T-joins.

Theorem 2.17 (Edmonds, Johnson [EJ73])**.** *Let $G = (V, E)$ be an undirected graph and $T \subseteq V$ with $|T|$ even. Then*

$$\Big\{x \in [0,1]^E : |F| - x(F) + x(\delta(U) \setminus F) \geq 1 \quad \text{for all } U \subseteq V \text{ and } F \subseteq \delta(U) \text{ with } |U \cap T| + |F| \text{ odd } \Big\}.$$

is the convex hull of incidence vectors of simple T-joins in G.

We call this the *T-join polytope*.

2.8 Network flows

Definition 2.18. *A flow network (G, u) consists of a weighted directed graph $G = (V, E, c)$ and edge capacities $u : E \to \mathbb{R}_{\geq 0}$. A* flow *in (G, u) is a function $f : E \to \mathbb{R}_{\geq 0}$ with $f(e) \leq u(e)$ for all $e \in E$. The* cost *of the flow is $c(f) := \sum_{e \in E} f(e) \cdot c(e)$.*

Let $b : V \to \mathbb{R}$ with $\sum_{v \in V} b(v) = 0$. A flow f is a b-flow *if $f(\delta^+(v)) - f(\delta^-(v)) = b(v)$ for all $v \in V(G)$.*

We define the *support* of a flow f to be the set $\mathrm{supp}(f) := \{e \in E : f(e) > 0\}$ of edges with positive flow value.

The following theorem characterizes flow networks (G, u) in which a b-flow exists.

Theorem 2.19. *Let (G, u) be a flow network and let $b : V \to \mathbb{R}$ with $\sum_{v \in V} b(v) = 0$. Then there exists a b-flow in (G, u) if and only if for every set $U \subseteq V$*

$$u(\delta^+(U)) \geq \sum_{v \in U} b(v).$$

A *circulation* is a b-flow for $b \equiv 0$. From Theorem 2.19 one can easily derive the following.

Theorem 2.20 (Hoffman's circulation theorem)**.** *Let $G = (V, E, c)$ be a weighted directed graph. Moreover, let $l : E \to \mathbb{R}_{\geq 0}$ and let $u : E \to \mathbb{R}_{\geq 0}$. Then a circulation $f : E \to \mathbb{R}_{\geq 0}$ with $l(e) \leq f(e) \leq u(e)$ for all $e \in E$ exists if and only if*

$$l(\delta^-(U)) \;\leq\; u(\delta^+(U)) \tag{2.6}$$

for all $U \subseteq V$.

If $b(v) > 0$ only for one vertex $s \in V$ and $b(v) < 0$ only for one vertex $t \in V$, we call a b-flow f also an *s-t-flow*. Then we say that $b(s) = -b(t)$ is the *value* of the flow f. The following theorem characterizes the maximum value of an s-t-flow in a flow network. It can be easily derived from Theorem 2.19 (and vice versa).

Theorem 2.21 (max-flow-min-cut theorem)**.** *Let (G, u) be a flow network and $s, t \in V(G)$. The maximum value of an s-t-flow in (G, u) equals the capacity of a minimum s-t-cut, i.e.*

$$\begin{aligned} &\max\Big\{\gamma : f \text{ an } s\text{-}t\text{-flow of value } \gamma \text{ in } (G,u)\Big\} \\ =\ &\min\Big\{u(\delta^+(U)) : \{s\} \subseteq U \subseteq V(G) \setminus \{t\}\Big\}. \end{aligned}$$

A maximum s-t-flow and a minimum s-t-cut can be computed in polynomial time.

In particular, a cut $\delta^+(U)$ with $u(\delta^+(U))$ minimum among all $\emptyset \neq U \subsetneq V(G)$ can be computed in polynomial time. Similarly, a cut $\delta(U)$ with $u(\delta(U))$ minimum an be computed in polynomial time.

Theorem 2.22 (flow decomposition)**.** *Let $G = (V, E)$ be a directed graph and let $b : V \to \mathbb{R}$ with $\sum_{v \in V} b(v) = 0$. Define*

$$\begin{aligned} V^+ &:= \{v \in V : b(v) > 0\} \\ V^- &:= \{v \in V : b(v) < 0\}. \end{aligned}$$

Given a b-flow $f : E \to \mathbb{R}_{\geq 0}$, we can compute in polynomial time

- *a set $\mathcal{P}$ of paths from V^+ to V^-,*
- *weights $\lambda_P > 0$ for $P \in \mathcal{P}$,*
- *a set $\mathcal{C}$ of cycles, and*
- *weights $\lambda_C > 0$ for $C \in \mathcal{C}$,*

such that

$$f \ = \ \sum_{P \in \mathcal{P}} \lambda_P \cdot \chi^{E(P)} + \sum_{C \in \mathcal{C}} \lambda_C \cdot \chi^{E(C)}.$$

Minimum cost flows

In the *minimum cost flow problem* we are given a flow network (G, u) and a function $b : V(G) \to \mathbb{R}$ with $\sum_{v \in V(G)} b(v) = 0$. The task is to compute a b-flow in (G, u) with minimum cost. This problem is a special case of linear programming and therefore solvable in polynomial time. Moreover, there are several faster algorithms specialized for the minimum cost flow problem.

Many of these algorithms use the so-called *residual graph*. For a fixed flow network (G, u) and a flow f in (G, u) we define the residual graph G_f as

$$G_f := \Big(V(G),\ \Big\{e \in E(G) : f(e) < u(e)\Big\} \,\dot\cup\, \Big\{\overleftarrow{e} \in E(G) : f(e) > 0\Big\}\Big),$$

where $\overleftarrow{e}$ for an edge $e = (u, v) \in E(G)$ is an edge from v to u with $c(\overleftarrow{e}) := -c(e)$. Note that if E contains both an edge (u, v) and an edge (v, u), then the residual graph might contain parallel edges even if G does not. We define the *residual capacities* $u_f : E(G_f) \to \mathbb{R}_{>0}$ to be

$$u_f(e) := \begin{cases} u(e) - f(e) & \text{if } e \in E(G) \\ f(e') & \text{if } e = \overleftarrow{e'} \text{ for some edge } e' \in E(G). \end{cases}$$

Note that all edges in G_f have a positive residual capacity. Using the residual graph, we can give the following characterization of a minimum cost flow.

Proposition 2.23. *Let (G, u) be a flow network and $b : V(G) \to \mathbb{R}$ with $\sum_{v \in V(G)} b(v) = 0$. Then a b-flow f in (G, u) has minimum cost if and only if the residual graph G_f does not contain any cycle of negative cost.*

It is easy to see that the condition in Proposition 2.23 is necessary: suppose there is a cycle of negative cost in G_f. Then we can improve the flow f by *augmenting* along this cycle: we can increase the value of f for every edge $e \in E(G)$ that is contained in the cycle and decrease the value for edges $e \in E(G)$ for which $\overleftarrow{e}$ is contained in the cycle.

Integral flows

Consider a weighted directed graph G with integral lower bounds $l(e)$ and upper bounds $u(e)$ on the flow $f(e)$ for all $e \in E(G)$. Moreover, suppose the function $b : V(G) \to \mathbb{Z}$ is integral. Then we can find an integral minimum cost b-flow, unless the instance is infeasible, i.e. unless there does not even exist a fractional feasible solution.

Theorem 2.24. *Let $G = (V, E, c)$ be a weighted directed graph. Let $l : E \to \mathbb{Z}_{\geq 0}$ and let $u : E \to \mathbb{Z}_{\geq 0}$. Moreover, let $b : V \to \mathbb{Z}$ with $\sum_{v \in V} b(v) = 0$. If the linear program*

$$\begin{aligned} \min\ & c(f) & & \\ \textit{s.t.}\quad & f(\delta^+(v)) - f(\delta^-(v)) = b(v) & & \textit{for all } v \in V \\ & f(e) \geq l(e) & & \textit{for all } e \in E \\ & f(e) \leq u(e) & & \textit{for all } e \in E. \end{aligned}$$

is feasible, then it has an integral optimum solution and we can find such a solution in polynomial time.

2.9 Steiner forest and iterative rounding

An instance of the *Steiner forest* problem consists of a weighted graph (V, E, c), a set $R \subseteq V$ of terminals, and a partition $\mathcal{C}$ of R. A feasible solution is a set $F \subseteq E$ such that for every $C \in \mathcal{C}$ and every two terminals $v, w \in C$, the graph (V, F) contains a v-w-path. So the task in the Steiner forest problem is to find such a solution F with minimum cost $c(F)$.

The Steiner forest problem is *NP*-hard, but there are 2-approximation algorithms known [GGP+94, Jai01]. The 2-approximation algorithm by Jain even applies to the following much more general setting. Consider a function $f : 2^V \to \mathbb{Z}_{\geq 0}$ and the linear program

$$\begin{aligned} \min\ & c(x) & & \\ \text{s.t.}\quad & x(\delta(U)) \geq f(U) & & \text{for all } U \subseteq V \\ & x_e \leq 1 & & \text{for all } e \in E \\ & x_e \geq 0 & & \text{for all } e \in E. \end{aligned} \tag{2.7}$$

Jain [Jai01] showed that one can find in polynomial time an integral solution to (2.7) of cost at most $2 \cdot \text{LP}$ if the following two conditions hold:

- the function f is weakly supermodular, i.e. for any two sets $A, B \subseteq V$ we have $f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$ or $f(A) + f(B) \leq f(A \setminus B) + f(B \setminus A)$; and
- given a vector $x \in [0,1]^E$, we can in polynomial time either find a set $U \subseteq V$ such that $f(U) > x(\delta(U))$ or decide that no such set exists.

To prove this, Jain developed an iterative rounding scheme. He proved that at least one variable of every extreme point solution of (2.7) has value at least $\frac{1}{2}$; then this variable is rounded up and one can iterate on the residual problem where this variable is fixed to 1. For the function f defined by

$$f(U) := \begin{cases} 1, & \text{if there exists } C \in \mathcal{C} \text{ such that } C \setminus U \neq \emptyset \text{ and } U \cap C \neq \emptyset \\ 0, & \text{else} \end{cases}$$

Jain's iterative rounding framework yields a 2-approximation algorithm for the Steiner forest problem.

Part I

The asymmetric traveling salesman problem and its path version

Chapter 3

ATSP and s-t-path ATSP

In the first part of this chapter we introduce the classical linear programming relaxations for ATSP and its path version. Moreover, we provide an overview of known approximation algorithms. In the second part of the chapter we then discuss properties of dual LP solutions. We show that we may assume that the dual solutions have a particular structure, strengthening a result from [STV18a].

3.1 Introduction and overview of previous work

Let us first recall the definition of the asymmetric traveling salesman problem (ATSP). Given a directed graph $G = (V, E)$ and nonnegative edge weights $c : E \to \mathbb{R}_{\geq 0}$, we want to find a minimum cost closed walk in G that visits all vertices. A *closed walk* in G is a sequence $v_0, v_1, \ldots, v_k$ with $v_0 = v_k$ and $(v_{i-1}, v_i) \in E$ for all $i \in \{1, \ldots, k\}$. We are looking for such a sequence that contains every vertex at least once and minimizes $\sum_{i=1}^{k} c(v_{i-1}, v_i)$.

In the metric closure of (G, c) we have a complete graph with vertex set V and the costs of an edge (v, w) is defined by the length of a shortest v-w-path in (G, c); then the edge cost $\bar{c}$ fulfill the triangle inequality $\bar{c}(u, v) + \bar{c}(v, w) \geq \bar{c}(u, w)$ for all $u, v, w \in V$. Every walk in G also is a walk in the metric closure of at most the same cost. Moreover, having a walk in the metric closure that visits every vertex, we get such a walk of the same cost in G by replacing every edge (v, w) by a shortest v-w-path. Hence, we may assume that G is complete and the triangle inequality $c(u, v) + c(v, w) \geq c(u, w)$ holds. If we make these two assumptions, we may require that every vertex is visited exactly once: whenever a vertex is visited more than once, we can shortcut the walk, i.e. skip the second occurrence of the vertex in the sequence $v_0, \ldots, v_{k-1}$; due to the triangle inequality this does not increase the cost.

A *subtour* is a multi edge set F such that (W, F) is connected and Eulerian for some $W \subseteq V$. By Euler's theorem (Theorem 2.1), subtours are the multi edge sets of closed walks in G. A *tour* in G is a multi edge set F such that (V, F) is connected and Eulerian. Tours are the multi edge sets of closed walks that visit all vertices. Hence ATSP is the problem of finding a minimum cost tour in G.

Note that an instance (G, c) of ATSP has a feasible solution if and only if G is strongly connected. The standard LP relaxation for ATSP is

$$\begin{aligned} &\min c(x) \\ &\text{s.t.} \qquad x(\delta^-(v)) - x(\delta^+(v)) = 0 \quad \text{for } v \in V \\ &\qquad\qquad x(\delta(U)) \geq 2 \quad \text{for } \emptyset \neq U \subsetneq V \\ &\qquad\qquad x_e \geq 0 \quad \text{for } e \in E. \end{aligned} \tag{ATSP LP}$$

The integral solutions of (ATSP LP) are precisely the incidence vectors of tours. Hence they correspond to closed walks that visit all vertices. The relaxation (ATSP LP) has a feasible solution if and only if G is strongly connected.

If G is a complete graph and the edge costs fulfill the triangle inequality, we could also add degree constraints $x(\delta^+(v)) = x(\delta^-(v)) = 1$ for all vertices $v \in V$. However, this does not change the optimum LP value as one can show by applying the directed splitting-off technique due to Mader [Mad82] similar to Cunningham (see [MMP90]) and Goemans and Bertsimas [GB93].

Because the separation problem for (ATSP LP) can be easily reduced to a minimum cut problem, one can solve (ATSP LP) in polynomial time. Alternatively, one can solve (ATSP LP) using an extended formulation with only a polynomial number of constraints.

3.1.1 Approximating ATSP

The first nontrivial approximation algorithm for ATSP was a $\log_2(n)$-approximation algorithm due to Frieze, Galbiati and Maffioli [FGM82]; here $n := |V|$ denotes the number of vertices. We now consider the formulation of ATSP where G is a complete graph and the triangle inequality holds. To compute a tour F, the algorithm of Frieze, Galbiati and Maffioli initially sets $F := \emptyset$. Then it iteratively does the following: first, select an arbitrary vertex from every connected component of (V, F) and denote by V' the set of selected vertices. Then compute a minimum cost cycle cover $\mathcal{C}$ in $G[V']$, i.e. a set of cycles (with at least two edges each) such that every vertex $v \in V'$ is contained in exactly one of them. Computing such a set of cycles can be reduced to a minimum weight perfect matching problem in a bipartite graph (or to a minimum cost flow problem) and can thus be done in polynomial time. Finally, add the edges of all the cycles in $\mathcal{C}$ to F and iterate until (V, F) is connected.

The result of this algorithm is clearly a feasible solution to ATSP because (V, F) remains Eulerian throughout the algorithm and is connected at the end. The cost of a single cycle cover $\mathcal{C}$ can be bounded by the length of an optimum tour: we can shortcut every tour in G to a tour in $G[V']$, which is a cycle cover consisting of a single cycle. Due to the triangle inequality, shortcutting does not increase the length of the tour. Moreover, since the number of connected components of (V, F) decreases in every iteration by at least a factor of two, we have at most $\log_2(n)$ iterations. This implies that the algorithm described above is indeed a $\log_2(n)$-approximation algorithm.

One can also bound the cost of the computed tour with respect to the optimum value LP of (ATSP LP) by proving that the cost of a single cycle cover $\mathcal{C}$ can be bounded by LP. This yields an upper bound of $\log_2(n)$ on the integrality ratio of (ATSP LP). The best known lower bound on the integrality ratio of this LP relaxation is 2, which

is due to Charikar, Goemans, and Karloff [CGK06]. In Section 4.6 we show that the integrality ratio is at least 2 even for unit-weight instances (with $c \equiv 1$).

The approximation ratio $\log_2(n)$ was improved to $0.99 \log_2(n)$ by Bläser [Blä03], to $0.842 \log_2(n)$ by Kaplan, Lewenstein, Shafrir, and Sviridenko [KLSS05], and to $\frac{2}{3} \log_2(n)$ by Feige and Singh [FS07]. However, asymptotically the best known approximation ratio for ATSP remained $\Theta(\log(n))$ until in 2010 Asadpour, Goemans, Mądry, Oveis Gharan, and Saberi gave an algorithm with approximation ratio $O(\log(n)/\log(\log(n)))$ [AGM+17].

Their algorithm is based on the concept of *thin trees*. They show that given an optimum solution x^* to (ATSP LP), there exists an (arbitrarily oriented) spanning tree (V, S) such that the cost of S can be bounded by $2 \cdot c(x^*)$ and and S is α-thin for $\alpha = 4\log(n)/\log(\log(n))$, i.e. $|S \cap C| \leq \alpha \cdot x^*(C)$ for every undirected cut C. Moreover, such a spanning tree can be computed in randomized polynomial time. Having such an oriented spanning tree, they compute a minimum cost multi-set F of edges such that $(V, S \dot\cup F)$ is a tour. Computing such an F is equivalent to a minimum cost flow problem. Using the α-thinness of S one can show $c(F) \leq 2\alpha \cdot c(x^*)$, which implies that the algorithm is an $O(\log(n)/\log(\log(n)))$-approximation algorithm with respect to the LP relaxation (ATSP LP).

Anari and Oveis Gharan [AOG15] proved the existence of an α-thin spanning tree S for $\alpha = \log(\log(n))^{O(1)}$. This implies that the integrality ratio of (ATSP LP) is at most $\log(\log(n))^{O(1)}$. However, it is not known if such a tree S can be found in polynomial time.

Major progress towards a constant-factor approximation algorithm was made by Svensson [Sve15]: he devised such an algorithm for the special case for node-weighted instances. In a node-weighted instance, we have a nonnegative weight c_v for every vertex v and the cost of an edge $(v, w) \in E$ is given as $c_v + c_w$. Notice that such an instance is not necessarily symmetric since G does not need to be a complete graph. Node-weighted instances are essentially equivalent to unit-weight instances, i.e. instances with $c(e) = 1$ for all $e \in E$; see Section 4.6. The result of Svensson [Sve15] for unit-weight instances was extended to two different edge weights by Svensson, Tarnawski, and Végh [STV18b].

In a recent breakthrough, Svensson, Tarnawski, and Végh [STV18a] devised the first constant-factor approximation algorithm for the general ATSP and they also proved that its standard LP relaxation has constant integrality ratio. In Chapter 4 we will present a variant of this algorithm by Svensson, Tarnawski and Végh that is simpler than the original one and achieves a much better approximation ratio.

3.1.2 The path version of ATSP

We now define the path version of the asymmetric traveling salesman problem, the s-t-path ATSP. Besides a directed graph $G = (V, E)$ and nonnegative edge weights $c : E \to \mathbb{R}_{\geq 0}$, we are given a start vertex s and an end vertex t. The goal is to find a minimum cost s-t-walk in G that contains all vertices. In other words, we are looking for a sequence $v_0 = s, v_1, \ldots, v_k = t$ that contains every vertex at least once, fulfills $(v_{i-1}, v_i) \in E$ for $i = 1, \ldots, k$, and minimizes $\sum_{i=1}^{k} c(v_{i-1}, v_i)$.

An *s-t-tour* in G is a multi edge set F such that (V, F) is connected and becomes Eulerian by adding one edge (t, s) for $s \neq t$. In the s-t-path ATSP we are looking for a

minimum cost s-t-tour in G. The classical linear programming relaxation is the ATSP path LP

$$\begin{aligned} &\min c(x) \\ &\text{s.t.} \qquad x(\delta^-(v)) - x(\delta^+(v)) = \begin{cases} -1, & \text{if } v = s \text{ and } s \neq t \\ 1, & \text{if } v = t \text{ and } s \neq t \\ 0, & \text{else} \end{cases} \qquad \text{(ATSPP LP)} \\ &\qquad\qquad x(\delta(U)) \geq 2 \quad \text{for } \emptyset \neq U \subseteq V \setminus \{s,t\} \\ &\qquad\qquad x_e \geq 0 \quad \text{for } e \in E. \end{aligned}$$

Note that the integral solutions of (ATSPP LP) are precisely the incidence vectors of s-t-tours. Hence they correspond to walks from s to t that visit all vertices. For $s = t$ the linear program (ATSPP LP) is equivalent to (ATSP LP). In this case the set of feasible solutions and the objective function are the same for those two linear programs.

Nagarajan and Ravi [NR08] proved that the integrality ratio of (ATSPP LP) is at most $O(\sqrt{n})$, where again $n = |V|$. This bound was improved to $O(\log(n))$ by Friggstad, Salavatipour, and Svitkina [FSS13] and to $O(\log(n)/\log(\log(n)))$ by Friggstad, Gupta, and Singh [FGM16]. In Chapter 5 we prove that the integrality ratio of (ATSPP LP) is in fact constant.

Feige and Singh [FS07] showed that any α-approximation algorithm for ATSP implies a $(2\alpha + \varepsilon)$-approximation algorithm for s-t-path ATSP (for any $\varepsilon > 0$). Hence s-t-path ATSP also has a constant-factor approximation algorithm. In Chapter 5 we will prove a similar relation for the integrality ratios of the LP relaxations.

3.2 Structured dual LP solutions

We now study properties of the dual of the classical linear programming relaxations of ATSP and its path version. We will use these in the next chapters to derive both approximation algorithms and upper bounds on the integrality ratios. We consider the dual LP of (ATSP LP):

$$\begin{aligned} &\max \sum_{\emptyset \neq U \subsetneq V} 2y_U \\ &\text{s.t.} \quad a_w - a_v + \sum_{U : e \in \delta(U)} y_U \leq c(e) \quad \text{for } e = (v,w) \in E \qquad \text{(ATSP DUAL)} \\ &\qquad\qquad y_U \geq 0 \quad \text{for } \emptyset \neq U \subsetneq V, \end{aligned}$$

and the dual LP of (ATSPP LP):

$$\begin{aligned} &\max \; a_t - a_s + \sum_{\emptyset \neq U \subseteq V \setminus \{s,t\}} 2y_U \\ &\text{s.t.} \qquad a_w - a_v + \sum_{U : e \in \delta(U)} y_U \leq c(e) \quad \text{for } e = (v,w) \in E \qquad \text{(ATSPP DUAL)} \\ &\qquad\qquad y_U \geq 0 \quad \text{for } \emptyset \neq U \subseteq V \setminus \{s,t\}. \end{aligned}$$

For $s = t$ the linear program (ATSPP DUAL) is equivalent to (ATSP DUAL) in the following sense. Any feasible solution to (ATSPP DUAL) can be turned into a feasible solution for (ATSP DUAL) with the same objective value by setting $y_U := 0$ for every set U with $s \in U$. Moreover any feasible solution $(\bar{a}, \bar{y})$ to (ATSP DUAL) can be turned into a feasible solution (a, y) for (ATSPP DUAL) with the same objective value by setting $a := \bar{a}$ and for $\emptyset \neq U \subseteq V \setminus \{s, t\}$ setting $y_U := \bar{y}_U + \bar{y}_{V \setminus U}$.

Lemma 3.1. *We can in polynomial time either compute an optimum solution to* (ATSPP LP) *and* (ATSPP DUAL) *or decide that no optimum solution exists.*

Proof. We first solve the primal LP, i.e. (ATSPP LP) or (ATSPP LP), by the ellipsoid method (using a separation oracle for the exponentially many cut constraints). We record the sets U for which the separation oracle produced a constraint $x(\delta(U)) \geq 2$, and set all y-variables of other sets to zero in the dual LP. Then the primal solution is also optimal with respect to this subset of constraints, and hence restricting the dual LP to these variables does not change the optimum value. But now the restricted dual LP has a polynomial number of variables and constraints, so we can solve it in polynomial time. □

In this section we show that we can make several assumptions on the structure of our optimum dual LP solution (a, y).

The *support* of y is the set of nonempty subsets U of $V \setminus \{s, t\}$ for which $y_U > 0$. We denote it by $\text{supp}(y)$. First we observe that we may assume that the support of y is a laminar family. This is a well known result (see e.g. [STV18a]) that one can prove using the standard technique of uncrossing.

Lemma 3.2. *Let (a, y) be an optimum solution to* (ATSPP DUAL)*. Then there is a vector y' such that (a, y') is an optimum solution to* (ATSPP DUAL)*, and $\mathcal{L} := \{U : y_U > 0\}$ is laminar. Such a vector y' can be computed in polynomial time.*

Proof. As long as there exist vertex sets A and B such that $y_A > 0$, $y_B > 0$, and A and B cross, i.e. $A \cap B \neq \emptyset$, $A \setminus B \neq \emptyset$, and $B \setminus A \neq \emptyset$, we could change y as follows. For $\varepsilon := \min\{y_A, y_B\} > 0$ we could decrease y_A and y_B by ε and increase $y_{A \setminus B}$ and $y_{B \setminus A}$ by ε while maintaining dual feasibility. Karzanov [Kar96] showed that the uncrossing as above can be performed in polynomial time if one chooses crossing sets A and B in a suitable way. □

We say that a dual LP solution as in Lemma 3.2 has *laminar support*.

Definition 3.3. *A solution (a, y) to* (ATSPP DUAL) *has* laminar support *if for any two sets $A, B \in \text{supp}(y)$ we have $A \cap B = \emptyset$, $A \subseteq B$, or $B \subseteq A$.*

Next, we show that we can assume even stronger properties of our dual LP solutions.

Definition 3.4. *Let (G, c) be an instance of ATSP or (G, c, s, t) an instance of its path version. Moreover, let (a, y) be a dual LP solution, i.e. a solution to* (ATSP DUAL) *or* (ATSPP DUAL)*, respectively. We say that y and (a, y) have* strongly laminar support *if*

- *$\mathcal{L} := \{U : y_U > 0\}$ is a laminar family, and*
- *for every set $U \in \mathcal{L}$, the graph $G[U]$ is strongly connected.*

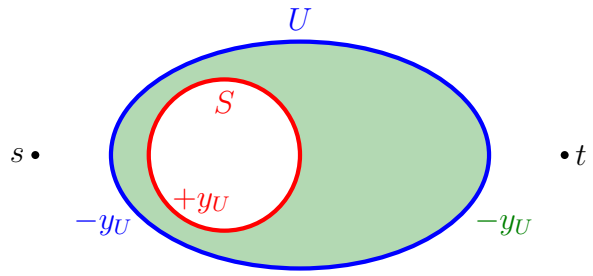

Figure 3.1: Illustration of the proof of Lemma 3.5. The set $U \subseteq V \setminus \{s,t\}$ is shown in blue. In red the vertex set S of the first strongly connected component of $G[U]$ in a topological order is shown. We have $\delta^-(S) \subseteq \delta^-(U)$. We modify our dual solution by increasing the dual variable corresponding to S, decreasing the dual variable corresponding to U, and decreasing the variables corresponding to the vertices in $U \setminus S$ (shown in green).

The following lemma allows us to assume that our optimum dual solution has strongly laminar support. We will use this new observation to simplify the constant-factor approximation algorithm for ATSP from [STV18a, STV19] and to obtain a constant-factor approximation algorithm for s-t-path ATSP.

Lemma 3.5. *Let (G, c, s, t) be an instance of the path version of ATSP. Moreover, let x be an optimum solution to* (ATSPP LP) *and (a, y) an optimum solution to the LP* (ATSPP DUAL). *Suppose that $x_e > 0$ for all $e \in E(G)$. Then we can compute in polynomial time (a', y') such that*

- *(a', y') is an optimum solution of* (ATSPP DUAL),
- *(a', y') has strongly laminar support, and*
- *$a'_s = a_s$ and $a'_t = a_t$.*

Proof. First, we compute an optimum solution (a, y) to (ATSPP DUAL). This can be done in polynomial time by Lemma 3.1. Then we apply Lemma 3.2 to obtain an optimum dual solution $(a, \bar{y})$ such that the support of $\bar{y}$ is a laminar family $\mathcal{L}$.

As long as there is a set U with $\bar{y}_U > 0$, but $G[U]$ not strongly connected, we do the following. Let U be a minimal set with $\bar{y}_U > 0$ and such that $G[U]$ is not strongly connected. Moreover, let S be the vertex set of the first strongly connected component of $G[U]$ in a topological order. Then we have $\delta^-(S) \subseteq \delta^-(U)$.

Define a dual solution (a', y') as follows. We set $y'_U := 0$, $y'_S := \bar{y}_S + \bar{y}_U$, and $y'_W := \bar{y}_W$ for other sets W. Moreover, $a'_v := a_v - \bar{y}_U$ for $v \in U \setminus S$ and $a'_v := a_v$ for all other vertices v. See Figure 3.1. Because $s, t \notin U$, we have $a'_s = a_s$ and $a'_t = a_t$. The edges from $U \setminus S$ to S are the only edges $e = (v, w)$ for which the left hand-side of the corresponding constraint in (ATSPP DUAL) increases, i.e. $a'_w - a'_v + \sum_{U : e \in \delta(U)} y'_U > a_w - a_v + \sum_{U : e \in \delta(U)} \bar{y}_U$. However, such edges do not exist by the choice of S. Hence, (a', y') is a feasible dual solution. Since $a'_t - a'_s + \sum_{\emptyset \neq U \subseteq V(G) \setminus \{s,t\}} 2y'_U = a_t - a_s + \sum_{\emptyset \neq U \subseteq V(G) \setminus \{s,t\}} 2\bar{y}_U$, it is also optimum.

We now show that the support of y' is laminar. Suppose there is a set W in the support of y' that crosses S, i.e. $W \setminus S \neq \emptyset$, $S \setminus W \neq \emptyset$ and $S \cap W \neq \emptyset$. Then W must be in the support of $\bar{y}$ and hence a subset of U because the support of $\bar{y}$ is laminar. By

the minimal choice of U, $G[W]$ is strongly connected. But this implies that G contains an edge from $W \setminus S$ to $W \cap S$, contradicting $\delta^-(S) \subseteq \delta^-(U)$.

We now decreased the number of sets U in the support for which $G[U]$ is not strongly connected. After iterating this at most $2|V(G)|$ times (by Lemma 2.6) the dual solution has the desired properties. □

One advantage of this structure is the following.

Lemma 3.6. *Let $G = (V, E)$ be a strongly connected directed graph and let $\mathcal{L}$ be a laminar family such that $G[U]$ is strongly connected for every $U \in \mathcal{L}$. Let $v, w \in V$ and let $\tilde{U}$ be the minimal set in $\mathcal{L} \cup \{V\}$ with $v, w \in \tilde{U}$.*

Then there is a v-w-path in $G[\tilde{U}]$ that enters and leaves every set $U \in \mathcal{L}$ at most once; we will call such a path nice. *A nice v-w-path can be found in polynomial time.*

Proof. Let P be a path from v to w in $G[\tilde{U}]$. Now repeat the following, until P enters and leaves every set in $\mathcal{L}$ at most once. Let U be a maximal set with $U \in \mathcal{L}$ that P enters or leaves more than once. Let v' be the first vertex that P visits in U and let w' be the last vertex that P visits in U. Since $G[U]$ is strongly connected, we can replace the v'-w'-subpath of P by a path in $G[U]$. By Lemma 2.6, P is a nice v-w-path after at most $|\mathcal{L}| < 2|V|$ iterations. □

We now introduce the notion of *strongly laminar instances*. This captures all the useful assumptions that we can make on the structure of our ATSP instance and the optimum primal and dual LP solutions. We will then show that we can reduce general ATSP instances to strongly laminar ATSP instances (see Theorem 3.8). Note however, that this reduction will not apply to the path version of ATSP.

Definition 3.7. *A* strongly laminar ATSP instance *is a quadruple $(G, \mathcal{L}, x, y)$, where*

(i) $G = (V, E)$ is a strongly connected directed graph;

(ii) $\mathcal{L}$ is a laminar family of subsets of V such that $G[U]$ is strongly connected for all $U \in \mathcal{L}$;

(iii) x is a feasible solution to (ATSP LP) *such that $x(\delta(U)) = 2$ for all $U \in \mathcal{L}$;*

(iv) $y \in \mathbb{R}^{\mathcal{L}}_{\geq 0}$.

This induces the ATSP instance (G, c), where c is the induced weight function *defined by $c(e) := \sum_{U \in \mathcal{L}: e \in \delta(U)} y_U$ for all $e \in E$.*

A solution to a strongly laminar ATSP instance $\mathcal{I}$ is a solution to its induced ATSP instance $\mathcal{I}' = (G, c)$. By complementary slackness (Theorem 2.8), x and $(0, y)$ are optimum solutions of (ATSP LP) and (ATSP DUAL) for (G, c), where we set $y_U := 0$ for $U \notin \mathcal{L}$. We define $\mathrm{LP}(\mathcal{I}) := c(x) = \sum_{L \in \mathcal{L}} 2y_L = \mathrm{LP}(\mathcal{I}')$.

We now prove that for ATSP it is sufficient to consider strongly laminar instances. Svensson, Tarnawski, and Végh showed that it suffices to consider *laminarly weighted* instances [STV18a]. The difference between laminarly weighted instances and strongly laminar instances is that in a strongly laminar instance we require that $G[U]$ is strongly connected for all $U \in \mathcal{L}$.

Theorem 3.8. *Let $\alpha \geq 1$. Suppose there is a polynomial-time algorithm that computes for every strongly laminar ATSP instance $\mathcal{I}' = (G, \mathcal{L}, x, y)$ a solution of cost at most $\alpha \cdot \mathrm{LP}(\mathcal{I}')$. Then there is a polynomial-time algorithm that computes for every feasible instance $\mathcal{I}$ of ATSP a solution of cost at most $\alpha \cdot \mathrm{LP}(\mathcal{I})$, where $\mathrm{LP}(\mathcal{I})$ denotes the cost of an optimum solution to* (ATSP LP).

Proof. Let (G, c) be an arbitrary instance of ATSP (with at least one vertex) and let $s = t \in V(G)$ arbitrary. If (ATSPP LP) is feasible, the graph G is strongly connected. Then we compute an optimum solution x of (ATSPP LP) and an optimum solution (a, y) of (ATSPP DUAL). By Lemma 3.1, this can be done in polynomial time. Since (ATSP LP) and (ATSPP LP) for $s = t$ are equivalent, x is also an optimum solution to (ATSP LP).

Now let $E' := \{e \in E : x_e > 0\}$ be the support of x and define $G' := (V, E')$. Let x' be the vector x restricted to its support E'. Then we apply Lemma 3.5 to the instance (G', c, s, t) and the LP solutions x' and (a, y). We obtain an optimum dual solution (a', y') to (ATSPP DUAL) with strongly laminar support $\mathcal{L} := \mathrm{supp}(y')$. By complementary slackness (Theorem 2.8) we have $x'(\delta(U)) = 2$ for all $U \in \mathcal{L}$.

Let y'' be the restriction of y' to its support $\mathcal{L}$. The induced weight function of the strongly laminar ATSP instance $\mathcal{I}' = (G', \mathcal{L}, x', y'')$ is given by

$$c'(e) \;=\; \sum_{L \in \mathcal{L} : e \in \delta(L)} y''_U \;=\; \sum_{L \in \mathcal{L} : e \in \delta(L)} y'_L \;=\; c(e) + a_v - a_w$$

for all $e = (v, w) \in E'$, where we used complementary slackness (Theorem 2.8) in the last equation. Because every tour in G' is Eulerian, it has the same cost with respect to c and with respect to c'. Moreover, because $x'(\delta^-(v)) - x'(\delta^+(v)) = 0$ for all $v \in V$, we have $\mathrm{LP}(\mathcal{I}') = c'(x') = c(x) = \mathrm{LP}(\mathcal{I})$ and thus the theorem follows. □

We would like to emphasize that the proof of Theorem 3.8 does not transfer to the path version of ATSP. The fact that this reduction fails for the path version turns out to be the main difficulty when generalizing the proof of the constant integrality ratio from (ATSP LP) to (ATSPP LP).

Remark 3.9
In the definition of strongly laminar instances we could also require $x_e > 0$ for all $e \in E$ and $y_L > 0$ for all $L \in \mathcal{L}$. Then the reduction in Theorem 3.8 would still work. However, it will be useful that the statements that we will proof for strongly laminar instances in Chapter 4 also apply to the case where $y_L = 0$ is allowed. We will use this in Chapter 5 where we consider the path version of ATSP.

Chapter 4

A $(22+\varepsilon)$-approximation algorithm for ATSP

In [STV18a] Svensson, Tarnawski, and Végh gave the first constant-factor approximation algorithm for ATSP. There they proved an approximation ratio of roughly 5500. Later, in [STV19] they optimized the approximation ratio and gave a 506-approximation algorithm for ATSP and an upper bound of 319 on the integrality ratio.

In this chapter we present an improved and slightly simplified version of their algorithm. We achieve an approximation ratio of $(22+\varepsilon)$ (for any fixed $\varepsilon > 0$) and show that the integrality ratio of the standard LP relaxation is at most 22.

Section 4.2 and Section 4.4 of this chapter are based on joint work with Jens Vygen. We first give a brief outline and explain where the algorithm we present here differs from the algorithm by Svensson, Tarnawski, and Végh.

4.1 Outline

We have already seen in the previous chapter that it is sufficient to find a constant factor approximation algorithm for strongly laminar ATSP instances $(G, \mathcal{L}, x, y)$; see Theorem 3.8. Here the difference to the algorithm in [STV19] is that we can assume that $G[L]$ is strongly connected for all $L \in \mathcal{L}$ which simplifies the subsequent steps.

In Section 4.2 we reduce strongly laminar instances to even more structured instances called *vertebrate pairs*. In a vertebrate pair we already have a given subtour, called *backbone*, that visits not necessarily all vertices but all non-singleton elements of the laminar family $\mathcal{L}$. In contrast to the reduction to strongly laminar instances, the reduction to vertebrate pairs causes some loss in the approximation ratio. While Svensson, Tarnawski, and Végh also reduce to vertebrate pairs, they first reduce to what they call an *irreducible instance* as an intermediate step before reducing to vertebrate pairs. We show that this intermediate step is not necessary which leads to a simpler algorithm. Moreover, the loss in the approximation ratio in this step is much smaller. In fact, a significant part of the improvement of the overall approximation ratio is due to the simplification of the reduction to vertebrate pairs.

Finally, in Section 4.3 and Section 4.4 we explain how to compute good solutions for vertebrate pairs. The main algorithmic framework, essentially due to Svensson [Sve15], follows on a very high level the cycle cover approach by Frieze, Galbiati and Maffi-

oli [FGM82]. However, in order to achieve a constant factor approximation for ATSP the algorithm and its analysis are much more involved and need many additional ideas.

In Section 4.3 we explain a sub-routine that computes solutions for what we call the *Subtour Cover* problem. This can be viewed as the analogue of the cycle cover problem that is solved in every iteration of the $\log_2(n)$-approximation algorithm by Frieze, Galbiati and Maffioli [FGM82]. It is very similar to what Svensson, Tarnawski, and Végh call *Subtour Partition Cover* and Svensson [Sve15] calls *Local Connectivity ATSP*. Svensson, Tarnawski, and Végh compute a solution for Subtour Cover by rounding a circulation in a certain flow network, which is constructed from the LP solution using a so-called *witness flow*. By using a special witness flow with certain minimality properties our Subtour Cover solution will obey stronger bounds.

In Section 4.4 we then explain how to compute solutions for vertebrate pairs using the algorithm for *Subtour Cover* as a sub-routine. The essential idea is due to Svensson [Sve15], who considered node-weighted instances, and was later adapted to vertebrate pairs in [STV19]. In this part we make two improvements compared to the algorithm in [STV19].

The more important change is the following. Svensson's algorithm uses a potential function to measure progress, and in each of [Sve15] and [STV19] two different potential functions are considered. One potential function is used to obtain an exponential time algorithm that yields an upper bound on the integrality ratio of the linear programming relaxation, and the other potential function is used to obtain a polynomial time algorithm. This leads to different upper bounds on the integrality ratio of the LP and the approximation ratio of the algorithm. We show in Section 4.4 that we can make this discrepancy arbitrarily small by a slightly different choice of the potential function for the polynomial time algorithm. This leads to a better approximation ratio. Moreover, the analysis of the polynomial time algorithm then immediately implies the best upper bound we know on the integrality ratio and there is no need anymore to consider two different potential functions.

The second change compared to the algorithm in [STV19] is that we include an idea that Svensson [Sve15] used for node-weighted instances. This leads to another small improvement of the approximation guarantee.

Overall, we obtain a $(22+\varepsilon)$-approximation algorithm for ATSP for every fixed $\varepsilon > 0$. The algorithm computes a solution of cost at most $22+\varepsilon$ times the cost of an optimum solution to (ATSP LP) and hence the integrality ratio of (ATSP LP) is at most 22.

4.2 Reducing to vertebrate pairs

Let $\mathcal{L}_{\geq 2} := \{L \in \mathcal{L} : |L| \geq 2\}$ be the family of all non-singleton elements of $\mathcal{L}$. In this section we show how to reduce ATSP to the case where we have already a given subtour B, called *backbone*, that visits all elements of $\mathcal{L}_{\geq 2}$; see Figure 4.1. We call a strongly laminar ATSP instance together with a given backbone a *vertebrate pair*.

Definition 4.1. *A* vertebrate pair *consists of*

- *a strongly laminar ATSP instance* $\mathcal{I} = (G, \mathcal{L}, x, y)$ *and*
- *a connected Eulerian multi-subgraph* B *of* G *(the* backbone*) such that* $V(B) \cap L \neq \emptyset$ *for all* $L \in \mathcal{L}_{\geq 2}$.

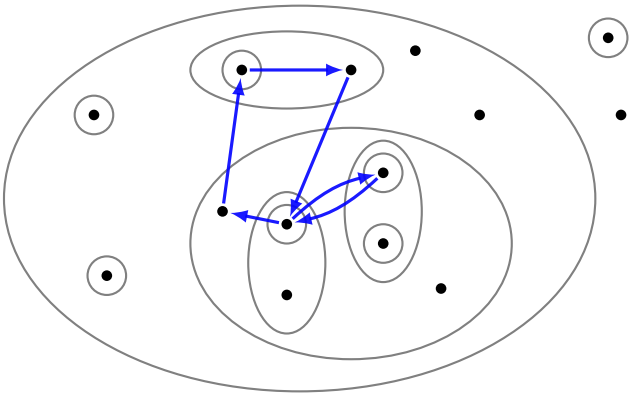

Figure 4.1: Example of a vertebrate pair. The laminar family $\mathcal{L}$ is shown in gray and a backbone B is shown in blue.

Let $\kappa, \eta \geq 0$. A (κ, η)-algorithm for vertebrate pairs *is an algorithm that computes, for any given vertebrate pair $(\mathcal{I}, B)$, a multi-set F of edges such that $E(B) \dot\cup F$ is a tour and*

$$c(F) \leq \kappa \cdot \mathrm{LP}(\mathcal{I}) + \eta \cdot \sum_{v \in V \setminus V(B) : \{v\} \in \mathcal{L}} 2y_{\{v\}}$$

Note that this definition is slightly different to the one in [STV18a] (where $G[L]$ was not required to be strongly connected for $L \in \mathcal{L}$), but this will not be relevant.

In this section we will show that a (κ, η)-algorithm for vertebrate pairs (for any constants κ and η) implies a $(3\kappa + \eta + 2)$-approximation algorithm for ATSP.

Let $(G, \mathcal{L}, x, y)$ be a strongly laminar ATSP instance and c the induced cost function. In the following we fix for every $u, v \in V$ a nice u-v-path $P_{u,v}$. Such paths can be computed in polynomial time by Lemma 3.6.

Lemma 4.2. *Let $W \in \mathcal{L} \cup \{V\}$ and let $u, v \in W$. Then*

$$c\left(E\left(P_{u,v}\right)\right) \leq \sum_{L \in \mathcal{L} : L \subsetneq W, L \cap V(P_{u,v}) \neq \emptyset} 2y_L - \sum_{L \in \mathcal{L} : u \in L \subsetneq W} y_L - \sum_{L \in \mathcal{L} : v \in L \subsetneq W} y_L.$$

Proof. Since the path $P_{u,v}$ is nice, it is contained in $G[W]$. Moreover, it leaves every set $L \in \mathcal{L}$ at most once and enters every set $L \in \mathcal{L}$ at most once. A set $L \in \mathcal{L}$ with $u \in L$ is never entered by $P_{u,v}$ and a set $L \in \mathcal{L}$ with $w \in L$ is never left by $P_{u,v}$. □

We define

$$\mathrm{value}(W) := \sum_{L \in \mathcal{L} : L \subsetneq W} 2y_L.$$

and

$$D_W(u, v) := \sum_{L \in \mathcal{L} : u \in L \subsetneq W} y_L + \sum_{L \in \mathcal{L} : v \in L \subsetneq W} y_L + c(E(P_{u,v}))$$

for $u, v \in W$. Note that $D_W(u, v) \leq \mathrm{value}(W)$ by Lemma 4.2. We write

$$D_W := \max\{D_W(u, v) : u, v \in W\}.$$

The intuitive meaning of D_W in the analysis of our reduction to vertebrate pairs is the following. On the one hand, it can be useful if D_W is small: if we enter the set W at some vertex $s \in W$ and leave it at some other vertex $t \in W$, we can always find a cheap s-t-walk inside $G[W]$. On the other hand, if D_W is large, we can find a nice path inside W that visits many sets $L \in \mathcal{L}$ (or more precisely, sets of high weight in the dual solution y).

The reduction to vertebrate pairs is via a recursive algorithm. For a given set $W \in \mathcal{L} \cup \{V\}$ it constructs a tour in $G[W]$. See Figure 4.2 for an illustration.

Algorithm 1: Recursive algorithm to reduce to vertebrate pairs.

Input: a strongly laminar ATSP instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ with $G = (V, E)$,
a set $W \in \mathcal{L} \cup \{V\}$, and
a (κ, η)-algorithm $\mathcal{A}$ for vertebrate pairs (for some constants $\kappa, \eta \geq 0$)
Output: a tour F in $G[W]$

1. If $W \neq V$, contract $V \setminus W$ into a single vertex $v_{\bar{W}}$ and redefine $y_W := \frac{D_W}{2}$.

2. **Construct a vertebrate pair:** Let $u^*, v^* \in W$ such that $D_W(u^*, v^*) = D_W$. Let B be the multi-graph corresponding to the closed walk that results from appending P_{u^*,v^*} and P_{v^*,u^*}.

   Let $\mathcal{L}_{\bar{B}}$ be the set of all maximal sets $L \in \mathcal{L}$ with $L \subsetneq W$ and $V(B) \cap L = \emptyset$. Contract every set $L \in \mathcal{L}_{\bar{B}}$ to a single vertex v_L and set $y_{\{v_L\}} := y_L + \frac{D_L}{2}$. Let G' be the resulting graph.

   Let $\mathcal{L}'$ be the laminar family of subsets of $V(G')$ that contains singletons $\{v_L\}$ for $L \in \mathcal{L}_{\bar{B}}$ and all the sets arising from $L \in \mathcal{L}$ with $L \subseteq W$ and $L \cap V(B) \neq \emptyset$.

   Let $\mathcal{I}' = (G', \mathcal{L}', x, y)$ be the resulting strongly laminar instance.

3. **Compute a solution for the vertebrate pair:** Apply the given algorithm $\mathcal{A}$ to the vertebrate pair $(\mathcal{I}', B)$. Let F' be the resulting Eulerian edge set.

4. **Lift the solution to a subtour:** Fix an Eulerian walk in every connected component of F'. Now uncontract every $L \in \mathcal{L}_{\bar{B}}$. Whenever an Eulerian walk passes through v_L, we get two edges $(u', u) \in \delta^-(L)$ and $(v, v') \in \delta^+(L)$. To connect u and v within L, add the path $P_{u,v}$.

   Moreover, if $W \neq V$ do the following. Whenever an Eulerian walk passes through $v_{\bar{W}}$ using the edges $(u, v_{\bar{W}})$ and $(v_{\bar{W}}, v)$, replace them by the path $P_{u,v}$.

5. **Recurse to complete to a tour of the original instance:** For every set $L \in \mathcal{L}_{\bar{B}}$, apply Algorithm 1 recursively to obtain a tour F_L in $G[L]$. Let F'' be the union of F' and all these tours F_L for $L \in \mathcal{L}_{\bar{B}}$.

6. Return $F := F'' \dot{\cup} E(B)$.

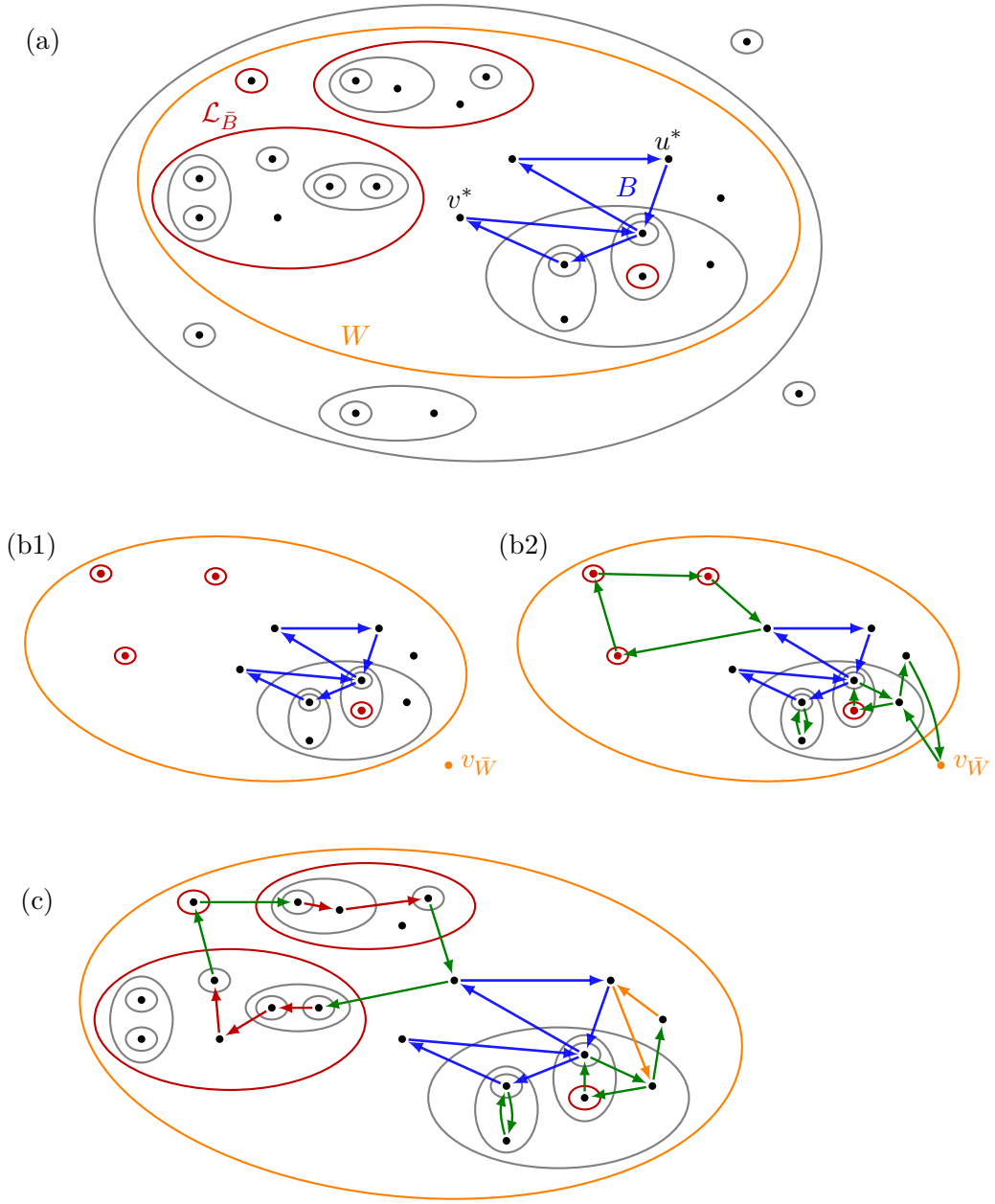

Figure 4.2: Illustration of Algorithm 1. The ellipses show the laminar family $\mathcal{L}$. Picture (a) shows the set W (orange), the subtour B (blue), and the elements of $\mathcal{L}_{\bar{B}}$ (red). The subtour B is the union of the paths P_{u^*,v^*} and P_{v^*,u^*}. Picture (b1) shows the resulting vertebrate pair instance as constructed in step 2 of Algorithm 1. The vertices resulting from the contraction of elements of $\mathcal{L}_{\bar{B}}$ are shown in red and the vertex $v_{\bar{W}}$ that results from the contraction of $V \setminus W$ is shown in orange. Picture (b2) shows in green a possible solution to this vertebrate pair. Picture (c) illustrates step 4 of Algorithm 1: the green edges are those that arise from the vertebrate pair solution from Picture (b2) by undoing the contraction of the sets in $\mathcal{L}_{\bar{B}}$. The red edges are the paths that we add to connect within $L \in \mathcal{L}_{\bar{B}}$ when uncontracting L. The orange edges show the u-v-path in $G[W]$ that we add to replace the edges $(u, v_{\bar{W}})$ and $(v_{\bar{W}}, v)$ in the vertebrate pair solution from Picture (b2).

First, we observe that Algorithm 1 indeed returns a tour in $G[W]$.

Lemma 4.3. *Let $\kappa, \eta \geq 1$. Suppose we have a polynomial-time (κ,η)-algorithm $\mathcal{A}$ for vertebrate pairs. Then Algorithm 1 has polynomial runtime and returns a tour in $G[W]$ for every strongly laminar ATSP instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ and every $W \in \mathcal{L} \cup \{V\}$.*

Proof. We apply induction on $|W|$. For $|W| = 1$, the algorithm returns $F = \emptyset$. Now let $|W| > 1$. At the end of step 3, we have that F' is Eulerian and $F' \dot\cup E(B)$ is a tour in the instance $\mathcal{I}'$. In step 4, the set F' remains Eulerian and $F' \dot\cup E(B)$ remains connected. Moreover, the subtour $F' \dot\cup E(B)$ visits all sets in $\mathcal{L}_{\bar{B}}$, i.e. we have $F' \cap \delta(L) \neq \emptyset$ for all $L \in \mathcal{L}_{\bar{B}}$. The subtour $F' \dot\cup E(B)$ also visits all vertices in W that are not contained in any set $L \in \mathcal{L}_{\bar{B}}$, i.e. for these vertices v we have $\delta(v) \cap (F' \cup E(B)) \neq \emptyset$. After step 4, we have $F' \subseteq E[W]$. We conclude that the graph $(W, F'' \dot\cup E(B))$ is connected and Eulerian; here we applied the induction hypothesis to the sets $L \in \mathcal{L}_{\bar{B}}$.

To see that the runtime of the algorithm is polynomially bounded we observe that by Lemma 2.6 there are in total at most $|\mathcal{L}| + 1 \leq 2|V|$ recursive calls of the algorithm because $\mathcal{L}$ is a laminar family. □

Next we observe that our backbone B visits many sets $L \in \mathcal{L}$ inside W if D_W is large.

Lemma 4.4. *Let $\mathcal{I} = (G, \mathcal{L}, x, y)$ be a strongly laminar ATSP instance, and let $W \in \mathcal{L} \cup \{V\}$. Moreover, let B be as in step 2 of Algorithm 1. Then*

$$\sum_{L \in \mathcal{L}_{\bar{B}}} (2y_L + \text{value}(L)) \leq \text{value}(W) - D_W. \tag{4.1}$$

Proof. By Lemma 4.2 and the choice of u^* and v^* we get

$$\begin{aligned}
\text{value}(W) - \sum_{L \in \mathcal{L}_{\bar{B}}} (2y_L + \text{value}(L)) &= \sum_{L \in \mathcal{L}:\, L \subsetneq W,\, L \cap V(B) \neq \emptyset} 2y_L \\
&\geq \sum_{L \in \mathcal{L}:\, L \subsetneq W,\, L \cap V(P_{u^*,v^*}) \neq \emptyset} 2y_L \\
&= c(E(P_{u^*,v^*})) + \sum_{L \in \mathcal{L}:\, u^* \in L \subsetneq W} y_U + \sum_{L \in \mathcal{L}:\, v^* \in L \subsetneq W} y_U \\
&= D_W(u^*, v^*) \\
&= D_W.
\end{aligned}$$

□

Now we analyze the cost of the tour F in $G[W]$ computed by Algorithm 1.

Lemma 4.5. *Let $\kappa, \eta \geq 0$. Suppose we have a (κ,η)-algorithm $\mathcal{A}$ for vertebrate pairs. Let $\mathcal{I} = (G, \mathcal{L}, x, y)$ be a strongly laminar ATSP instance, c the induced cost function, and $W \in \mathcal{L} \cup \{V\}$. Then the tour F in $G[W]$ returned by Algorithm 1 has cost at most*

$$c(F) \;\leq\; (2\kappa + 2) \cdot \text{value}(W) + (\kappa + \eta) \cdot (\text{value}(W) - D_W).$$

Proof. By induction on $|W|$. The statement is trivial for $|W| = 1$ since then $c(F) = 0$ (because $F \subseteq E[W] = \emptyset$). Let now $|W| \geq 2$. By definition of D_W, we have

$$c(E(B)) \;=\; c(P_{u^*,v^*}) + c(P_{v^*,u^*}) \;\leq\; 2D_W. \tag{4.2}$$

We now analyze the cost of F' in step 3 of Algorithm 1. Since F' is the output of a (κ, η)-algorithm applied to the vertebrate pair $(\mathcal{I}', B)$, we have $c(F') \leq \kappa \cdot \mathrm{LP}(\mathcal{I}') + \eta \cdot \sum_{L \in \mathcal{L}_{\bar{B}}} 2y_{\{v_L\}}$. Using $\sum_{L \in \mathcal{L}_{\bar{B}}} 2y_{\{v_L\}} = \sum_{L \in \mathcal{L}_{\bar{B}}} (2y_L + D_L)$ and

$$\mathrm{LP}(\mathcal{I}') \leq D_W + \sum_{L \in \mathcal{L}, L \subsetneq W, L \cap V(B) \neq \emptyset} 2y_L + \sum_{L \in \mathcal{L}_{\bar{B}}} 2y_{\{v_L\}},$$

this implies

$$c(F') \leq \kappa \cdot D_W + \sum_{L \in \mathcal{L}, L \subsetneq W, L \cap V(B) \neq \emptyset} \kappa \cdot 2y_L + \sum_{L \in \mathcal{L}_{\bar{B}}} (\kappa + \eta) \cdot (2y_L + D_L) \tag{4.3}$$

at the end of step 3. The lifting and all the amendments of F' in step 4 do not increase the cost of F' by Lemma 4.2 and the choice of the values $y_{\{v_L\}}$ in step 2 and y_W in step 1. (Here we use that whenever a Eulerian walk passes through $v_{\bar{W}}$, we leave and enter W.)

To bound the cost increase in step 5 we apply the induction hypothesis. Adding the edges resulting from a single recursive call of Algorithm 1 in step 5 for some $L \in \mathcal{L}_{\bar{B}}$ increases the cost by at most $c(F_L) \leq (2\kappa + 2) \cdot \mathrm{value}(L) + (\kappa + \eta)(\mathrm{value}(L) - D_L)$. Using (4.3), we obtain the following bound:

$$\begin{aligned}
c(F'') &\leq \kappa \cdot D_W + \sum_{L \in \mathcal{L}, L \subsetneq W, L \cap V(B) \neq \emptyset} \kappa \cdot 2y_L \\
&\quad + \sum_{L \in \mathcal{L}_{\bar{B}}} \big((2\kappa + 2) \cdot \mathrm{value}(L) + (\kappa + \eta) \cdot (2y_L + \mathrm{value}(L))\big) \\
&\leq \kappa \cdot D_W + \kappa \cdot \mathrm{value}(W) \\
&\quad + \sum_{L \in \mathcal{L}_{\bar{B}}} \big((\kappa + 2) \cdot \mathrm{value}(L) + (\kappa + \eta) \cdot (2y_L + \mathrm{value}(L))\big) \\
&\leq \kappa \cdot D_W + \kappa \cdot \mathrm{value}(W) \\
&\quad + (\kappa + 2) \cdot (\mathrm{value}(W) - D_W) + (\kappa + \eta) \cdot (\mathrm{value}(W) - D_W) \\
&= (2\kappa + 2) \cdot \mathrm{value}(W) - 2 \cdot D_W + (\kappa + \eta) \cdot (\mathrm{value}(W) - D_W),
\end{aligned}$$

where we used the definition of $\mathcal{L}_{\bar{B}}$ for the second inequality and Lemma 4.4 for the third inequality. Together with (4.2) this implies the claimed bound on $c(F)$. □

Now we prove the main result of this section.

Theorem 4.6. *Let $\kappa, \eta \geq 0$. Suppose we have a polynomial-time (κ, η)-algorithm for vertebrate pairs. Then there is a polynomial-time algorithm that computes a solution of cost at most*

$$3\kappa + \eta + 2$$

times the value of (ATSP LP) *for any given ATSP instance.*

Proof. By Theorem 3.8 it suffices to show that there is a polynomial-time algorithm that computes a solution of cost at most $(3\kappa + \eta + 2) \cdot \mathrm{LP}(\mathcal{I})$ for any given strongly laminar ATSP instance $\mathcal{I}$. Given such an instance, we apply Algorithm 1 to $W = V$.

By Lemma 4.3 and Lemma 4.5, this algorithm computes in polynomial time a tour of cost at most

$$\begin{aligned} c(F) &\leq (2\kappa+2)\cdot \text{value}(V) + (\kappa+\eta)\cdot(\text{value}(V) - D_V) \\ &= (2\kappa+2)\cdot \text{LP}(\mathcal{I}) + (\kappa+\eta)\cdot(\text{LP}(\mathcal{I}) - D_V) \\ &\leq (3\kappa+\eta+2)\cdot \text{LP}(\mathcal{I}). \end{aligned}$$ □

In the following we will present a $(2, 14+\varepsilon)$-algorithm for vertebrate pairs, improving on the $(2, 37+\varepsilon)$-algorithm by Svensson, Tarnawski, and Végh [STV18a]. Using their vertebrate pair algorithm, Theorem 4.6 immediately implies a $(45+\varepsilon)$-approximation algorithm for ATSP.

Remark 4.7
One could achieve a slightly better overall approximation ratio for ATSP by the following modifications. Change Algorithm 1 and generalize the notion of vertebrate pairs as follows. First, in the definition of a vertebrate pair allow that the backbone is not necessarily Eulerian but could also be an s-t-path for some $s, t \in V$. In this case the solution for the vertebrate pair would again be an Eulerian multi-set F of edges such that $(V, E(B) \dot\cup F)$ is connected; then $E(B) \dot\cup F$ is an s-t-tour. The algorithm for vertebrate pairs that we will describe in later sections extends to this more general version.

Then fix a constant $\delta \in [0, 1]$ depending on κ and η and change step 5 of Algorithm 1 as follows. If in step 4 we added a u-v-path $P_{u,v}$ in $G[L]$ with $D_L(u, v) \geq (1-\delta)\cdot D_L$ for a set $L \in \mathcal{L}_{\bar{B}}$, then we use this path as a backbone in the recursive call of Algorithm 1 instead of constructing a new backbone. This saves the cost $2D_L$ of the backbone in the recursive call, but we also pay some additional cost. Because the total y-weight of the sets in $\mathcal{L}$ visited by P is not D_L (as with the old choice of the backbone) but slightly less, we obtain a worse bound in Lemma 4.4. If for a set $L \in \mathcal{L}_{\bar{B}}$ we did not add a path in $G[L]$ of length at least $(1-\delta)\cdot D_L$ in step 4, then we do not change the recursive call of Algorithm 1 in step 5. In this case we gain because the bound on the cost of the edges that we added in step 4 is not tight.

Optimizing δ depending on κ and η leads to an improvement of the overall approximation ratio. However, the improvement is small. We will later show that there is a polynomial-time $(2, 14+\varepsilon)$-algorithm for vertebrate pairs for any fixed $\varepsilon > 0$. For $\kappa = 2$ and $\eta > 14$ the improvement is less than 0.2, and it is less than 1 for any κ and η.

4.3 Computing subtour covers

Very roughly, the algorithm that Svensson, Tarnawski, and Végh [STV19] for vertebrate pairs follows the cycle cover approach by Frieze, Galbiati and Maffioli [FGM82]. The algorithm by Frieze, Galbiati and Maffioli always maintains an Eulerian (multi-)set H of edges and repeatedly computes another Eulerian (multi-)set F of edges that enters and leaves every connected component of (V, H) at least once. Then it adds the edges of F to H and iterates until (V, H) is connected.

In order to achieve a constant approximation ratio, the algorithm for vertebrate pairs and its analysis are much more involved. The main algorithm is essentially due to Svensson [Sve15], and we describe an improved version of it in Section 4.4.

In this section we discuss a sub-routine called by Svensson's algorithm. The sub-routine we present here is an improved version of an algorithm by Svensson, Tarnawski, and Végh [STV18a]. It computes solutions to the *Subtour Cover* problem, which we define below. One can view the Subtour Cover problem as the analogue of the cycle cover problem that is solved in every iteration of the algorithm by Frieze, Galbiati and Maffioli. However, we do not only require that the multi edge set F that we compute is Eulerian and enters and leaves every connected component of (V, H), but require in addition that every component of (V, F) that crosses the boundary of a set $L \in \mathcal{L}_{\geq 2}$ is connected to the backbone B.

Definition 4.8. *An instance of* Subtour Cover *consists of a vertebrate pair $(\mathcal{I}, B)$ with $\mathcal{I} = (G, \mathcal{L}, x, y)$ and a multi-subset H of $E[V \setminus V(B)]$ such that*

- *(V, H) is Eulerian, and*
- *$H \cap \delta(L) = \emptyset$ for all $L \in \mathcal{L}_{\geq 2}$.*

A solution to such an instance $(\mathcal{I}, B, H)$ is a multi edge set F such that the following three conditions are fulfilled:

(i) *$(V \setminus V(B), F)$ is Eulerian.*

(ii) *$\delta(W) \cap F \neq \emptyset$ for all vertex sets W of connected components of $(V \setminus V(B), H)$.*

(iii) *If for a connected component D of (V, F) there is a set $L \in \mathcal{L}_{\geq 2}$ with $E(D) \cap \delta(L) \neq \emptyset$, then $V(D) \cap V(B) \neq \emptyset$.*

Subtour Cover is very similar to the notions of Subtour Partition Cover from [STV19] and Local Connectivity ATSP from [Sve15]. The difference between instances of Subtour Cover and Subtour Partition Cover is that we require that $H \cap \delta(L) = \emptyset$ for all $L \in \mathcal{L}_{\geq 2}$ in Definition 4.8. Moreover, a solution for Subtour Partition Cover is not required to fulfill condition (*iii*). However, the instances to which Svensson, Tarnawski, and Végh apply their algorithm for Subtour Partition Cover also fulfill the definition of Subtour Cover and the solutions computed by this algorithm also fulfill condition (*iii*). We include these properties explicitly in Definition 4.8 because we will exploit them for some improvement in Svensson's algorithm (see Section 4.4).

For the analysis of Svensson's algorithm for vertebrate pairs it is not sufficient to have only a bound on the total cost of a solution to Subtour Cover. In this section we explain an algorithm that computes solutions to Subtour Cover that fulfill certain "local" cost bounds. More precisely, the goal of this section is to show the following theorem, where we write $y_v := y_{\{v\}}$ if $\{v\} \in \mathcal{L}$ and $y_v := 0$ otherwise.

Theorem 4.9. *There is a polynomial-time algorithm for Subtour Cover that computes for every instance $(\mathcal{I}, B, H)$ a solution F such that*

$$c(F) \;\leq\; 2 \cdot \mathrm{LP}(\mathcal{I}) + \sum_{v \in V \setminus V(B)} 2y_v, \tag{4.4}$$

and for every connected component D of (V, F) with $V(D) \cap V(B) = \emptyset$ we have

$$c(E(D)) \;\leq\; 3 \cdot \sum_{v \in V(D)} 2y_v. \tag{4.5}$$

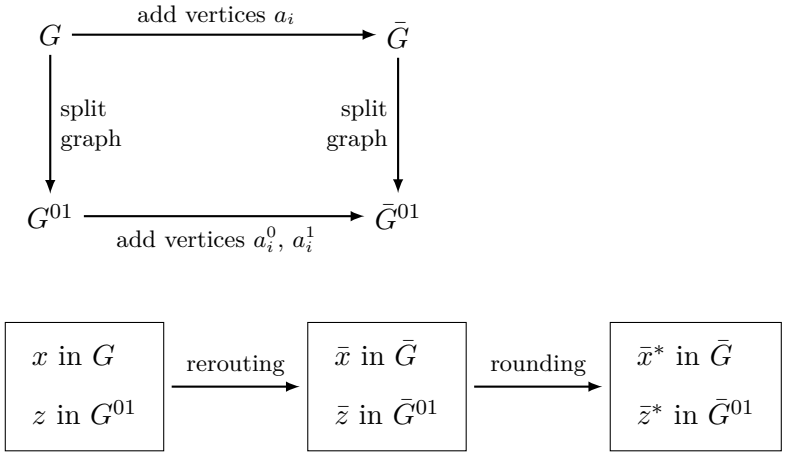

Figure 4.3: Overview of the different graphs and circulations occurring in the proof of Theorem 4.9. The integral circulation $\bar{x}^*$ corresponds to an Eulerian (multi-)edge set $\bar{F}$ in $\bar{G}$.

Svensson, Tarnawski, and Végh [STV18a] proved a similar statement, but instead of (4.5) they showed the weaker bound $c(E(D)) \leq 4 \cdot \sum_{v \in V(D)} 2y_v$.

The reason why we need bounds on the cost of single connected components rather than the total Subtour Cover solution is the following. When Svensson's algorithm computes a solution F to Subtour Cover, it does not include all edges of F in the tour that it computes but only those edges that are part of some carefully selected connected components of (V, F).

In the rest of this section we prove Theorem 4.9. We first give a brief outline.

4.3.1 Outline

Let $W_1, \ldots, W_k$ be the vertex sets of the connected components of $(V \setminus V(B), H)$. To find a solution F that fulfills the properties (i) and (ii) we would like to find an integral circulation x^* in G that satisfies $x^*(\delta(W_i)) \geq 2$ for $i = 1, \ldots, k$. Note that x is a fractional circulation with this property. However, if we include the constraints $x^*(\delta(W_i)) \geq 2$ in the linear program describing a minimum cost circulation problem, we will in general not obtain an integral optimum solution. Svensson [Sve15] suggested the following. We can introduce new vertices a_i for $i = 1, \ldots, k$ and reroute one unit of flow going through the set W_i through the new vertex a_i. Then we can add constraints $x^*(\delta^-(a_i)) = 1$ to our flow problem and maintain integrality. After solving the minimum cost circulation problem, we can map the one unit of flow through a_i back to some flow entering and leaving W_i (with some additional cost).

The bound (4.4) is obtained by minimizing the total cost of the circulation. The most difficult properties to achieve are (iii) and (4.5). If we have (iii), it is relatively easy to obtain a bound of a similar form as (4.5) (with some other constant): we can add constraints of the form $x^*(\delta^-(v)) \leq \lceil x(\delta^-(v)) \rceil$ to our minimum cost circulation problem. Because of (iii) and the definition of the induced cost function c, this implies a bound similar to (4.5).

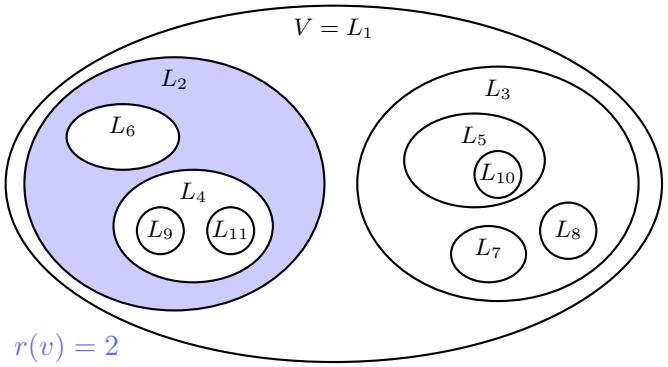

Figure 4.4: The laminar family $\mathcal{L} \cup \{V\} = \{L_1, \dots, L_{11}\}$. In this example, the set $L_2 \setminus (L_6 \cup L_4)$ is the set of all vertices v with $r(v) = 2$; it is shown in blue.

To achieve property (*iii*), Svensson, Tarnawski, and Végh [STV18a] introduced the concept of the *split graph*. This graph contains two copies of every vertex of the original graph G. Every Eulerian edge set in the split graph can be projected to an Eulerian edge set in the original graph G. The crucial property of the split graph is that every cycle that contains an edge corresponding to $e \in \delta_G(L)$ for some $L \in \mathcal{L}_{\geq 2}$ also contains a copy of a backbone vertex $v \in V(B)$. Therefore, if we round a circulation in the split graph (and then project the solution back to G), we will automatically fulfill property (*iii*).

While every circulation in the split graph can be projected to a circulation in the original graph G, we cannot lift any arbitrary circulation in G to a circulation in the split graph. However, Svensson, Tarnawski, and Végh [STV18a] showed that this is possible for every solution x to (ATSP LP). For this, they use a so-called *witness flow*. We will choose the witness-flow with a certain minimality condition to achieve the bound (4.5), improving on the Subtour Cover algorithm from [STV18a]. To obtain the improved bound we also choose the flow that is rerouted through the auxiliary vertices a_i more carefully.

Because we cannot lift an arbitrary circulation in G to a circulation in the split graph G^{01} of G, we proceed in the following order. First, we lift the circulation x to a circulation z in the split graph G^{01}. Then we add the auxiliary vertices a_i to G and add the two corresponding copies a_i^0 and a_i^1 to the split graph G^{01}. In the resulting split graph $\bar{G}^{01}$ we reroute flow through the new auxiliary vertices a_i^0, a_i^1 and round our fractional circulation to an integral one. See Figure 4.3.

We now explain our algorithm in detail.

4.3.2 The split graph

In this section we explain the concept of the *split graph* due to Svensson, Tarnawski, and Végh [STV18a]. This is an important tool for achieving property (*iii*) of a solution to Subtour Cover. This property will also be crucial in the proof of (4.5). For defining the split graph, we number the non-singleton elements of our laminar family $\mathcal{L}$ as follows. Number $\mathcal{L}_{\geq 2} \cup \{V\} = \{L_1, \dots, L_{r_{\max}}\}$ such that $|V| = |L_1| \geq \cdots \geq |L_{r_{\max}}| \geq 2$. Let $r(v) := \max\{i : v \in L_i\}$, and call an edge $e = (v, w) \in E$ *forward* if $r(v) < r(w)$, *backward* if $r(v) > r(w)$, and *neutral* if $r(v) = r(w)$. See Figure 4.4.

We will need the following simple observation.

Lemma 4.10. *Let C be the edge set of a cycle. If there exists a set $L \in \mathcal{L}_{\geq 2}$ with $C \cap \delta(L) \neq \emptyset$, then C contains a forward edge and a backward edge.*

Proof. Because C is Eulerian there exists an edge $e = (v, w) \in C \cap \delta^+(L)$. By the choice of the numbering $L_1, \ldots, L_{r_{\max}}$, we have $L_{r(v)} \subseteq L$ and hence $w \notin L_{r(v)}$. Therefore, the cycle with edge set C contains vertices v, w with $r(v) \neq r(w)$. Hence, C contains both a forward and a backward edge. □

Next we define the *split graph* G^{01} of G.

- For every vertex $v \in V$ it contains two vertices v^0 and v^1 (on the lower and upper level).
- For every $v \in V$ it contains an edge $e_v^{\downarrow} = (v^1, v^0)$ with $c(e_v^{\downarrow}) = 0$.
- For every $v \in V(B)$ it also contains an edge $e_v^{\uparrow} = (v^0, v^1)$ with $c(e_v^{\uparrow}) = 0$.
- For every forward edge $e = (v, w) \in E$, the split graph contains an edge $e^0 = (v^0, w^0)$ with $c(e^0) = c(e)$.
- For every backward edge $e = (v, w) \in E$, the split graph contains an edge $e^1 = (v^1, w^1)$ with $c(e^1) = c(e)$.
- For every neutral edge $e = (v, w) \in E$, the split graph contains edges $e^0 = (v^0, w^0)$ and $e^1 = (v^1, w^1)$ with $c(e^0) = c(e^1) = c(e)$.

We write $V^0 := \{v^0 : v \in V\}$ and call $G^{01}[V^0]$ the *lower level* of the split graph G^{01}. Similarly, we write $V^1 := \{v^1 : v \in V\}$ and call $G^{01}[V^1]$ the *upper level* of G^{01}. For a set $W \subseteq V$ let $W^{01} := \{v^j : v \in W, j \in \{0, 1\}\}$ be the vertex set of G^{01} that corresponds to W.

For any subgraph of G^{01} we obtain a subgraph of G (its *image*) by replacing both v^0 and v^1 by v and removing loops. Then, obviously, the image of a circuit is an Eulerian graph. The next lemma shows how we can use the split graph to achieve property *(iii)* of a solution to Subtour Cover.

Lemma 4.11. *If the image of a circuit in G^{01} contains an edge $e \in \delta(L)$ for some $L \in \mathcal{L}_{\geq 2}$, it also contains a vertex of B.*

Proof. Let C^{01} be a circuit in G^{01}, such that its image C (an Eulerian subgraph of G) contains an edge $e \in \delta(L)$ for some $L \in \mathcal{L}_{\geq 2}$. By Lemma 4.10, C contains a forward edge and a backward edge. Therefore C visits both levels of G^{01} and thus contains an edge $e_v^{\uparrow}$ for some $v \in V(B)$. □

4.3.3 Witness flows

We now want to map x to a circulation z in the split graph G^{01}. To this end, we define a flow $f \leq x$, which we will call a *witness flow*. In the construction of z, we will map the witness flow f to the lower level of G^{01} and map the remaining flow $x - f$ to the upper level of G^{01}. See Figure 4.5.

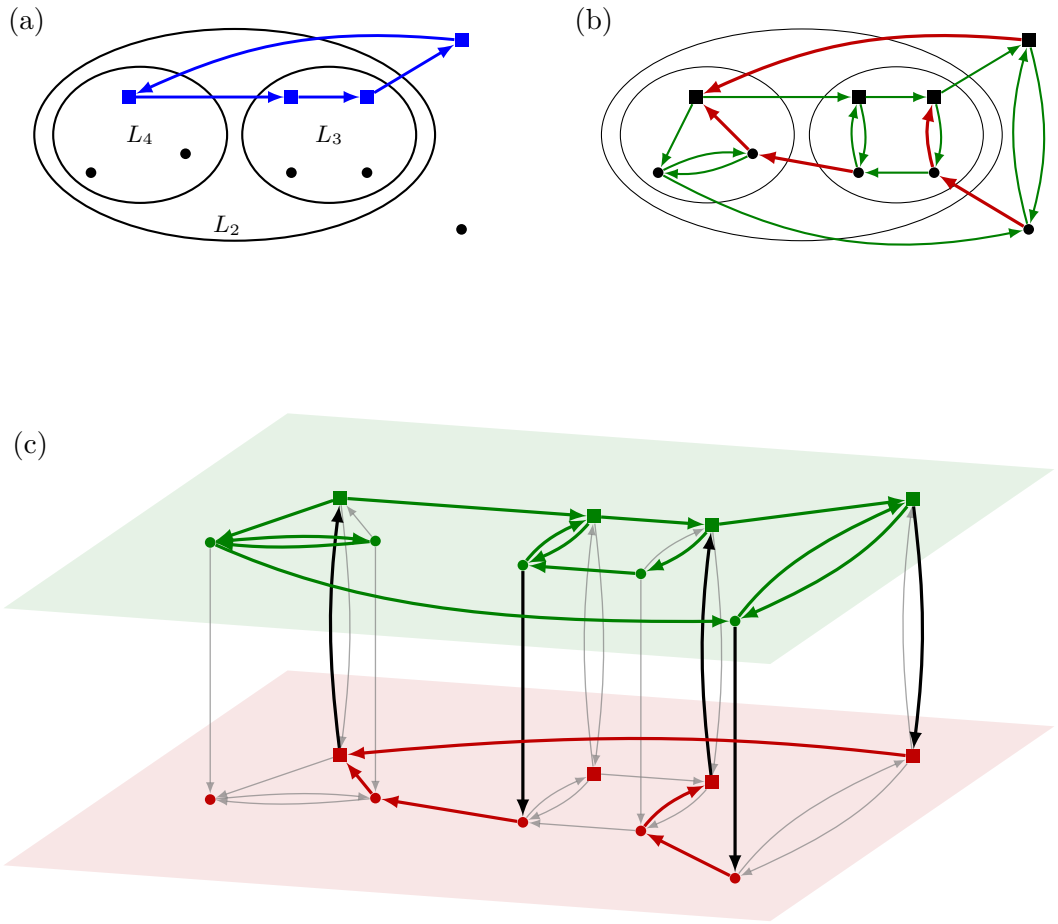

Figure 4.5: An example of the construction of the circulation z in G^{01}. Picture (a) shows the laminar family $\mathcal{L}_{\geq 2} = \{L_2, L_3, L_4\}$ and in blue the backbone B. Picture (b) shows a solution x to (ATSP LP) where we have $x_e = \frac{1}{2}$ for all edges; a witness flow f is shown in red. The vertices in $V(B)$ are shown as squares. Every cycle crossing the boundary of a set $L \in \mathcal{L}_{\geq 2}$ contains both a green and a red edge. Picture (c) shows the resulting circulation z in G^{01}, where we have $z_e > 0$ for every thick edge e and and $z_e = 0$ for all thin edges. The green vertices are those on the upper level of the split graph; the red vertices those on the lower level. The flow $x - f$ is mapped to the upper level (green) and the flow f is mapped to the lower level (red).

Definition 4.12 (witness flow)**.** *Let x' be a circulation in G. Then we call a flow f' in G a* witness flow (for x') *if*

(a) $f'(e) = 0$ *for every backward edge e;*

(b) $f'(e) = x'(e)$ *for every forward edge e;*

(c) $0 \leq f'(e) \leq x'(e)$ *for every neutral edge e; and*

(d) $f'(\delta^+(v)) \geq f'(\delta^-(v))$ *for all $v \in V \setminus V(B)$.*

The concept of witness flow was introduced in [STV19]. We now show that the pairs (x', f') where f' is a witness flow for the circulation x' in G, correspond to circulations in the split graph G^{01}.

Lemma 4.13. *Let z' be a circulation in G^{01}. Define $\pi(z') := (x', f')$ where x', f' are flows in G defined by*

- $x'(e) := z'(e^0) + z'(e^1)$*, and*
- $f'(e) := z'(e^0)$*,*

where we set $z'(e^1) := 0$ for forward edges e and $z'(e^0) := 0$ for backward edges e. Then x' is a circulation in G with $c(x') = c(z')$ and f' is a witness flow for x'.

Proof. (a) holds because for a backward edge e, the graph G^{01} does not contain an edge e^0. Similarly, (b) holds because for a forward edge e, the graph G^{01} does not contain an edge e^1. Property (c) is obvious by construction and (d) holds because for $v \in V \setminus V(B)$ the split graph does not contain an edge $e_v^{\uparrow}$. □

Having a circulation x' in G and a witness flow f' for x', we can map x' to a circulation z' in G^{01} with $\pi(z') = (x', f')$ as follows:

- For every edge e^0 of the lower level of G^{01} we set $z'(e^0) = f'(e)$.
- For every edge e^1 of the upper level of G^{01} we set $z'(e^1) = x'(e) - f'(e)$.
- For every edge $e_v^{\uparrow}$ (for $v \in V(B)$) we set $z(e_v^{\uparrow}) = \max\{0, f'(\delta^-(v)) - f'(\delta^+(v))\}$.
- For every edge $e_v^{\downarrow}$ (for $v \in V$) we set $z(e_v^{\downarrow}) = \max\{0, f'(\delta^+(v)) - f'(\delta^-(v))\}$.

Notice that $x'(e) = z'(e^0)$ for every forward edge e and $x'(e) = z'(e^1)$ for every backward edge e. Moreover, $x'(e) = z'(e^0)+z'(e^1)$ for every neutral edge e. Furthermore, z' indeed defines a circulation in G^{01} because $f'(\delta^+(v)) \geq f'(\delta^-(v))$ for all $v \in V \setminus V(B)$.

The following was already proved in [STV18a]. Here we give a simpler proof.

Lemma 4.14. *Let $(\mathcal{I}, B)$ be a vertebrate pair, with $\mathcal{I} = (G, \mathcal{L}, x, y)$. Then there exists a witness flow f for x.*

Proof. Consider G', which arises from G by adding a new vertex a and edges (a, v) for all $v \in V$ and edges (v, a) for all $v \in V(B)$. Set $l(e') = 0$ and $u(e') = \infty$ for the new edges. Moreover, for $e \in E$ set the lower bound $l(e)$ and the upper bound $u(e)$ according to the requirements from Definition 4.12, i.e. set $u(e) = x(e)$ if e is a forward

or neutral edge and $u(e) = 0$ otherwise and set $l(e) = x(e)$ if e is a forward edge and $l(e) = 0$ otherwise.

Then we are looking for a circulation f' in G' with $l \leq f' \leq u$. By Hoffman's circulation theorem (Theorem 2.20), this exists if

$$l(\delta^-(U)) \leq u(\delta^+(U)) \tag{4.6}$$

for all $U \subseteq V \cup \{a\}$. We show that this is indeed true. Suppose not, and let U be a minimal set violating (4.6). Since (4.6) obviously holds whenever $a \in U$ or $B \cap U \neq \emptyset$, we have $U \subseteq V \setminus V(B)$. Let i be the largest index so that $U \cap L_i \neq \emptyset$.

Case 1: $U \setminus L_i \neq \emptyset$.
Then (by the minimality of U) we have $l(\delta^-(U \cap L_i)) \leq u(\delta^+(U \cap L_i))$ and $l(\delta^-(U \setminus L_i)) \leq u(\delta^+(U \setminus L_i))$. Since all edges from $U \setminus L_i$ to $U \cap L_i$ are forward edges and all edges from $U \cap L_i$ to $U \setminus L_i$ are backward edges, we get

$$\begin{aligned} l(\delta^-(U)) + x(\delta^+(U \setminus L_i) \cap \delta^-(U \cap L_i)) &= l(\delta^-(U \cap L_i)) + l(\delta^-(U \setminus L_i)) \\ &\leq u(\delta^+(U \cap L_i)) + u(\delta^+(U \setminus L_i)) \\ &= u(\delta^+(U)) + x(\delta^+(U \setminus L_i) \cap \delta^-(U \cap L_i)) \end{aligned}$$

and hence (4.6).

Case 2: $U \subseteq L_i$.
Then $r(u) = i$ for all $u \in U$ and $r(w) \geq i$ for all $w \in L_i$. Hence $l(\delta^-(U)) \leq x(\delta^-(L_i) \cap \delta^-(U))$ because we have $l(e) > 0$ only for forward edges and all edges in $\delta^-(U) \setminus \delta^-(L_i)$ are neutral or backward edges. Moreover, edges in $\delta^+(U) \setminus \delta^+(L_i)$ are no backward edges, implying $x(\delta^+(U) \setminus \delta^+(L_i)) = u(\delta^+(U) \setminus \delta^+(L_i)) \leq u(\delta^+(U))$. Therefore,

$$\begin{aligned} l(\delta^-(U)) &\leq x(\delta^-(L_i) \cap \delta^-(U)) \\ &= x(\delta^-(L_i)) + x(\delta^+(U) \setminus \delta^+(L_i)) - x(\delta^-(L_i \setminus U)) \\ &\leq x(\delta^-(L_i)) + u(\delta^+(U)) - x(\delta^-(L_i \setminus U)). \end{aligned}$$

Since $L_i \setminus U \neq \emptyset$ (because $L_i \cap V(B) \neq \emptyset = U \cap V(B)$), we have $x(\delta^-(L_i \setminus U)) \geq 1$. Moreover, $L_i \in \mathcal{L} \cup \{V\}$ implies $x(\delta(L_i)) \in \{0, 2\}$ and hence $x(\delta^-(L_i)) \leq 1$. Hence (4.6) follows. □

Working with an arbitrary witness flow f is sufficient to obtain a constant-factor approximation for ATSP and this is essentially what Svensson, Tarnawski, and Végh did. However, to obtain a better approximation ratio we will not work with an arbitrary witness flow f, but will choose f with some additional properties. Recall that $W_1, \ldots, W_k$ are the vertex sets of the connected components of $(V \setminus V(B), H)$.

Lemma 4.15. *We can compute in polynomial time a witness flow f for x such that*

(e) the support of f is acyclic, and

(f) $\sum_{i=1}^k f(\delta(W_i)) \leq \sum_{i=1}^k f'(\delta(W_i))$ for every witness flow f' for x.

Proof. We first compute a witness flow $\tilde{f}$ by minimizing $\sum_{i=1}^k f(\delta(W_i))$ subject to the constraints $(a) - (d)$ from Definition 4.12. This linear program is feasible by Lemma 4.14. Then the flow $\tilde{f}$ fulfills property (f).

To compute the flow f we minimize $\sum_{e\in E} f(e)$ subject to the constraints $(a)-(d)$ and $f(e) \leq \tilde{f}(e)$ for all $e \in E$. This linear program is feasible because $\tilde{f}$ is a feasible solution. Then f is a witness flow for x with $\sum_{i=1}^{k} f(\delta(W_i)) \leq \sum_{i=1}^{k} \tilde{f}(\delta(W_i))$. Since the flow $\tilde{f}$ fulfills property (f), the same holds for the flow f.

Suppose f does not fulfill (e), i.e. f is not acyclic. Then there is a cycle $C \subseteq E$ with $f(e) > 0$ for all $e \in C$. As f fulfills (a), the set C does not contain any backward edge. This implies that C also contains no forward edge because C is a cycle. Let $\varepsilon := \min_{e\in C} f(e)$. For $e \in E$ we set $f'(e) := f(e) - \varepsilon \leq \tilde{f}(e)$ if $e \in C$ and $f'(e) := f(e) \leq \tilde{f}(e)$ otherwise. Because C contains neither forward nor backward edges, f' fulfills (a) and (b). By the choice of ε, we have $f'(e) \geq 0$ for all $e \in E$, implying (c). Finally, $f'(\delta^+(v)) - f'(\delta^-(v)) = f(\delta^+(v)) - f(\delta^-(v)) \geq 0$ for all $v \in V \setminus V(B)$, where we used that C is a cycle and f fulfills (d). This shows that f' is a witness flow and $f'(e) \leq \tilde{f}(e)$ for all $e \in E$, but $\sum_{e\in E} f'_e < \sum_{e\in E} f_e$, a contradiction to the choice of f. □

4.3.4 Rerouting and rounding

Recall that the sets $W_1, \ldots, W_k$ are the vertex sets of the connected components of $(V \setminus V(B), H)$. Thus they are pairwise disjoint subsets of $V \setminus V(B)$.

Lemma 4.16. *Let $i \in \{1, \ldots, k\}$ and $v, w \in W_i$. Then $r(v) = r(w)$.*

Proof. For all $L \in \mathcal{L}_{\geq 2}$ we have $H \cap \delta(L) = \emptyset$ and therefore $W_i \subseteq L$ or $W_i \cap L = \emptyset$. This implies $r(v) = \max\{j : v \in L_j\} = \max\{j : w \in L_j\} = r(w)$. □

We will now work with a flow f as in Lemma 4.15. Let G_f denote the residual graph of the flow f and the graph G with edge capacities x. So for every edge $e = (v, w) \in E$ with $f(e) < x(e)$, the residual graph contains an edge (v, w) with residual capacity $u_f((v, w)) = x(e) - f(e)$. For every edge $e = (v, w) \in E$ with $f(e) > 0$ the residual graph contains an edge (w, v) with residual capacity $u_f((w, v)) = f(e)$. Parallel edges can arise.

We will transform the graph G into another graph $\bar{G}$. The circulation z in G^{01} will be transformed into a circulation $\bar{z}$ in the split graph $\bar{G}^{01}$ of $\bar{G}$. We construct $\bar{G}$ from G by doing the following for $i = 1, \ldots, k$.

We add an auxiliary vertex a_i to G and set $r(a_i) := r(v)$ for $v \in W_i$; this is well-defined by Lemma 4.16. Let $\hat{W}_i$ be the vertex set of the first strongly connected component of $G_f[W_i]$ in some topological order. For every edge $(v, w) \in \delta^-(\hat{W}_i)$ we add an edge (v, a_i) of the same cost. Similarly, for every edge $(v, w) \in \delta^+(\hat{W}_i)$ we add an edge (a_i, w) of the same cost. Note that then a new edge is a forward/backward/neutral edge if and only if its corresponding edge in G is forward/backward/neutral. Then the split graph $\bar{G}^{01}$ of $\bar{G}$ contains new vertices a_i^0 and a_i^1, connected by an edge $e^{\downarrow}_{a_i} = (a_i^1, a_i^0)$ of cost zero. Let $\bar{G}$ the graph resulting from G by the modifications described above and let $\bar{G}^{01}$ be its split graph.

We will now reroute some of the flow z going through $\hat{W}_i$ such that it goes through one of the new vertices a_i^0, a_i^1. See Figure 4.6.

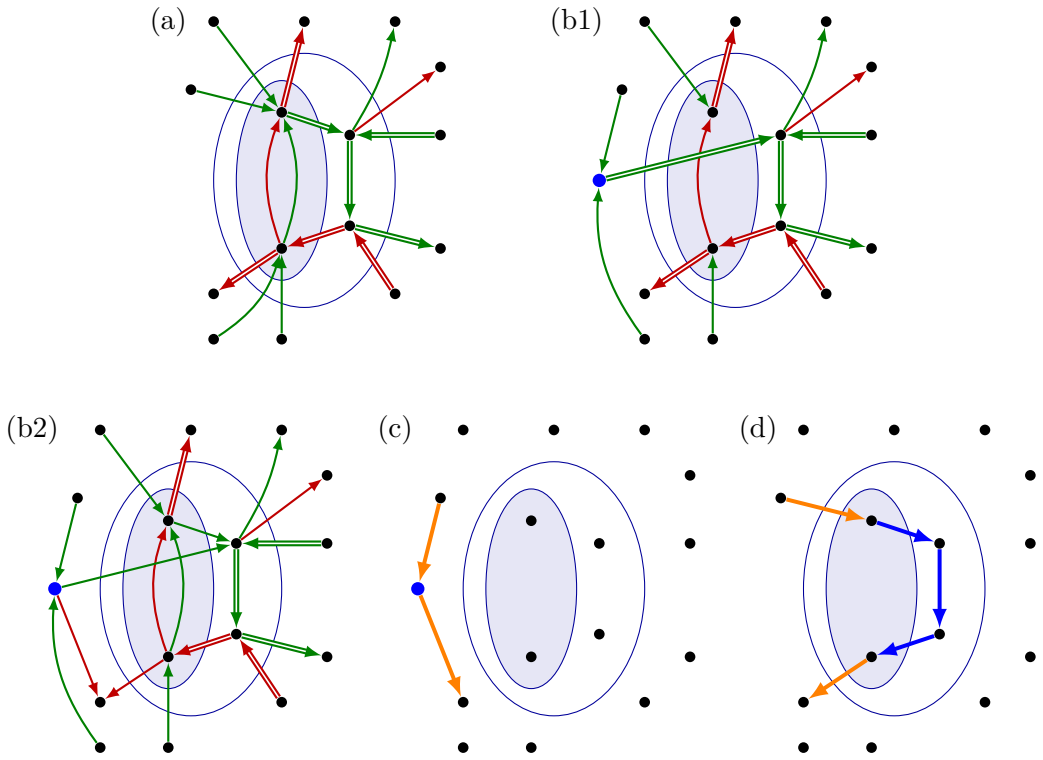

Figure 4.6: Example of the construction of the solution F from the witness flow f. On all pictures, a set W_i (blue with white interior) and the subset $\hat{W}_i$ (blue and filled) is shown. The pictures show only edges with at least one endpoint in W_i. Picture (a) shows (parts of) a possible solution x to (ATSP LP) (green and red) and a witness flow f (red). The edges drawn with a single line have value $\frac{1}{4}$, the edges drawn with a double line have value $\frac{1}{2}$. Pictures (b1) and (b2) show a possible flow $\bar{x}$ in $\bar{G}$ resulting from rerouting of flow through a_i (blue); the witness flow $\bar{f}$ is shown in red. Picture (c) shows in orange an possible integral flow $\bar{x}^*$ in $\bar{G}$ that could result if we rerouted flow through a_i as in (b2); The orange edges are elements of the edge set $\bar{F}$ with $\chi^{\bar{F}} = \bar{x}^*$. Picture (d) shows the result of mapping $\bar{F}$ back to G. In blue the path P_i in $G[W_i]$ is shown; it completes the orange edges to a circulation.

We need the following lemma.

Lemma 4.17. *Let G' be a directed graph graph and z' a circulation in G'. Let $U \subseteq V(G')$ with $z'(\delta(U)) \geq 2$. Then we can compute in polynomial time a multi-set $\mathcal{P}$ of paths in $G'[U]$ and for every $P \in \mathcal{P}$ starting in $s \in U$ and ending in $t \in U$*

- *a weight $\lambda_P > 0$,*
- *an edge $e_P^{\mathsf{in}} = (s', s) \in \delta^-(U)$, and*
- *an edge $e_P^{\mathsf{out}} = (t, t') \in \delta^+(U)$,*

such that $\sum_{P \in \mathcal{P}} \lambda_P = 1$ and

$$\sum_{P \in \mathcal{P}} \lambda_P \cdot \left(\chi^{e_P^{in}} + \chi^{E(P)} + \chi^{e_P^{out}}\right) \leq z'.$$

Proof. We contract $V(G') \setminus U$ to a vertex v_{outside}. Then $z'(\delta(v_{\mathsf{outside}})) = z'(\delta(U)) \geq 2$. Because z' remains a circulation, by Theorem 2.22 we can compute in polynomial time a set $\mathcal{C}$ of cycles containing v_{outside} and weights $\lambda_C > 0$ for $C \in \mathcal{C}$ with $\sum_{C \in \mathcal{C}} \lambda_C = 1$ such that

$$\sum_{C \in \mathcal{C}} \lambda_C \cdot \chi^E(C) \leq z'.$$

After undoing the contraction, every cycle C results in an edge $e^{\mathsf{in}} = (s', s) \in \delta^-(U)$, an edge $e^{\mathsf{out}} = (t, t') \in \delta^+(U)$, and an s-t-path P in $G'[U]$. □

We construct a circulation $\bar{z}$ in $\bar{G}^{01}$ from z by doing the following for $i = 1, \ldots, k$. We apply Lemma 4.17 to the vertex set $U = \hat{W}_i^{01}$. We partition the resulting set $\mathcal{P}$ into sets $\mathcal{P}^0$ and $\mathcal{P}^1$ such that $\mathcal{P}^0$ contains the paths $P \in \mathcal{P}$ for which e_P^{in} is contained in the lower level of the split graph and $\mathcal{P}^1$ contains the paths $P \in \mathcal{P}$ for which e_P^{in} is contained in the upper level of the split graph. Since $\sum_{P \in \mathcal{P}} \lambda_P = 1$, we have $\sum_{P \in \mathcal{P}^q} \lambda_P \geq \frac{1}{2}$ for some $q \in \{0, 1\}$. We can choose values $0 \leq \lambda'_P \leq \lambda_P$ such that $\sum_{P \in \mathcal{P}^q} \lambda'_P = \frac{1}{2}$. We now show how to construct $\bar{z}$ from z. For every $P \in \mathcal{P}^q$ we do the following:

- We decrease the flow on e_P^{in} and increase the flow on its corresponding edge in $\delta^-(a_i^q)$ by λ'_P.
- We decrease the flow on every edge $e \in E(P)$ by λ'_P.
- Let $p = 0$ if e_P^{out} is contained in the lower level of the split graph and $p = 1$ otherwise. We decrease the flow on e_P^{out} and increase the flow on its corresponding edge in $\delta^+(a_i^p)$ by λ'_P.
- Because $W_i \cap V(B) = \emptyset$, the path P contains no edge from the lower to the upper level; hence $p \leq q$. If $p < q$, i.e. $q = 1$ and $p = 0$, we increase the flow on $e_{a_i}^{\downarrow}$ by λ'_P.

Note that we maintain a circulation in the split graph $\bar{G}^{01}$.

Let $\bar{z}$ be the circulation in $\bar{G}^{01}$ resulting from z. Note that $c(\bar{z}) \leq c(z)$. Moreover, $\bar{z}$ is a circulation such that for every $i \in \{1, \ldots, k\}$ we have $\bar{z}(\delta^-(a_i^0)) = \frac{1}{2}$ or $\bar{z}(\delta^-(a_i^1)) = \frac{1}{2}$. Because we could only reroute $\frac{1}{2}$ unit of flow through a_i^0 or a_i^1, we consider the circulation $2\bar{z}$.

We round $2\bar{z}$ to an integral circulation: by Theorem 2.24 there is an integral circulation $\bar{z}^*$ in $\bar{G}^{01}$ with

(A) $0 \leq \bar{z}^*(e) \leq \lceil 2\bar{z}(e) \rceil$ for all $e \in E(\bar{G}^{01})$,

(B) $c(\bar{z}^*) \leq c(2\bar{z})$,

(C) $\bar{z}^*(\delta^-(v^1)) \leq \lceil 2\bar{z}(\delta^-(v^1)) \rceil$ for all $v \in V$, and

(D) for every $i \in \{1, \ldots, k\}$ we have $\bar{z}^*(\delta^-(a_i^0)) = 1$ or $\bar{z}^*(\delta^-(a_i^1)) = 1$.

Let $(\bar{x}, \bar{f}) := \pi(\bar{z})$ and $(\bar{x}^*, \bar{f}^*) := \pi(\bar{z}^*)$. Let $\bar{F} \subseteq E(\bar{G})$ be the multi-set of edges with $\chi^{\bar{F}} = \bar{x}^*$; see Figure 4.6 (c). Then $\bar{F}$ is Eulerian because $\bar{x}^*$ is a circulation.

We now show several properties of $\bar{F}$, before we show how to map $\bar{F}$ to a solution F for Subtour Cover in G (in Section 4.3.5). First we observe

$$c(\bar{F}) \;=\; c(\bar{x}^*) \;=\; c(\bar{z}^*) \;\leq\; 2 \cdot c(\bar{z}) \;\leq\; 2 \cdot c(z) \;=\; 2 \cdot c(x) \;=\; 2 \cdot \mathrm{LP}(\mathcal{I}). \tag{4.7}$$

The following lemma will be used in the proof of property (ii).

Lemma 4.18. *Let $i \in \{1, \ldots, k\}$. Then $|\delta^-_{\bar{F}}(a_i)| = 1$.*

Proof. We have

$$|\delta^-_{\bar{F}}(a_i)| = \bar{x}^*(\delta^-(a_i)) = \bar{z}^*(\delta^-(a_i^1)) + \bar{z}^*(\delta^-(a_i^0) \setminus \{e^{\downarrow}_{a_i}\})$$

By property (D), we have $\bar{z}^*(\delta^-(a_i^0)) = 1$ or $\bar{z}^*(\delta^-(a_i^1)) = 1$. Moreover, by property (A), the support of the integral flow $\bar{z}^*$ is contained in the support of the flow $\bar{z}$. If we have $\bar{z}^*(\delta^-(a_i^1)) = 1$, then we have by construction of $\bar{z}$ that $\bar{z}^*(e) \leq \lceil 2\bar{z}(e) \rceil = 0$ for all $e \in \delta^-(a_i^0) \setminus \{e^{\downarrow}_{a_i}\})$, implying $|\delta^-_{\bar{F}}(a_i)| = 1$. Otherwise, we have $\bar{z}^*(\delta^-(a_i^0)) = 1$ and by construction of $\bar{z}$ we have $\bar{z}(\delta^-(a_i^1)) = 0$ and $\bar{z}(e^{\downarrow}_{a_i}) = 0$. Therefore, by property (A) we have $\bar{z}^*(\delta^-(a_i^1)) = 0$ and $\bar{z}^*(e^{\downarrow}_{a_i}) = 0$. □

The proof of the following lemma is where we use our choice of f as in Lemma 4.15. Here, an arbitrary witness flow is not sufficient. See Figure 4.7 (a) – (b) for an illustration.

Lemma 4.19. *The flows $\bar{f}$ and $\bar{f}^*$ have acyclic support.*

Proof. Since the support of $\bar{f}^*$ is contained in the support of $\bar{f}$ by (A), it suffices to show that $\bar{f}$ has acyclic support. Suppose the support of $\bar{f}$ contains a cycle $\bar{C}$. Then there exists $i \in \{1, \ldots, k\}$ such that $a_i \in V(\bar{C})$ because otherwise $\bar{C}$ is contained in the support of f (which is acyclic). Let $\bar{e} = (a_i, v) \in E(\bar{C})$ and let $e = (u, v) \in \delta^+(\hat{W}_i)$ be the edge of G corresponding to $\bar{e}$. Then $f(e) > 0$ and hence the residual graph G_f contains an edge $(v, u) \in \delta^-_{G_f}(\hat{W}_i)$. Therefore $v \notin W_i$ since $\hat{W}_i$ is the vertex set of the first strongly connected component of $G_f[W_i]$. This shows $E(\bar{C}) \cap \delta(W_i \cup \{a_i\}) \neq \emptyset$.

We claim that we can map $\bar{C}$ to a closed walk C in the residual graph G_f. See Figure 4.7 (b) – (c). We first map every edge of the cycle $\bar{C}$ to its corresponding edge in G. Notice that the resulting edge set F is not necessarily a cycle: if $a_i \in V(\bar{C})$ for some $i \in \{1, \ldots, k\}$, then F contains an edge entering $\hat{W}_i$ and an edge leaving $\hat{W}_i$, but might be disconnected in between.

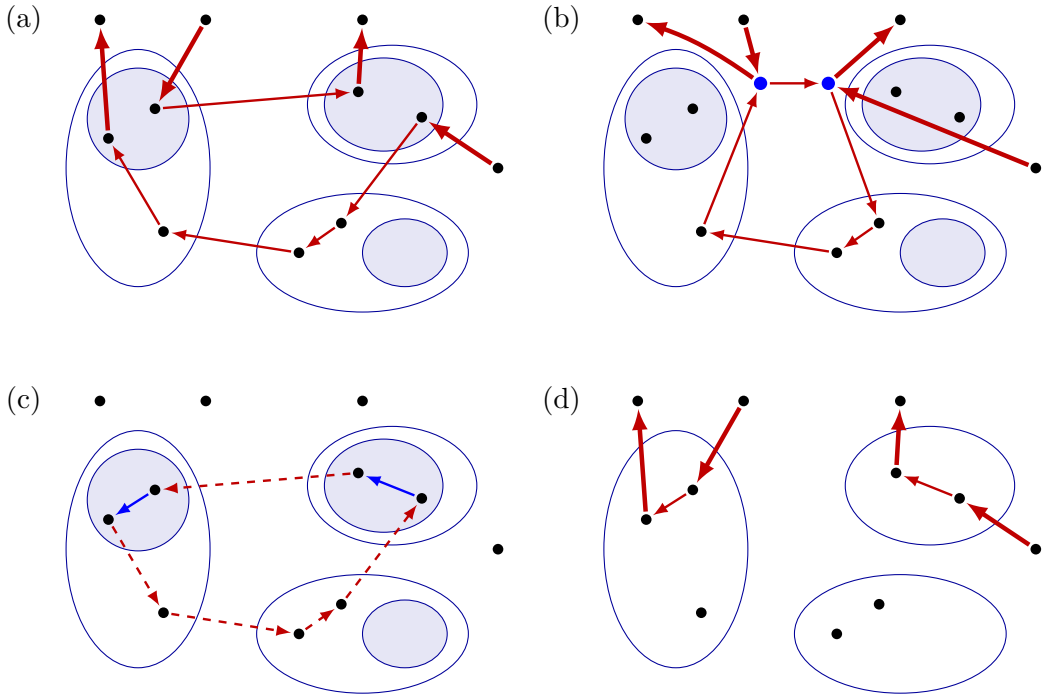

Figure 4.7: Illustration of the proof of Lemma 4.19 and the reason why choosing an arbitrary flow f as in Lemma 4.14 is not sufficient. The three sets W_i are shown in blue with white interior; pictures (a)–(c) also show their subsets $\hat{W}_i$ (blue and filled). Picture (a) shows (parts of) a flow f as in Lemma 4.14 (red); the thick edges show forward edges. This flow f will not be chosen by our algorithm; it does not minimize $\sum_{i=1}^{k} f(\delta(W_i))$. Picture (b) shows what would happen if we chose this flow anyway. We see a possible result of rerouting this flow through the vertices $a_i \in V(\bar{G})$ (shown in blue). In this example, the support of $\bar{f}$ contains a cycle $\bar{C}$. Picture (c) shows a corresponding closed walk C in the residual graph G_f. The blue edges show paths inside the sets $\hat{W}_i$; these exist because $G[\hat{W}_i]$ is strongly connected. Picture (d) shows the flow resulting from f by augmenting along C. The augmentation decreased $\sum_{i=1}^{k} f(\delta(W_i))$, but did not change the flow on forward edges.

We have $f(e) > 0$ for every edge $e \in F$. Thus, by reversing all edges in F we obtain edges in G_f (with positive residual capacity u_f). Moreover, we can complete this edge set to a closed walk C in G_f (with positive residual capacity u_f) by adding only edges of $G_f[\hat{W}_i]$ for $i \in \{1, \dots, k\}$; this is possible because for every $i \in \{1, \dots, k\}$, the subgraph $G_f[\hat{W}_i]$ is strongly connected by the choice of $\hat{W}_i$. We found a closed walk C in G_f. Let $i \in \{1, \dots, k\}$ such that $E(\bar{C}) \cap \delta(W_i \cup \{a_i\}) \neq \emptyset$. Then $E(C) \cap \delta(W_i) \neq \emptyset$.

Also note that $r(v) \geq r(w)$ for all $(v, w) \in E(C)$: every edge $(v, w) \in E(G_f)$ of C has a corresponding edge $(w, v) \in E(G)$ with $f(e) > 0$ or it has both endpoints in the same set $\hat{W}_i \subseteq W_i$. In the first case, we can conclude that (w, v) is not a backward edge and hence $r(w) \leq r(v)$. In the latter case, $r(v) = r(w)$ by Lemma 4.16. Since C is a closed walk we conclude that $r(v) = r(w)$ for all $v, w \in V(C)$.

This shows that augmenting f along the closed walk C changes flow only on neutral edges. We augment by some sufficiently small but positive amount and maintain a witness flow. We claim that this augmentation decreases $\sum_{i=1}^{k} f(\delta(W_i))$, which is a contradiction to our choice of f. See Figure 4.7 (d). The only edges of C contained in a cut $\delta(W_i)$ for some $i \in \{1, \dots, k\}$ result from mapping the edges of the cycle $\bar{C}$ in $\bar{G}$ to G_f and reversing them; for these edges the augmentation decreases the flow value. The other edges that we added to C are contained in some $G_f[\hat{W}_i]$ for $i \in \{1, \dots, k\}$ and hence they do not cross the boundary of any set W_i. Therefore, augmenting f along C decreases the flow value on all edges in $E(C) \cap (\delta(W_1) \cup \dots \cup \delta(W_k))$ and we have already shown that this set is nonempty. □

Lemma 4.20. *Let $\bar{D}$ be a connected component of $(V, \bar{F})$ with $V(\bar{D}) \cap V(B) = \emptyset$. Then $\bar{f}^*(E(\bar{D})) = 0$.*

Proof. Because $\bar{f}^*$ is a witness flow, we have $\bar{f}^*(\delta^-(v)) \leq \bar{f}^*(\delta^+(v))$ for every $v \in V(\bar{D})$. Since

$$\bar{f}^*(E(\bar{D})) \;=\; \sum_{v \in V(\bar{D})} \bar{f}^*(\delta^-(v)) \;\leq\; \sum_{v \in V(\bar{D})} \bar{f}^*(\delta^+(v)) \;=\; \bar{f}^*(E(\bar{D})),$$

we have $\bar{f}^*(\delta^-(v)) = \bar{f}^*(\delta^+(v))$ for every $v \in V(\bar{D})$. In other words, $\bar{f}^*$ restricted to $E(\bar{D})$ is a circulation. Because the support of $\bar{f}^*$ is acyclic by Lemma 4.19, this implies $\bar{f}^*(E(\bar{D})) = 0$. □

Lemma 4.21. *Let $i \in \{1, \dots, k\}$. Then $\bar{F} \cap \delta(W_i \cup \{a_i\}) \neq \emptyset$.*

Proof. By Lemma 4.18 there exists an edge $\bar{e} = (v, a_i) \in \bar{F}$. If $v \notin W_i$, we have $\bar{e} \in \bar{F} \cap \delta(W_i \cup \{a_i\}))$. Otherwise, the edge e of G that corresponds to $\bar{e}$ fulfills $e \in E[W_i] \cap \delta^-(\hat{W}_i)$. Therefore, we have $f(e) = x(e)$ as otherwise also the residual graph G_f contained e, contradicting the choice of $\hat{W}_i$. This implies

$$\bar{z}^*(\bar{e}^1) \;\leq\; \lceil 2\bar{z}(\bar{e}^1) \rceil \;\leq\; \lceil 2z(e^1) \rceil \;=\; \lceil 2(x(e) - f(e)) \rceil \;=\; 0.$$

But then

$$\bar{f}^*(\bar{e}) \;=\; \bar{z}^*(\bar{e}^0) \;=\; \bar{z}^*(\bar{e}^0) + \bar{z}^*(\bar{e}^1) \;=\; \bar{x}^*(\bar{e}) \;\geq\; 1,$$

because $\bar{e} \in \bar{F}$. By Lemma 4.20, this implies that the connected component $\bar{D}$ of $(V(\bar{G}), \bar{F})$ that contains a_i, also contains a vertex $w \in V(B)$. Since $W_i \cap V(B) = \emptyset$, this completes the proof. □

In the proof of property (*iii*) we will use the following observation.

Lemma 4.22. *Let $\bar{D}$ be a connected component of $(V, \bar{F})$ with $V(\bar{D}) \cap V(B) = \emptyset$. Then $E(\bar{D})$ contains no forward edge.*

Proof. By Lemma 4.20 we have $\bar{f}^*(E(\bar{D})) = 0$. Since $\bar{f}^*$ is a witness flow for $\bar{x}^* = \chi^{\bar{F}}$, this implies that $E(\bar{D})$ contains no forward edge. □

The following lemma will be used in the proof of (4.5).

Lemma 4.23. *Let $\bar{D}$ be a connected component of $(V, \bar{F})$ with $V(\bar{D}) \cap V(B) = \emptyset$. Then for every vertex $v \in V(\bar{D}) \setminus \{a_1, \dots, a_k\}$ with $y_v > 0$ we have $|\delta^-_{\bar{F}}(v)| \leq 2$.*

Proof. For every vertex $v \in V(\bar{D})$ we have

$$\begin{aligned} |\delta^-_{\bar{F}}(v)| &= \bar{x}^*(\delta^-(v)) = \bar{z}^*(\delta^-(v^1)) + \bar{z}^*(\delta^-(v^0) \setminus \{e_v^\downarrow\}) \\ &= \bar{z}^*(\delta^-(v^1)) + \bar{f}^*(\delta^-(v)) \\ &= \bar{z}^*(\delta^-(v^1)), \end{aligned}$$

where we used Lemma 4.20. For all $v \in V(\bar{D}) \setminus \{a_1, \dots, a_k\}$ with $y_v > 0$ we have $\{v\} \in \mathcal{L}$ and hence $x(\delta^-(v)) = 1$. Therefore, by (C) we get

$$|\delta^-_{\bar{F}}(v)| = \bar{z}^*(\delta^-(v^1)) \leq \lceil 2\bar{z}(\delta^-(v^1)) \rceil \leq \lceil 2x(\delta^-(v)) \rceil \leq 2.$$

□

4.3.5 Mapping back to G

We now transform $\bar{F}$ into a solution F of the Subtour Cover problem in G. See Figure 4.6 (c)–(d). By Lemma 4.18, every vertex a_i for $i \in \{1, \dots, k\}$ has exactly one incoming edge in $\bar{F}$ and because $\bar{F}$ is Eulerian, a_i also has exactly one outgoing edge. We replace all the edges in $\delta_{\bar{F}}(a_i)$ for $i \in \{1, \dots, k\}$ by their corresponding edges in G. For every $i \in \{1, \dots, k\}$ we added one edge $(v, s) \in \delta^-(\hat{W}_i)$ and an edge $(t, w) \in \delta^+(\hat{W}_i)$; to obtain an Eulerian edge set we add an s-t-path P_i in $G[W_i]$. Such a path exists because $G[W_i]$ is strongly connected. Let F be the resulting Eulerian multi-set of edges in G. Note that if two vertices $a, b \in V(G)$ are in the same connected component of $(V(\bar{G}), \bar{F})$, then they are also in the same connected component of $(V(G), F)$.

Lemma 4.24. *Let $i \in \{1, \dots, k\}$ and $L \in \mathcal{L}_{\geq 2}$. Then $E(P_i) \cap \delta(L) = \emptyset$.*

Proof. We have $H \cap \delta(L) = \emptyset$ and the sets $W_1, \dots, W_k$ are the vertex sets of the connected components of $(V \setminus V(B), H)$. Now $E(P_i) \subseteq E[W_i]$ implies $E(P_i) \cap \delta(L) = \emptyset$. □

We claim that F is a solution to the Subtour Cover problem and fulfills (4.4) and (4.5). Property (*i*) of a solution of the Subtour Cover problem holds because F is Eulerian. Property (*ii*) follows from Lemma 4.21.

We now show property (*iii*). Let D be a connected component of (V, F) with $E(D) \cap \delta(L) \neq \emptyset$ for some $L \in \mathcal{L}_{\geq 2}$. Because D is Eulerian it then contains a cycle C with $E(C) \cap \delta(L) \neq \emptyset$. But then $E(C) \subseteq E(D)$ contains a forward edge by Lemma 4.10. By Lemma 4.16, the edges of the paths P_i (for $i \in \{1, \dots, k\}$) are neutral edges. Hence

Lemma 4.22 implies $V(D) \cap V(B) \neq \emptyset$. This shows that F is a solution to the Subtour Cover problem. It remains to show (4.4) and (4.5).

Lemma 4.24 implies

$$c(E(P_i)) = \sum_{v \in V(P_i)} |E(P_i) \cap \delta(v)| \cdot y_v \leq \sum_{v \in W_i} 2y_v.$$

Moreover, the sets W_i for $i \in \{1, \dots, k\}$ are pairwise disjoint. Using also $V(B) \cap W_i = \emptyset$ for $i \in \{1, \dots, k\}$, we obtain $\sum_{i=1}^{k} c(E(P_i)) \leq \sum_{v \in V \setminus V(B)} 2y_v$. Together with (4.7), this implies (4.4).

Finally, we prove (4.5). Let D be a connected component of (V, F) with $V(D) \cap V(B) = \emptyset$. By property *(iii)*, $c(E(D)) = \sum_{v \in V(D)} |F \cap \delta^-(v)| \cdot 2y_v$. Because the sets $W_1, \dots, W_k$ are pairwise disjoint, we have $|F \cap \delta^-(v)| \leq |\bar{F} \cap \delta^-(v)| + 1$ for every vertex $v \in V(D)$. By Lemma 4.23, this implies $|F \cap \delta^-(v)| \leq 3$ for every vertex $v \in V(D)$ with $y_v > 0$. This shows (4.5) and concludes the proof of Theorem 4.9.

4.4 Algorithm for vertebrate pairs

In this section we present an algorithm for vertebrate pairs. This algorithm is essentially due to Svensson [Sve15] who used it for node-weighted ATSP instances. Later Svensson, Tarnawski, and Végh adapted the algorithm to work with vertebrate pairs [STV19]. Here, we present an improved variant of their algorithm.

As a subroutine we will use the Subtour Cover algorithm from Theorem 4.9. In order to exhibit the dependence of the approximation guarantee of the algorithm on the subroutine we introduce the notion of an (α, κ, β)-*algorithm* for Subtour Cover. Theorem 4.9 yields a $(3, 2, 1)$-algorithm for Subtour Cover.

Definition 4.25. *Let $\alpha, \kappa, \beta \geq 0$. An* (α, κ, β)-algorithm *for Subtour Cover is a polynomial-time algorithm that computes a solution F for every instance $(\mathcal{I}, B, H)$ such that*

$$c(F) \;\leq\; \kappa \cdot \mathrm{LP}(\mathcal{I}) + \beta \cdot \sum_{v \in V \setminus V(B)} 2y_v, \tag{4.8}$$

and for every connected component D of (V, F) with $V(D) \cap V(B) = \emptyset$ we have

$$c(E(D)) \;\leq\; \alpha \cdot \sum_{v \in V(D)} 2y_v. \tag{4.9}$$

Let $\alpha, \kappa, \beta \geq 0$ such that there is an (α, κ, β)-*algorithm* $\mathcal{A}$ for Subtour Cover and let $\varepsilon > 0$ be a fixed constant. The goal of this section is to show that there is a polynomial-time $(\kappa, 4\alpha + \beta + 1 + \varepsilon)$-algorithm for vertebrate pairs.

4.4.1 Outline

Let $(\mathcal{I}, B)$ be a vertebrate pair. Svensson's algorithm is initialized with an Eulerian multi-set $\tilde{H} \subseteq E[V \setminus V(B)]$ and then computes either a "better" initialization $\tilde{H}'$ or extends $\tilde{H}$ to a solution H of the given vertebrate pair $(\mathcal{I}, B)$.

The initialization $\tilde{H}$ of the algorithm will always be *light* (see Definition 4.26). To define what a *light* edge set is, we introduce a function $\ell : V \to \mathbb{R}_{\geq 0}$. For $v \in V$ we set

$$\ell(v) \;:=\; \begin{cases} (1+\varepsilon') \cdot 2\alpha \cdot 2y_v + \frac{\varepsilon'}{n} \cdot \sum_{u \in V \setminus V(B)} 2y_u & \text{, if } v \in V \setminus V(B) \\ \frac{\kappa \cdot \mathrm{LP}(\mathcal{I}) + \beta \cdot \sum_{u \in V \setminus V(B)} 2y_u}{|V(B)|} & \text{, if } v \in V(B), \end{cases}$$

where $\varepsilon' := \frac{\varepsilon}{3+4\alpha+\frac{1}{2\alpha}}$.

Definition 4.26. *Let $\tilde{H}$ be a (multi-)subset of E. We call $\tilde{H}$* light *if $c(E(D)) \leq \ell(V(D))$ for every connected component D of $(V, \tilde{H})$.*

Note that for $v \in V \setminus V(B)$ the first term of the definition of $\ell(v)$ is proportional to the corresponding dual variable y_v. We need the additional term $\frac{\varepsilon'}{n} \cdot \sum_{u \in V \setminus V(B)} 2y_u$ to guarantee that $\ell(v)$ cannot be too close to zero; see the proof of Lemma 4.34. For vertices in $V(B)$ we will only need that $\ell(V(B)) = \kappa \cdot \mathrm{LP}(\mathcal{I}) + \beta \cdot \sum_{u \in V \setminus V(B)} 2y_u$.

To measure what a "better" initialization for Svensson's algorithm is, we introduce a potential function Φ. For a multi-subset $\tilde{H}$ of $E[V \setminus V(B)]$ such that the connected components of $(V \setminus V(B), \tilde{H})$ have vertex sets $\tilde{W}_1, \ldots, \tilde{W}_k$, we write

$$\Phi(\tilde{H}) = \sum_{i=1}^{k} \ell(\tilde{W}_i)^{1+p},$$

where $p \coloneqq \log_{1+\varepsilon'}(\frac{2+\varepsilon'}{\varepsilon'})$. The following lemma states the result of Svensson's algorithm.

Lemma 4.27. *Let $\alpha, \kappa, \beta \geq 0$ such that there is an* (α, κ, β)-algorithm *for Subtour Cover and let $\varepsilon > 0$ be a fixed constant. Then there exists a constant $C > 0$ such that the following holds.*

Given a vertebrate pair $(\mathcal{I}, B)$ with $\mathcal{I} = (G, \mathcal{L}, x, y)$ and a light Eulerian multi-subset $\tilde{H}$ of $E[V \setminus V(B)]$ we can compute in polynomial time

(a) a solution H for the vertebrate pair $(\mathcal{I}, B)$ such that

$$c(H) \leq \ell(V(B)) + (2 + \tfrac{1}{2\alpha}) \cdot \ell(V \setminus V(B)), \tag{4.10}$$

or

(b) a light Eulerian multi-subset $\tilde{H}'$ of $E[V \setminus V(B)]$ such that

$$\Phi(\tilde{H}') - \Phi(\tilde{H}) \;>\; \left(\tfrac{1}{C \cdot n^2} \cdot \ell(V \setminus V(B))\right)^{1+p}. \tag{4.11}$$

From Lemma 4.27 we can derive the main result of this section.

Theorem 4.28. *Let $\alpha, \kappa, \beta \geq 0$ such that there is an* (α, κ, β)-algorithm *for Subtour Cover and let $\varepsilon > 0$ be a fixed constant.*

Then there is a polynomial-time $(\kappa, 4\alpha + \beta + 1 + \varepsilon)$-algorithm for vertebrate pairs.

Proof. Define $\varepsilon' := \frac{\varepsilon}{3+4\alpha+\frac{1}{2\alpha}}$, p, ℓ and Φ as above. We start with $\tilde{H} = \emptyset$ and apply Lemma 4.27. If we obtain a set $\tilde{H}'$ as in Lemma 4.27 *(b)*, we set $\tilde{H} := \tilde{H}'$ and iterate,

i.e. we apply Lemma 4.27 again until we obtain a set H as in Lemma 4.27 *(a)*. Since $0 \leq \Phi(\tilde{H}) \leq \ell(V \setminus V(B))^{1+p}$, we need at most $\left(C \cdot n^2\right)^{1+p}$ iterations. At the end, the algorithm guaranteed by Lemma 4.27 returns a solution H for the vertebrate pair $(\mathcal{I}, B)$ such that

$$\begin{aligned} c(H) &\leq \ell(V(B)) + \left(2 + \tfrac{1}{2\alpha}\right) \cdot \ell(V \setminus V(B)) \\ &\leq \kappa \cdot \mathrm{LP}(I) + \left(\beta + \left(2 + \tfrac{1}{2\alpha}\right) \cdot \left((1+\varepsilon') \cdot 2\alpha + \varepsilon'\right)\right) \cdot \sum_{v \in V \setminus V(B)} 2y_v \\ &= \kappa \cdot \mathrm{LP}(I) + \left(\beta + 4\alpha + 1 + 4\alpha \cdot \varepsilon' + \varepsilon' + (2 + \tfrac{1}{2\alpha}) \cdot \varepsilon'\right) \cdot \sum_{v \in V \setminus V(B)} 2y_v \\ &= \kappa \cdot \mathrm{LP}(I) + (4\alpha + \beta + 1 + \varepsilon) \cdot \sum_{v \in V \setminus V(B)} 2y_v. \end{aligned}$$

□

4.4.2 Basic properties of the function ℓ and algorithm $\mathcal{A}$

In this section we describe the key properties of the function ℓ and our given (α, κ, β)-algorithm $\mathcal{A}$ for Subtour Cover.

Lemma 4.29. *Let $\mathcal{A}$ be an (α, κ, β)-algorithm for Subtour Cover. Let F be the output of $\mathcal{A}$ applied to an instance $(\mathcal{I}, B, H)$.*

(i) For every connected component D of (V, F) with $V(D) \cap V(B) = \emptyset$ we have

$$c(E(D)) \leq \frac{1}{2(1+\varepsilon')} \cdot \ell(V(D)).$$

(ii) Let the graph D_B be the union of all connected components D of (V, F) with $V(D) \cap V(B) \neq \emptyset$. Then

$$c(E(D_B)) \leq \ell(V(B)).$$

Proof. The claimed properties follow directly from the definition of ℓ and the definition of an (α, κ, β)-algorithm for Subtour Cover: property (i) follows from (4.9) and property (ii) follows from (4.8). □

The next lemma will be needed to show that Svensson's algorithm makes sufficient progress when finding "a better initialization". The proof will show that we could also replace the term n^2 in the statement of the lemma by n. However, the weaker bound stated in the lemma suffices and we will exploit this later (in Section 4.6) where we apply Svensson's algorithm to graph ATSP.

Lemma 4.30. *There exists a constant $C > 0$ such that for every vertex $v \in V \setminus V(B)$ we have*

$$\ell(v) \geq \frac{1}{C \cdot n^2} \cdot \ell(V \setminus V(B)).$$

Proof. We have

$$\begin{aligned}\ell(V \setminus V(B)) &\leq (1+\varepsilon') \cdot 2\alpha \cdot \sum_{u \in V \setminus V(B)} 2y_u + \varepsilon' \cdot \sum_{u \in V \setminus V(B)} 2y_u \\ &\leq \big((1+\varepsilon') \cdot 2\alpha + \varepsilon'\big) \cdot \sum_{u \in V \setminus V(B)} 2y_u.\end{aligned}$$

Therefore, for every vertex $v \in V \setminus V(B)$ we have

$$\ell(v) \;\geq\; \frac{\varepsilon'}{n} \cdot \sum_{u \in V \setminus V(B)} 2y_u \;\geq\; \frac{\varepsilon'}{((1+\varepsilon') \cdot 2\alpha + \varepsilon') \cdot n} \cdot \ell(V \setminus V(B)),$$

which completes the proof because α and ε' are constants. □

The following property of ℓ is not crucial for obtaining a constant-factor approximation, but allows us to obtain a better approximation ratio.

Lemma 4.31. *For every circuit C with $E(C) \cap \delta(L) = \emptyset$ for all $L \in \mathcal{L}_{\geq 2}$, we have*

$$c(E(C)) \leq \frac{1}{2\alpha(1+\varepsilon')} \cdot \ell(V(C)).$$

Proof. Let C be a circuit with $E(C) \cap \delta(L) = \emptyset$ for all $L \in \mathcal{L}_{\geq 2}$. Then we have $c(E(C)) = \sum_{v \in V(C)} 2y_v \leq \frac{1}{(1+\varepsilon') \cdot 2\alpha} \cdot \ell(V(C))$. □

Remark 4.32
In the following sections we will only use Lemma 4.29, Lemma 4.30 and Lemma 4.31 and we will not use the precise definition of ℓ anymore. We will also only use that $\mathcal{A}$ computes feasible solutions for instances $(\mathcal{I}, B, H)$ of Subtour Cover (and fulfills Lemma 4.29). Moreover, we will not directly use the fact that $\mathcal{I}$ is a laminarly weighted instance of ATSP anymore and $\mathcal{L}_{\geq 2}$ could be any laminar family; this is only used in the given algorithm $\mathcal{A}$ for Subtour Cover that we use as a black-box here. These observations will be useful later in Section 4.6, where we will again apply Svensson's algorithm, but with a different definition of ℓ, $\mathcal{A}$ and $\mathcal{L}_{\geq 2}$, to obtain a better approximation algorithm for graph ATSP.

4.4.3 Finding a better initialization

In this section we discuss how Svensson's algorithm finds in certain cases a better initialization $\tilde{H}'$. We will need the following well-known statement about the knapsack problem.

Lemma 4.33. *Suppose we are given a finite set I of items and for every item $j \in I$ a weight $w_j > 0$ and a profit $p_j \geq 0$. Moreover, let $\bar{w} < \sum_{j \in I} w_j$ be a given weight limit.*

Then we can compute in polynomial time a set $J \subseteq I$ such that

- $\sum_{j \in J} w_j \;\leq\; \bar{w}$, *and*
- $\sum_{j \in J} p_j \;\geq\; \frac{\bar{w}}{\sum_{j \in I} w_j} \cdot \sum_{j \in I} p_j - \max_{j \in I}\; p_j$

Proof. We run the following greedy algorithm. Sort the items with nonincreasing ratio $\frac{p_j}{w_j}$. Consider the items in this order and, starting with $J = \emptyset$, add items to the set J as long as $\sum_{j\in J} w_j \le \bar{w}$. Then adding the next item to J would result in a set J' with $\sum_{j\in J'} w_j > \bar{w}$. By the sorting of the items,

$$\sum_{j\in J'} p_j \;=\; \frac{\sum_{j\in J'} p_j}{\sum_{j\in J'} w_j} \cdot \sum_{j\in J'} w_j \;\ge\; \frac{\sum_{j\in I} p_j}{\sum_{j\in I} w_j} \cdot \sum_{j\in J'} w_j \;>\; \frac{\sum_{j\in I} p_j}{\sum_{j\in I} w_j} \cdot \bar{w}.$$

Because $J' \setminus J$ contains only one element, this implies

$$\sum_{j\in J} p_j \;\ge\; \frac{\sum_{j\in I} p_j}{\sum_{j\in I} w_j} \cdot \bar{w} - \max_{j\in I}\; p_j \;=\; \frac{\bar{w}}{\sum_{j\in I} w_j} \cdot \sum_{j\in I} p_j - \max_{j\in I}\; p_j.$$

□

Let $\tilde{H}$ be a light Eulerian multi-subset of $E[V \setminus V(B)]$. Let $\tilde{W}_0 = V(B)$ and let $\tilde{W}_1, \dots, \tilde{W}_k$ be the vertex sets of the connected components of $(V \setminus V(B), \tilde{H})$, ordered so that $\ell(\tilde{W}_1) \ge \cdots \ge \ell(\tilde{W}_k)$. For a connected multi-subgraph D of G we define the *index* of D to be

$$\mathrm{ind}(D) \;:=\; \min\{j \in \{0,\dots,k\} : V(D) \cap \tilde{W}_j \neq \emptyset\}.$$

The following is the main lemma that we will use to find a better initialization $\tilde{H}'$.

Lemma 4.34. *Let D be a connected and Eulerian multi-subgraph of the graph G with $V(D) \cap V(B) = \emptyset$,*

$$c(E(D)) \;\le\; \frac{2}{2+\varepsilon'} \cdot \ell(V(D)), \tag{4.12}$$

and

$$\ell(V(D)) \;>\; (1+\varepsilon') \cdot \ell(\tilde{W}_{\mathrm{ind}(D)}). \tag{4.13}$$

Then we can compute in polynomial time a light Eulerian multi-subset $\tilde{H}'$ of $E[V\setminus V(B)]$ such that (4.11) *holds.*

Proof. Let $I := \{j \in \{0,\dots,k\} : V(D) \cap \tilde{W}_j \neq \emptyset\}$ and $i := \min I = \mathrm{ind}(D)$. We have $i > 0$ because $V(D) \cap V(B) = \emptyset$. We will compute a subset J of I and replace the components $\tilde{H}[\tilde{W}_j]$ for $j \in I$ by one new component that is the union of $E(D)$ and all $\tilde{H}[\tilde{W}_j]$ with $j \in J$. More precisely, we set

$$\tilde{H}' \;:=\; \bigcup_{h\in\{1,\dots,k\}\setminus I} \tilde{H}[\tilde{W}_h] \;\dot\cup\; E(D) \;\dot\cup\; \bigcup_{j\in J} \tilde{H}[\tilde{W}_j].$$

See Figure 4.8. Let D^* be the connected component of $(V, \tilde{H}')$ with edge set

$$E(D) \,\dot\cup \bigcup_{j\in J} \tilde{H}[\tilde{W}_j].$$

We will choose J such that

$$\sum_{j\in J} \ell(\tilde{W}_j \cap V(D)) \;\le\; \frac{\varepsilon'}{2+\varepsilon'} \cdot \ell(V(D)). \tag{4.14}$$

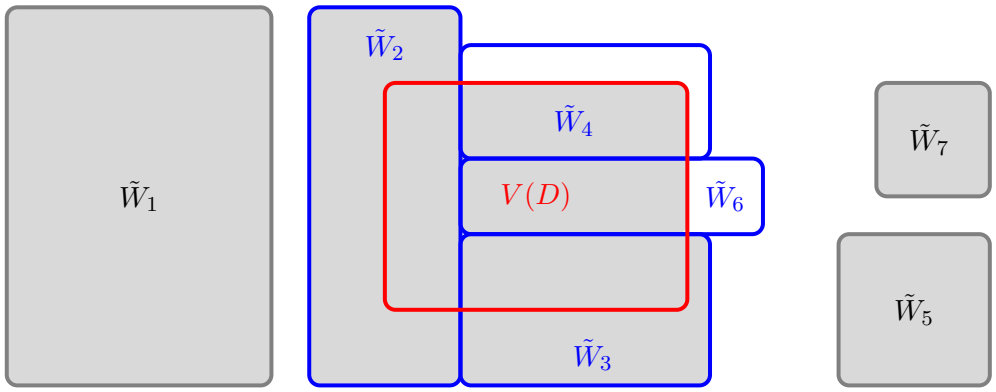

Figure 4.8: Illustration of the proof of Lemma 4.34. The gray and blue rectangles show the partition of $V \setminus V(B)$ into $\tilde{W}_1, \dots, \tilde{W}_7$. In red we see the vertex set $V(D)$ of the given connected graph D. The rectangles with blue boundary show the sets $\tilde{W}_i$ with $i \in I$. In this example $I = \{2, 3, 4, 6\}$. The filled areas show vertex sets of connected components of $(V \setminus V(B), \tilde{H}')$. In this example we have $J = \{2, 3\}$. The connected components $\tilde{H}[\tilde{W}_1]$, $\tilde{H}[\tilde{W}_5]$, and $\tilde{H}[\tilde{W}_7]$ remain unchanged and we get a new component D^* with vertex set $V(D) \cup \tilde{W}_2 \cup \tilde{W}_3$; we also get singleton components (without edges) for all vertices in $\tilde{W}_4 \setminus V(D)$ and $\tilde{W}_6 \setminus V(D)$.

We first show that then $c(E(D^*)) \leq \ell(V(D^*))$, which implies that $\tilde{H}'$ is light. Indeed, using (4.12) in the first inequality and (4.14) in the last inequality,

$$\begin{aligned} c(E(D^*)) &\leq \frac{2}{2+\varepsilon'} \cdot \ell(V(D)) + \sum_{j \in J} \ell(\tilde{W}_j) \\ &= \frac{2}{2+\varepsilon'} \cdot \ell(V(D)) + \sum_{j \in J} \ell(\tilde{W}_j \setminus V(D)) + \sum_{j \in J} \ell(\tilde{W}_j \cap V(D)) \\ &\leq \frac{2}{2+\varepsilon'} \cdot \ell(V(D)) + \sum_{j \in J} \ell(\tilde{W}_j \setminus V(D)) + \frac{\varepsilon'}{2+\varepsilon'} \cdot \ell(V(D)) \\ &= \ell\left(V(D^*)\right). \end{aligned}$$

We conclude the proof by showing that we can choose J such that (4.14) and (4.11) hold. To this end, we would like to make the new component, spanning $V(D) \cup \bigcup_{j \in J} \tilde{W}_j$, as large as possible. More precisely, we want to maximize $\sum_{j \in J} \ell(\tilde{W}_j \setminus V(D))$ subject to (4.14). This is a knapsack problem: the items are indexed by I, and item $j \in I$ has weight $w_j = \ell(\tilde{W}_j \cap V(D))$ and profit $p_j = \ell(\tilde{W}_j \setminus V(D))$. Since $\sum_{j \in I} \ell(\tilde{W}_j \cap V(D)) = \ell(V(D))$, the weight limit $\bar{w} = \frac{\varepsilon'}{2+\varepsilon'} \cdot \ell(V(D))$ is an $\frac{\varepsilon'}{2+\varepsilon'}$ fraction of the total weight of all items. Since any item $j \in I$ has profit at most $\ell(\tilde{W}_j \setminus V(D)) \leq \ell(\tilde{W}_j) \leq \ell(\tilde{W}_i)$, Lemma 4.33 yields a set J with (4.14) and

$$\sum_{j \in J} \ell(\tilde{W}_j \setminus V(D)) \;\geq\; \frac{\varepsilon'}{2+\varepsilon'} \cdot \sum_{j \in I} \ell(\tilde{W}_j \setminus V(D)) \;-\; \ell(\tilde{W}_i).$$

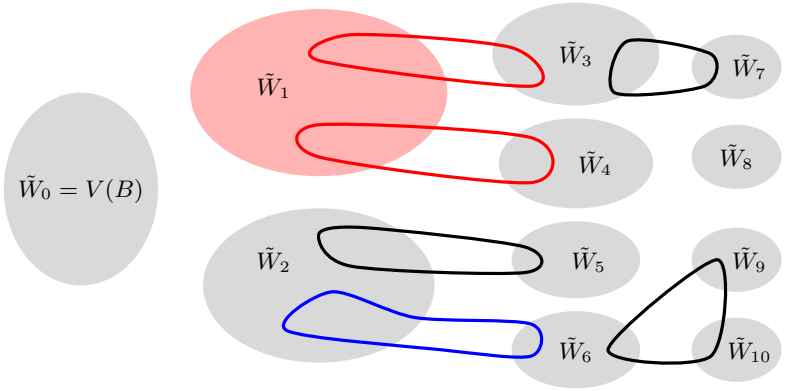

Figure 4.9: Illustration of Lemma 4.35 and Lemma 4.36. Here the filled ellipses show the partition $\tilde{W}_0, \ldots, \tilde{W}_{10}$ of V. The curves show a possible solution F to some instance $(\mathcal{I}, H)$ of Subtour Cover; the set H is not shown here. In red we see a subgraph D as in Lemma 4.35: here the red curves are the graph F_1 and D is the union of F_1 and $\tilde{H}[\tilde{W}_1]$. In blue we see a subgraph D as in Lemma 4.36: here D is a single connected component of F and in this example we have $\mathrm{ind}(D) = 2$.

Finally we show (4.11). Using (4.13) in the strict inequality, $(1+\varepsilon')^p = \frac{2+\varepsilon'}{\varepsilon'}$ in the second equation, and $\ell(\tilde{W}_i) \geq \ell(\tilde{W}_j)$ for all $j \in I$ in the last inequality, we obtain

$$\begin{aligned}
\ell(V(D^*))^{1+p} &= \left(\ell(V(D)) + \sum_{j\in J} \ell(\tilde{W}_j \setminus V(D))\right)^{1+p} \\
&\geq \ell(V(D))^p \cdot \left(\ell(V(D)) + \frac{\varepsilon'}{2+\varepsilon'} \sum_{j\in I} \ell(\tilde{W}_j \setminus V(D)) - \ell(\tilde{W}_i)\right) \\
&> \left((1+\varepsilon') \cdot \ell(\tilde{W}_i)\right)^p \cdot \\
&\qquad \left(\frac{\varepsilon'}{2+\varepsilon'} \cdot \ell(\tilde{W}_i) + \frac{\varepsilon'}{2+\varepsilon'} \ell(V(D)) + \frac{\varepsilon'}{2+\varepsilon'} \sum_{j\in I} \ell(\tilde{W}_j \setminus V(D))\right) \\
&= \frac{2+\varepsilon'}{\varepsilon'} \cdot \ell(\tilde{W}_i)^p \cdot \left(\frac{\varepsilon'}{2+\varepsilon'} \cdot \ell(\tilde{W}_i) + \frac{\varepsilon'}{2+\varepsilon'} \sum_{j\in I} \ell(\tilde{W}_j)\right) \\
&\geq \ell(\tilde{W}_i)^{1+p} + \sum_{j\in I} \ell(\tilde{W}_j)^{1+p}.
\end{aligned}$$

and hence

$$\Phi(\tilde{H}') - \Phi(\tilde{H}) \;\geq\; \ell(V(D^*))^{1+p} - \sum_{j\in I} \ell(\tilde{W}_j)^{1+p} \;>\; \ell(\tilde{W}_i)^{1+p}.$$

Since $\tilde{W}_i$ contains at least one vertex, by Lemma 4.30, $\ell(\tilde{W}_i) \geq \frac{1}{C \cdot n^2} \cdot \ell(V \setminus V(B))$ for the constant C from Lemma 4.30. □

The two different ways how we obtain D during Svensson's algorithm are described by Lemma 4.35 and Lemma 4.36. See also Figure 4.9.

Lemma 4.35. *Let $\mathcal{A}$ be an (α, κ, β)-algorithm for Subtour Cover. Let F be the output of $\mathcal{A}$ applied to an instance $(\mathcal{I}, B, H)$. For $i \in \{0, \dots, k\}$ let the graph F_i be the union of the connected components D' of (V, F) with $\operatorname{ind}(D') = i$.*

Suppose we have $c(E(F_i)) > \ell(\tilde{W}_i)$ for some $i \in \{0, \dots, k\}$. Then the union

$$D := \left(\tilde{W}_i \cup V(F_i),\ \tilde{H}[\tilde{W}_i] \cup E(F_i)\right)$$

of $\tilde{H}[\tilde{W}_i]$ and F_i fulfills the conditions Lemma 4.34, i.e. D is a connected Eulerian multi-subgraph of G with $V(D) \cap V(B) = \emptyset$, (4.12) and (4.13).

Proof. Let $i \in \{0, \dots, k\}$ such that $c(E(F_i)) > \ell(\tilde{W}_i)$. Note that $i > 0$ because $c(E(F_0)) \leq \ell(\tilde{W}_0)$ by Lemma 4.29 (ii). This implies $V(D) \cap V(B) = \emptyset$. Moreover, we have

$$\frac{1}{2+2\varepsilon'} \cdot \ell(V(D)) \ \geq\ \frac{1}{2+2\varepsilon'} \cdot \ell(V(F_i)) \ \geq\ c(E(F_i)) \ >\ \ell(\tilde{W}_i),$$

where the second inequality holds by Lemma 4.29 (i). This shows (4.13) and implies

$$c(E(D)) = c(\tilde{H}[\tilde{W}_i] \cup E(F_i)) \ \leq\ \ell(\tilde{W}_i) + c(E(F_i)) \ \leq\ \frac{2}{2+2\varepsilon'} \cdot \ell(V(D)).$$

Therefore also (4.12) holds. □

Lemma 4.36. *Let $\mathcal{A}$ be an (α, κ, β)-algorithm for Subtour Cover. Let F be the output of $\mathcal{A}$ applied to an instance $(\mathcal{I}, B, H)$. Suppose (V, F) has a connected component D with $\operatorname{ind}(D) > 0$ and*

$$l(V(D)) > (1+\varepsilon') \cdot l(\tilde{W}_{\operatorname{ind}(D)}).$$

Then D fulfills the conditions Lemma 4.34, i.e. D is a connected Eulerian multi-subgraph of G with $V(D) \cap V(B) = \emptyset$, (4.12) and (4.13).

Proof. We have (4.13) by assumption. Moreover, $V(D) \cap V(B) = \emptyset$ because $\operatorname{ind}(D) > 0$. Since D is a connected component of (V, F) that does not intersect the backbone, Lemma 4.29 (i) implies

$$c(E(D)) \ \leq\ \frac{1}{2+2\varepsilon'} \cdot l(V(D)) \ \leq\ \frac{2}{2+\varepsilon'} \cdot l(V(D)),$$

implying (4.12). □

4.4.4 Svensson's algorithm

In this section we prove Lemma 4.27. To this end we consider Algorithm 2, essentially due to Svensson [Sve15]. We maintain an Eulerian edge set H which is initialized with $H = \tilde{H}$. Then we iterate the following steps. First, we call the given algorithm for Subtour Cover, then we try to find an improved initialization $\tilde{H}'$ as discussed in the previous section, and finally, if we could not find a better initialization, we extend the set H. The careful update of H in step 3 of Algorithm 2 is illustrated in Figure 4.10.

To implement step (3c), consider each edge $e = (v, w) \in \delta^+(V(Z))$ and compute a shortest w-v-path P in $\left(V, E \setminus \left(\cup_{L \in \mathcal{L}_{\geq 2}} \delta(L)\right)\right)$ and check if $c(e) + c(P) \leq \frac{1}{2\alpha} \cdot \ell(\tilde{W}_{\operatorname{ind}(Z)})$.

Algorithm 2: Svensson's Algorithm

Input: a vertebrate pair $(\mathcal{I}, B)$ with $\mathcal{I} = (G, \mathcal{L}, x, y)$,
a light Eulerian multi-subset $\tilde{H} \subseteq E[V \setminus V(B)]$,
$\alpha, \kappa, \beta \geq 0$, $\varepsilon > 0$, and
an (α, κ, β)-algorithm $\mathcal{A}$ for Subtour Cover

Output: either $\tilde{H}'$ as in Lemma 4.27 *(b)* or H as in Lemma 4.27 *(a)*

Let $\tilde{W}_0 := V(B)$ and let $\tilde{W}_1, \dots, \tilde{W}_k$ be the vertex sets of the connected components of $(V \setminus V(B), \tilde{H})$ such that $\ell(\tilde{W}_1) \geq \ell(\tilde{W}_2) \geq \cdots \geq \ell(\tilde{W}_k)$.
Set $H := \tilde{H}$.

While $(V, E(B) \cup H)$ is not connected, repeat the following:

1. **Compute a solution to Subtour Cover:**

   (1a) Apply $\mathcal{A}$ to the Subtour Cover instance $(\mathcal{I}, B, H)$ to obtain a solution F'.

   (1b) Let F result from F' by deleting all edges of connected components of (V, F') whose vertex sets are contained in a connected component of $(V, E(B) \cup H)$.

2. **Try to find a better initialization $\tilde{H}'$:**

   For $i \in \{0, \dots, k\}$ let the graph F_i be the union of the connected components D' of (V, F) with $\mathrm{ind}(D') = i$.

   (2a) If for some $i \in \{0, \dots, k\}$ we have $c(E(F_i)) > \ell(\tilde{W}_i)$, apply Lemma 4.34 to $D = (\tilde{W}_i \cup V(F_i), \tilde{H}[\tilde{W}_i] \cup E(F_i))$ to obtain an edge set $\tilde{H}'$. Then return $\tilde{H}'$.

   (2b) If (V, F) has a connected component D with $\ell(V(D)) > (1 + \varepsilon') \cdot \ell(\tilde{W}_{\mathrm{ind}(D)})$ and $\mathrm{ind}(D) > 0$, apply Lemma 4.34 to obtain an edge set $\tilde{H}'$. Then return $\tilde{H}'$.

3. **Extend H:**

   (3a) Set $X := \emptyset$.

   (3b) Select the connected component Z of $(V, H \,\dot\cup\, F \,\dot\cup\, X)$ for which $\mathrm{ind}(Z)$ is largest.

   (3c) If there is a circuit C with
   - $E(C) \cap \delta(V(Z)) \neq \emptyset$,
   - $E(C) \cap \delta(L) = \emptyset$ for all $L \in \mathcal{L}_{\geq 2}$, and
   - $c(E(C)) \leq \frac{1}{2\alpha} \cdot \ell(\tilde{W}_{\mathrm{ind}(Z)})$,

   then add $E(C)$ to X and go to step (3b).

   (3d) Add the edges of $(V, F \,\dot\cup\, X)[V(Z)]$ to H.

Return H.

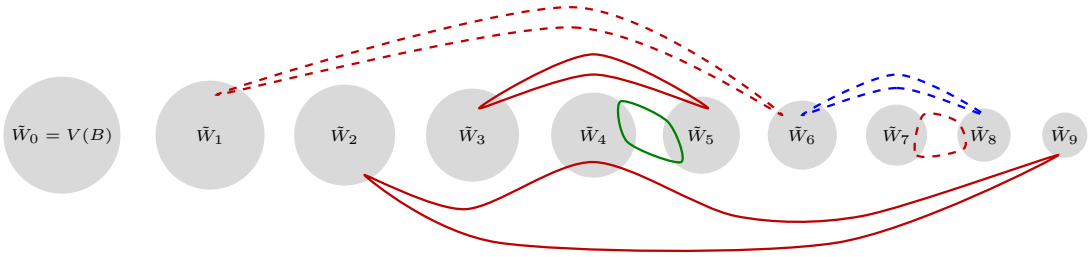

Figure 4.10: An illustration of step 3 in the first iteration of Svensson's algorithm. The edge set F is shown in red. First the component Z with vertex set $\tilde{W}_7 \cup \tilde{W}_8$ is considered, with $\mathrm{ind}(Z) = 7$. We may find the blue circuit C with $c(E(C)) \leq \frac{1}{2\alpha}\ell(\tilde{W}_7)$. After adding $E(C)$ to X, the component Z with vertex set $\tilde{W}_3 \cup \tilde{W}_5$ is considered next, with $\mathrm{ind}(Z) = 3$. Then we may find the green circuit C' with $c(E(C')) \leq \frac{1}{2\alpha}\ell(\tilde{W}_3)$. Then $E(C')$ is added to X, and now $(V, H \,\dot\cup\, F \,\dot\cup\, X)$ has three connected components. The component Z with vertex set $\tilde{W}_2 \cup \tilde{W}_3 \cup \tilde{W}_4 \cup \tilde{W}_5 \cup \tilde{W}_9$ is considered next. Suppose there is no circuit C'' connecting it to the rest and with $c(E(C'')) \leq \frac{1}{2\alpha}\ell(\tilde{W}_2)$. Then the edges drawn as solid curves are added to H, concluding the first iteration.

Note that adding $E(C)$ to X in step (3c) decreases the number of connected components of $(V, H \,\dot\cup\, F \,\dot\cup\, X)$, and adding edges to H in step (3d) decreases the number of connected components of (V, H). Thus the procedure terminates after a polynomial number of steps.

Also notice that step (1b) maintains all properties required for the output of an (α, κ, β)-algorithm for Subtour Cover. Hence, the computation of F in step 1 (including both step (1a) and step (1b)) is an (α, κ, β)-algorithm for Subtour Cover. Therefore, we can apply Lemma 4.35 for step (2a) and Lemma 4.36 for step (2b) to show that the application of Lemma 4.34 is indeed possible.

We conclude that if Algorithm 2 returns a (multi-)set $\tilde{H}'$ in step 2, then $\tilde{H}'$ is a multi-set as in Lemma 4.27 *(b)*.

Now suppose the algorithm does not terminate in step 2. Since H remains Eulerian throughout the algorithm and $(V, E(B) \cup H)$ is connected at the end of Algorithm 2, the returned edge set H is a solution for the vertebrate pair $(\mathcal{I}, B)$. It remains to show the upper bound (4.10) on the cost of H. Initially we have $c(H) = c(\tilde{H}) \leq \ell(V \setminus V(B))$. We bound the cost of the X-edges and the cost of the F-edges added to H separately.

Lemma 4.37. *The total cost of all X-edges that are added to H is at most*

$$\frac{1}{2\alpha} \cdot \ell(V \setminus V(B)).$$

Proof. A circuit C that is selected in step (3c) and will later be added to H connects Z with another connected component Y with $\mathrm{ind}(Y) < \mathrm{ind}(Z)$. We say that it marks $\mathrm{ind}(Z)$. It has cost at most $\frac{1}{2\alpha} \cdot \ell(\tilde{W}_{\mathrm{ind}(Z)})$. No circuit added later can mark $\mathrm{ind}(Z)$ because the new connected component of $(V, H \,\dot\cup\, F \,\dot\cup\, X)$ containing $Y \cup Z$ will have smaller index by the choice of Z. Hence the total cost of the added circuits is at most $\frac{1}{2\alpha} \cdot \sum_{i=1}^{k} \ell(\tilde{W}_i) = \frac{1}{2\alpha} \cdot \ell(V \setminus V(B))$. □

Lemma 4.38. *The total cost of all F-edges that are added to H is at most $\ell(V)$.*

Proof. Let Z^t denote Z at the end of iteration t of the while-loop. Let F_i^t be the graph F_i in iteration t if the set of edges of F_i is nonempty and is added to H at the end of this iteration, and let $F_i^t = \emptyset$ otherwise.

For $i = 0, \dots, t$ the total cost of F_i^t is $c(E(F_i^t)) \leq \ell(\tilde{W}_i)$ by step (2a). We claim that for any i, at most one of the F_i^t is nonempty. Then summing over all i and t concludes the proof.

Suppose there are $t_1 < t_2$ such that $F_i^{t_1} \neq \emptyset$ and $F_i^{t_2} \neq \emptyset$. We have $i > 0$ because otherwise the algorithm would terminate after iteration t_1 by the choice of Z^{t_1}. Then $V(F_i^{t_1}) \subseteq V(Z^{t_1})$ and thus $\tilde{W}_i \subseteq V(Z^{t_1})$. Moreover, $F_i^{t_2}$ contains a vertex of $\tilde{W}_i$ and is not completely contained in Z^{t_1} by step (1b) of the algorithm. Thus, $F_i^{t_2}$ contains a circuit C with $E(C) \cap \delta(V(Z^{t_1})) \neq \emptyset$. We have $E(C) \cap \delta(L) \subseteq E(F_i^{t_2}) \cap \delta(L) = \emptyset$ for all $L \in \mathcal{L}_{\geq 2}$ because $V(F_i^{t_2}) \cap V(B) = \emptyset$ (since $i > 0$) and F is a solution to Subtour Cover.

If $c(E(C)) \leq \frac{1}{2\alpha} \cdot \ell(\tilde{W}_{\mathrm{ind}(Z^{t_1})})$, due to step (3c), this is a contradiction to reaching step (3d) in iteration t_1 and adding Z^{t_1} there. Otherwise, let D be the connected component of $F_i^{t_2}$ containing C. Note that $\mathrm{ind}(D) = i \geq \mathrm{ind}(Z^{t_1})$.

Since C is a circuit with $E(C) \cap \delta(L) = \emptyset$ for all $L \in \mathcal{L}_{\geq 2}$, we can apply Lemma 4.31 to obtain

$$\frac{1}{(1+\varepsilon') \cdot 2\alpha} \cdot \ell(V(C)) \;\geq\; c(E(C)) \;>\; \frac{1}{2\alpha} \cdot \ell(\tilde{W}_{\mathrm{ind}(Z^{t_1})}) \;\geq\; \frac{1}{2\alpha} \cdot \ell(\tilde{W}_{\mathrm{ind}(D)}).$$

This shows

$$\ell(V(D)) \;\geq\; \ell(V(C)) \;>\; (1+\varepsilon') \cdot \ell(\tilde{W}_{\mathrm{ind}(D)}).$$

Due to step (2b), this is a contradiction to reaching step (3d) in iteration t_2 and adding $F_i^{t_2}$ there. □

Using $c(\tilde{H}) \leq \ell(V \setminus V(B))$, Lemma 4.37, and Lemma 4.38, we conclude that the cost of the returned edge set H is at most

$$\ell(V(B)) + \left(2 + \tfrac{1}{2\alpha}\right) \cdot \ell(V \setminus V(B)).$$

This concludes the proof of Lemma 4.27.

4.5 The main result

We can now combine the results of the previous sections and obtain the following.

Theorem 4.39. *For every $\varepsilon > 0$ there is a polynomial-time algorithm that computes for every instance (G, c) of ATSP a solution of cost at most $22 + \varepsilon$ times the cost of an optimum solution to* (ATSP LP)*.*

Proof. Theorem 4.9 yields a $(3, 2, 1)$-algorithm for Subtour Cover and by Theorem 4.28 this implies that there is a polynomial-time $(2, 14 + \varepsilon)$-algorithm for vertebrate pairs. Using Theorem 4.6 we then obtain a polynomial-time algorithm that finds a solution of cost at most $(22 + \varepsilon) \cdot \mathrm{LP}(\mathcal{I})$ for every ATSP instance $\mathcal{I}$. □

As a consequence of Theorem 4.39 we obtain the following.

Corollary 4.40. *The integrality ratio of* (ATSP LP) *is at most* 22.

Proof. Suppose there is an instance $\mathcal{I}$ of ATSP where $\frac{\mathrm{OPT}(\mathcal{I})}{\mathrm{LP}(\mathcal{I})} > 22$. Then there exists $\varepsilon > 0$ such that $\frac{\mathrm{OPT}(\mathcal{I})}{\mathrm{LP}(\mathcal{I})} > 22+\varepsilon$. By Theorem 4.39 we can compute an integral solution for $\mathcal{I}$ with cost at most $(22+\varepsilon)\cdot \mathrm{LP}(\mathcal{I}) < \mathrm{OPT}(\mathcal{I})$, a contradiction. □

Using the oberservation from Remark 4.7, one could slightly improve Theorem 4.39 and Corollary 4.40, but the improvement would be less than 1.

4.6 Graph ATSP

In this section we consider graph ATSP, i.e. the special case of ATSP with $c \equiv 1$. We show that for graph ATSP we can obtain a better approximation algorithm by using again Svensson's algorithm, but replacing the algorithm $\mathcal{A}$ for Subtour Cover and choosing a different function ℓ. This is essentially due to Svensson [Sve15], but we obtain a better approximation ratio because we use the improved variant of Svensson's algorithm that we explained in the previous section. Moreover, we give a family of instances that proves that the integrality ratio of (ATSP LP) is at least 2 also for graph ATSP.

Parts of this section have been published in [KTV19].

4.6.1 A $(13+\varepsilon)$-approximation algorithm for graph ATSP

Consider an instance $\mathcal{I}$ of graph ATSP. Then $\mathrm{LP}(\mathcal{I}) \geq n$, where n denotes the number of vertices. If $\mathrm{LP}(\mathcal{I}) = n$, the dual solution with $y_{\{v\}} := \frac{1}{2}$ for all vertices v and all other dual variables being 0 is an optimum dual solution. Then $\mathcal{L}_{\geq 2} = \emptyset$ and the empty graph is a backbone B. In this case an Eulerian multi edge set F is feasible for an instance $(\mathcal{I}, B, H)$ Subtour Cover if and only if it has the following property: for every connected component D of (V, H) we have $F \cap \delta(V(D)) \neq \emptyset$.

In general, we do not have $\mathcal{L}_{\geq 2} = \emptyset$ for every instance of graph ATSP. We will not use the reduction from ATSP to vertebrate pairs, but directly apply Svensson's algorithm. In Svensson's algorithm we will replace the algorithm $\mathcal{A}$ by the algorithm from the following lemma that is due to Svensson [Sve15].

Lemma 4.41. *Given a directed graph $G = (V, E)$, a solution x to* (ATSP LP)*, and an Eulerian multi-subset $H \subseteq E$, we can compute in polynomial time an Eulerian multi-set F of edges such that*

- *for every connected component D of (V, H) we have $F \cap \delta(V(D)) \neq \emptyset$, and*
- *for every connected component D of (V, F) we have $|E(D)| \leq \sum_{v \in V(D)} 3 \cdot x(\delta^-(v))$.*

Proof. Let $W_i, \ldots, W_k$ be the vertex sets of the connected components of (V, H). First, we observe that $G[W_i]$ is strongly connected for all $i \in \{1, \ldots, k\}$ because H is an Eulerian multi-subset of E.

For every set W_i with $i \in \{1, \ldots, k\}$ we do the following. First, we add a new vertex a_i. Then for every edge $(v, w) \in \delta^-(W_i)$ we add an edge (v, a_i) of the same cost. Similarly, for every edge $(v, w) \in \delta^+(W_i)$ we add an edge (a_i, w) of the same cost. Now we apply Lemma 4.17 to the set $U = W_i$ and the circulation x. For every path $P \in \mathcal{P}$

- we decrease the flow on e_P^{in} and e_P^{out} by λ_P and increase the flow on the corresponding edges in $\delta(a_i)$ by λ_P, and
- we decrease the flow on every edge $e \in E(P)$ by λ_P.

Note that we maintain a circulation. Let $\bar{G}$ be the resulting graph and let $\bar{x}$ denote the resulting circulation. We have $\bar{x}(\delta^-(a_i)) = 1$ for all $i \in \{1, \dots, k\}$. Therefore, we can compute an integral circulation $\bar{x}^*$ in $\bar{G}$ such that $\bar{x}^*(\delta^-(a_i)) = 1$ for all $i \in \{1, \dots, k\}$ and $\bar{x}^*(\delta^-(v)) \leq \lceil \bar{x}(\delta^-(v)) \rceil \leq \lceil x(\delta^-(v)) \rceil$ for all $v \in V$. Let $\bar{F}$ be the multi-set of edges with $\chi^{\bar{F}} = \bar{x}^*$.

Now we replace the edges in $\delta_{\bar{F}}(a_i)$ for $i \in \{1, \dots, k\}$ by their corresponding edges in G (i.e. by the edges from which they arose in the construction of $\bar{G}$). After this we do not necessarily have an Eulerian multi edge set anymore, but because of $\bar{x}^*(\delta^+(a_i)) = \bar{x}^*(\delta^-(a_i)) = 1$ we just added for every set W_i exactly one incoming edge $(v, s) \in \delta^-(W_i)$ and exactly one outgoing edge $(t, w) \in \delta^+(W_i)$. Because $G[W_i]$ is strongly connected, we can obtain an Eulerian multi edge set by adding an s-t-path in $G[W_i]$. Let F be the resulting multi edge set. Note that we have $|F \cap \delta^-(v)| \leq \lceil \bar{x}(\delta^-(v)) \rceil + 1$ for all $v \in V$ because the sets $W_1, \dots, W_k$ are pairwise disjoint.

For $i \in \{1, \dots, k\}$ we have $F \cap \delta(W_i) \neq \emptyset$. Moreover, for every connected component D of (V, F) we have $|E(D)| = \sum_{v \in V(D)} |\delta_F^-(v)| \leq \sum_{v \in V(D)} (\lceil x(\delta^-(v)) \rceil + 1) \leq \sum_{v \in V(D)} 3x(\delta^-(v))$, where we used $x(\delta^-(v)) \geq 1$ for all $v \in V$. □

We now explain how we apply Svensson's algorithm to graph ATSP. Let $\varepsilon > 0$. We define $\mathcal{L}_{\geq 2} := \emptyset$ and choose the backbone B to be the empty graph, i.e. $V(B) = \emptyset$ and $E(B) = \emptyset$. Moreover, we set $\alpha := 3$, $\varepsilon' := \frac{\varepsilon}{13}$ and define the function $\ell : V \to \mathbb{R}_{\geq 0}$ by

$$\ell(v) := 2\alpha(1 + \varepsilon') \cdot x(\delta^-(v)),$$

where x is an optimum solution to (ATSP LP) with $c \equiv 1$. Then we do not necessarily have a vertebrate pair because instances of graph ATSP do not necessarily have an optimum dual solution y with $y_v > 0$ only for singleton sets. Nevertheless we can run Svensson's algorithm (Algorithm 2), where we replace the algorithm $\mathcal{A}$ by the algorithm from Lemma 4.41 and we replace the strongly laminar instance $\mathcal{I}$ by the graph ATSP instance $G = (V, E)$.

As we remarked in Section 4.4.2 (Remark 4.32), for the analysis of Svensson's algorithm we only need the following properties of the function ℓ and the algorithm $\mathcal{A}$:

(i) For every connected component D of the output of $\mathcal{A}$ we have

$$|E(D)| \leq \frac{1}{2(1 + \varepsilon')} \cdot \ell(V(D)).$$

(ii) Let D_B be the union of all connected components D of the output of $\mathcal{A}$ with $V(D) \cap V(B) \neq \emptyset$. Then

$$|E(D_B)| \leq \ell(V(B)).$$

(iii) There exists a constant $C > 0$ such that for every vertex $v \in V \setminus V(B)$ we have

$$\ell(v) \geq \frac{1}{C \cdot n^2} \cdot \ell(V \setminus V(B)).$$

(iv) For every circuit C with $E(C) \cap \delta(L) = \emptyset$ for all $L \in \mathcal{L}_{\geq 2}$ we have

$$|E(C)| \leq \frac{1}{2\alpha(1+\varepsilon')} \cdot \ell(V(C)).$$

Property (i) follows directly from Lemma 4.41 and the definition of ℓ. For property (ii) there is nothing to show because B is the empty graph. Property (iv) holds because for every cycle C we have $|E(C)| = |V(C)| \leq \sum_{v \in V(C)} x(\delta^-(v))$. To show (iii) we show that for any feasible instance of graph ATSP we have LP $\leq n^2$.

We call an LP solution x minimal if there is no feasible solution $x' \neq x$ with $x' \leq x$ componentwise.

Lemma 4.42. *For every minimal solution x of* (ATSP LP)*, we have $x(E(G)) \leq n^2$, where $n = |V(G)|$.*

Proof. Choose an arbitrary root $r \in V$ and let $P = \{y \in \mathbb{R}^{E(G)}_{\geq 0} : y(\delta^-(U)) \geq 1 \text{ for } \emptyset \neq U \subseteq V \setminus \{r\}\}$. A vector is feasible for (ATSP LP) if and only if it is a circulation that belongs to P. Let $y \leq x$ be a minimal vector in P. The minimal vectors in P are the convex combinations of incidence vectors of spanning arborescences rooted at r [Edm67]; hence $y(E(G)) = n - 1$. There are cycles C_j and edge sets $S_j \subseteq C_j$ $(j = 1, \ldots, l)$ such that $x = \sum_{j=1}^{l} \lambda_j \chi^{C_j}$ and $y = \sum_{j=1}^{l} \lambda_j \chi^{S_j}$ for some positive coefficients λ_j. Note that none of the sets S_j can be empty because otherwise $x' = x - \lambda_j \chi^{C_j}$ would be a circulation that belongs to P, contradicting the minimality of x. We conclude $x(E(G)) = \sum_{j=1}^{l} \lambda_j |C_j| \leq \sum_{j=1}^{l} \lambda_j \cdot n|S_j| = n \cdot y(E(G)) = n(n-1)$. □

Lemma 4.42 implies LP $\leq n^2$. Therefore, for every vertex $v \in V$ we have $\ell(v) \geq 2\alpha(1+\varepsilon') = \frac{\ell(V)}{x(E)} \geq \frac{1}{n^2} \cdot \ell(V)$. Because $V(B) = \emptyset$, this shows (iii).

We have shown (i)–(iv) and hence as we explained in Remark 4.32, applying Svensson's algorithm (Algorithm 2) repeatedly as in the proof of Theorem 4.28 yields a solution H of graph ATSP with

$$c(H) \;\leq\; \ell(V(B)) + (2 + \tfrac{1}{2\alpha}) \cdot \ell(V \setminus V(B)) \;=\; 13 \cdot (1+\varepsilon') \cdot x(E) \;=\; (13+\varepsilon) \cdot \text{LP}.$$

This implies the main result of this section.

Theorem 4.43. *For every $\varepsilon > 0$ there is a $(13+\varepsilon)$-approximation algorithm for graph ATSP.* □

4.6.2 A lower bound on the integrality ratio.

In this section we give a sequence of instances of graph ATSP with integrality ratio converging to 2. For the general ATSP such a family of instances was given by Charikar, Goemans, and Karloff [CGK06]. Boyd and Elliott-Magwood [BEM05] gave another such family. Although they did not note this explicitly, it is easy to see that their examples are *node-weighted*, i.e. there exist non-negative node weights $c_v \geq 0$ for every vertex v such that for every edge (v, w) we have $c(v, w) = c_v + c_w$. In this section we observe that, for ATSP, node-weighted instances are not much more general than unweighted instances. From this we derive that the instances from [BEM05] can be turned into instances of graph ATSP with integrality ratio converging to 2.

Lemma 4.44. *Let $\varepsilon > 0$. Let $\mathcal{I} = (G, c)$ be a node-weighted instance of ATSP with n vertices. Then we can find in polynomial time a constant $M > 0$ and an unweighted directed graph G' with $O(\frac{n^2}{\varepsilon})$ vertices such that*

(i) $\mathrm{LP}(\mathcal{I}) \leq M \cdot \mathrm{LP}(G') \leq (1+\varepsilon)\mathrm{LP}(\mathcal{I})$,

(ii) $\mathrm{OPT}(\mathcal{I}) \leq M \cdot \mathrm{OPT}(G') \leq (1+\varepsilon)\mathrm{OPT}(\mathcal{I})$, *and*

(iii) for every tour F' in the unweighted directed graph G' there is a corresponding tour F in G such that $c(F) \leq M \cdot |F'|$ and F can be obtained from F' in polynomial time.

Proof. Let $c_v \geq 0$ ($v \in V(G)$) be the node weights, i.e., $c(v,w) = c_v + c_w$ for all $(v,w) \in E$. Let $c(V(G)) = \sum_{v \in V(G)} c_v$ denote the sum of all node weights. If $c(V(G)) = 0$, the instance is trivial, we can choose G' to consist of a single vertex.

Otherwise let $n = |V(G)|$, $M := \frac{2\varepsilon \cdot c(V(G))}{n^2}$ and $\bar{c}_v := \lfloor \frac{2c_v}{M} \rfloor$ for all $v \in V(G)$. Replace every vertex v of G with $\bar{c}_v > 0$ by two vertices v^- and v^+, such that v^- inherits the entering edges and v^+ inherits the outgoing edges, and add a path P_v of $\bar{c}_v$ edges from v^- to v^+. This defines G'. Note that $|V(G')| = n + \sum_{v \in V(G)} \bar{c}_v \leq n + \frac{n^2}{\varepsilon}$.

Every solution x to (ATSP LP) for $\mathcal{I} = (G, c)$ corresponds to a solution x' to (ATSP LP) for G', simply by setting $x'_e := x(\delta^+(v))$ for all edges e of P_v. Then

$$\begin{aligned} x'(E(G')) &= \sum_{v \in V(G)} (1 + \bar{c}_v)\, x(\delta^+(v)) \\ &= \sum_{v \in V(G)} \left(1 + \left\lfloor \frac{2c_v}{M} \right\rfloor\right) x(\delta^+(v)) \\ &= \delta \cdot x(E(G)) + \sum_{v \in V(G)} \frac{2c_v}{M}\, x(\delta^+(v)) \\ &= \delta \cdot x(E(G)) + \frac{1}{M} c(x) \end{aligned}$$

for some $\delta \in [0,1]$. Hence

$$c(x) \;\leq\; M \cdot x'(E(G)),$$

and for minimal solutions we have $x(E(G)) \leq n^2$ by Lemma 4.42, which implies $\delta \cdot x(E(G)) \leq n^2 = \varepsilon \frac{2c(V(G))}{M} \leq \varepsilon \frac{c(x)}{M}$ and thus

$$M \cdot x'(E(G)) \;\leq\; (1+\varepsilon)c(x).$$

Because tours are integral LP solutions, and optimum LP solutions and optimum tours can be assumed to be minimal, this completes the proof of (i) and (ii). To prove (iii), observe that contracting the paths P_v in a tour F' yields a tour F as claimed. □

This immediately implies:

Theorem 4.45. *The integrality ratio of* (ATSP LP) *is the same for unweighted and for node-weighted instances. For any constants $\alpha \geq 1$ and $\varepsilon > 0$, there is a polynomial-time $(\alpha + \varepsilon)$-approximation algorithm for node-weighted instances if there is a polynomial-time α-approximation algorithm for unweighted instances.*

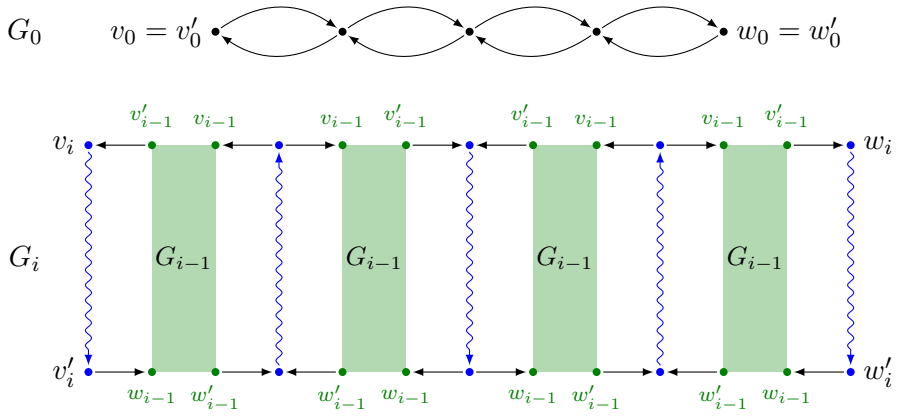

Figure 4.11: Constructing a family of directed graphs with integrality ratio arbitrarily close to 2 for graph ATSP. For a fixed even number $l \geq 4$ we define graphs $G_0, G_1, \ldots$. The graph G_0 consists of a bidirected path of length l. Then we construct G_i from G_{i-1} as in the picture. The picture shows the construction for $l = 4$; in general, there are l copies of the graph G_{i-1} (shown in green). The blue wiggly paths indicate paths of length d_i, where $d_0 = 0$ and $d_i = l^i - d_{i-1} - 2$. Let G'_i be the graph arising from G_i by identifying the blue v_i-v'_i-path with the blue w_i-w'_i-path. Then for $i \to \infty$, the integrality ratio of G'_i converges to $2 - \frac{2}{l}$ (Boyd and Elliott-Magwood [BEM05]).

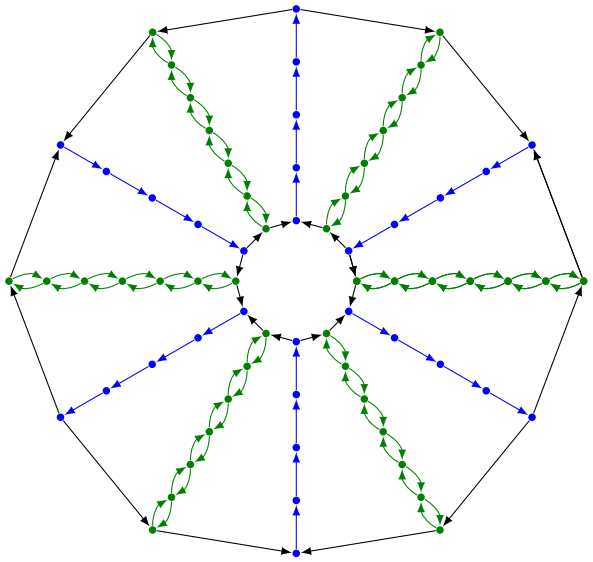

Figure 4.12: The graph G'_1 for $l = 6$. An optimum LP solution has value 1 on the blue edges and value $\frac{1}{2}$ on all other edges and hence we have LP $= |V(G'_1)|$.

Proof. The equality of the integrality ratio for unweighted and for node-weighted instances follows from Lemma 4.44 (i) and (ii). Now suppose we have a polynomial-time α-approximation algorithm for unweighted instances. Then for a node-weighted instance $\mathcal{I} = (G, c)$ we apply Lemma 4.44 with $\varepsilon' = \frac{\varepsilon}{\alpha}$ and apply our α-approximation algorithm to the resulting directed graph G'. Let F' be the resulting tour in G'. By (iii) of Lemma 4.44, this tour corresponds to a tour F in G such that

$$c(F) \;\leq\; M \cdot |F'| \;\leq\; \alpha \cdot M \cdot \mathrm{OPT}(G') \;\leq\; (1+\varepsilon')\alpha \cdot \mathrm{OPT}(\mathcal{I}) \;=\; (\alpha+\varepsilon) \cdot \mathrm{OPT}(\mathcal{I}). \;\square$$

In particular, this implies that the node-weighted instances from [BEM05] can be transformed to unweighted instances whose integrality ratio tends to 2. For convenience we show these instances in Figure 4.11 and Figure 4.12. Figure 4.11 shows the general construction of the family of instances, Figure 4.12 a concrete example. To obtain these instances we have replaced every vertex v in the node-weighted instances with node-weight c_v by a path of length $2c_v - 1$ similar to the proof of Lemma 4.44. So, contracting the blue paths of length d_i in Figure 4.11 and setting the node-weight of the resulting vertex to $\frac{d_i+1}{2}$ and node-weights in G_0 to $\frac{1}{2}$ results in the instances from Boyd and Elliott-Magwood [BEM05]. Then, LP solutions (and tours) in the node-weighted instance correspond to LP solutions (and tours) of the same cost in the unweighted instance. It seems that previously only unweighted instances with integrality ratio at most $\frac{3}{2}$ were known (see e.g. [Got13]).

By splitting an arbitrary vertex into two copies s and t, both inheriting all incident edges, this also yields a family of instances of s-t-path graph ATSP whose integrality ratio tends to 2. We summarize:

Corollary 4.46. *The integrality ratio for instances with $c \equiv 1$ is at least 2, both for* (ATSP LP) *and* (ATSPP LP). $\square$

Chapter 5

The ATSP path LP has constant integrality ratio

In this chapter we prove that the integrality ratio of (ATSPP LP) is constant. In Section 5.1, we reduce s-t-path ATSP to strongly laminar s-t-path ATSP. In contrast to ATSP, here we lose a factor of two in the approximation ratio and integrality ratio. Then in Section 5.2 we use this to prove that we can compute in polynomial time a solution of cost at most $43 + \varepsilon$ times the optimum value of (ATSPP LP) (for every fixed $\varepsilon > 0$). In particular, the integrality ratio of (ATSPP LP) is at most 43.

Feige and Singh [FS07] showed that any α-approximation algorithm for ATSP implies a $(2\alpha + \varepsilon)$-approximation algorithm for s-t-path ATSP (for every fixed $\varepsilon > 0$). In Section 5.3 we give a similar result for the integrality ratio: if ρ is the integrality ratio of (ATSP LP), then the integrality ratio of (ATSPP LP) is at most $4\rho - 3$.

Section 5.3 is based on joint work with Anna Köhne and Jens Vygen. Parts of this chapter have been published in [KTV19].

5.1 Reducing to strongly laminar instances

Similar to ATSP, we now also define strongly laminar instances of s-t-path ATSP.

Definition 5.1. *A* strongly laminar s-t-path ATSP instance *is a tuple* $(G, s, t, \mathcal{L}, x, y)$, *where*

(i) $G = (V, E)$ *is a directed graph;*

(ii) $s, t \in V$*;*

(iii) $\mathcal{L}$ *is a laminar family of subsets of* $V \setminus \{s, t\}$ *such that* $G[U]$ *is strongly connected for all* $U \in \mathcal{L}$*;*

(iv) x *is a feasible solution to* (ATSPP LP) *such that* $x(\delta(U)) = 2$ *for all* $U \in \mathcal{L}$*;*

(v) $y : \mathcal{L} \to \mathbb{R}_{\geq 0}$.

This induces the s-t*-path ATSP instance* (G, c, s, t), *where* c *is the induced weight function defined by* $c(e) := \sum_{U \in \mathcal{L} : e \in \delta(U)} y_U$ $(e \in E)$.

A solution to a strongly laminar s-t-path ATSP instance $\mathcal{I}$ is a solution to its induced s-t-path ATSP instance $\mathcal{I}' = (G, c, s, t)$. Note that for the induced instance

by complementary slackness (Theorem 2.8) x is an optimum solution to (ATSPP LP) and $(0, y)$ is an optimum solution to (ATSPP DUAL). We define $\mathrm{LP}(\mathcal{I}) := c(x) = \sum_{L\in\mathcal{L}} 2y_L = \mathrm{LP}(\mathcal{I}')$ (by Theorem 2.7).

The goal of this section is to give a reduction of general instances of s-t-path ATSP to strongly laminar ones. However, in contrast to ATSP, this reduction will lose a factor 2 in the approximation guarantees and integrality ratios. More precisely, we show the following.

Theorem 5.2. *Let $\alpha \geq 1$. Suppose there is a polynomial-time algorithm that computes a solution of cost at most $\alpha \cdot \mathrm{LP}(\mathcal{I}')$ for every strongly laminar s-t-path ATSP instance $\mathcal{I}'$. Then there is a polynomial-time algorithm that computes a solution of cost at most $(2\alpha - 1) \cdot \mathrm{LP}(\mathcal{I})$ times the optimum value of* (ATSPP LP) *for any instance $\mathcal{I}$ of s-t-path ATSP.*

If ρ is the integrality ratio of (ATSPP LP) *restricted to strongly laminar instances, then the integrality ratio of* (ATSPP LP) *(for general instances) is at most $2\rho - 1$.*

For a graph $G = (V, E)$ and a vector $x \in \mathbb{R}^E$, the *support graph* of x is the graph $(V, \{e : x_e > 0\})$. The following is the main lemma that we need to prove Theorem 5.2.

Lemma 5.3. *Let $\mathcal{I} = (G, c, s, t)$ be an instance of s-t-path ATSP, where G is the support graph of an optimum solution to* (ATSPP LP). *Then there is an optimum solution (a, y) of* (ATSPP DUAL) *with strongly laminar support and $a_s - a_t \leq \mathrm{LP}$.*

Before proving Lemma 5.3 we show that it implies Theorem 5.2.

Proof of Theorem 5.2. Given an instance $\mathcal{I} = (G, c, s, t)$ of s-t-path ATSP and an optimum solution x^* to (ATSPP LP), we may assume that the given graph $G = (V, E)$ is the support graph of x^*; so $x^*_e > 0$ for all $e \in E$. (This is because omitting edges e with $x^*_e = 0$ does not change the optimum LP value and can only increase the cost of an optimum integral solution.) Moreover, let y be an optimum solution to (ATSPP DUAL) with strongly laminar support $\mathcal{L}$ and $a_s - a_t \leq \mathrm{LP}(\mathcal{I})$; this is guaranteed to exist by Lemma 5.3.

Since $x^*_e > 0$ for all $e = (v, w) \in E$, we have $c(e) = \sum_{U:e\in\delta(U)} y_U - a_v + a_w$ by complementary slackness (Theorem 2.8). Then $\mathcal{I}' = (G, s, t, \mathcal{L}, x^*, y)$ is a strongly laminar s-t-path ATSP instance with the induced cost function c^y, where for $e = (v, w) \in E$ we have $c^y(e) = \sum_{U:e\in\delta(U)} y_U = c(v, w) + a_v - a_w$.

Now, for every feasible solution x to (ATSPP LP), we have

$$c(x) = \sum_{e\in E} x_e \cdot c(e) = \sum_{e=(v,w)\in E} x_e \cdot (c^y(e) - a_v + a_w) = c^y(x) - a_s + a_t,$$

where we used that x is an s-t-flow of value 1. Now let $\gamma \geq 1$ and let x and x' be feasible solutions to (ATSPP LP) such that $c^y(x') \leq \gamma \cdot c^y(x)$. Then

$$\begin{aligned} c(x') &= c^y(x') - a_s + a_t \\ &\leq \gamma \cdot c^y(x) - a_s + a_t \\ &= \gamma \cdot (c(x) + a_s - a_t) - a_s + a_t \\ &\leq \gamma \cdot c(x) + (\gamma - 1) \cdot (a_s - a_t) \\ &\leq \gamma \cdot c(x) + (\gamma - 1) \cdot \mathrm{LP}(\mathcal{I}). \end{aligned}$$

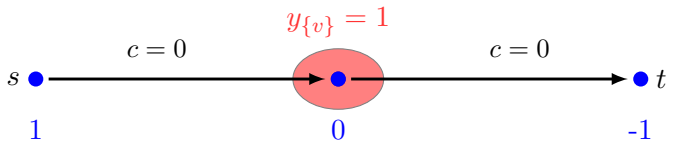

Figure 5.1: Example of an instance with LP = 0 and an optimum dual solution with $a_s - a_t = 2$. The blue numbers below the vertices show the dual variables $a_s = 1$, $a_v = 0$ and $a_t = -1$. Of course, this instance has a different optimum dual solution in which all variables are zero.

This now implies the theorem as follows. For $\gamma = \alpha$, $x = x^*$ an optimum solution of (ATSPP LP) for $\mathcal{I}$, and x' the incidence vector of an integral solution to the strongly laminar instance $\mathcal{I}'$ with $c^y(x') \leq \alpha \cdot c^y(x)$, we obtain the claimed bound on the approximation ratio with respect to (ATSPP LP).

Moreover, for $\gamma = \rho$, $x = x^*$, and x' an optimum integral solution to the strongly laminar instance $\mathcal{I}'$, we obtain the claimed bound on the integrality ratio. □

Figure 5.1 shows that we cannot bound $a_s - a_t$ by LP for an arbitrary optimum dual solution (a, y). Thus, we will work with an optimum dual solution (a, y) with $a_s - a_t$ minimum. Note that this minimum is attained because for every feasible dual solution (a, y) we have $a_s - a_t \geq -\text{LP}$.

First, we give an equivalent characterization of the minimum value of $a_s - a_t$ in any optimum dual solution. This will not be needed to prove Lemma 5.3, but might help to get some intuition.

Lemma 5.4. *Let $\mathcal{I} = (G, c, s, t)$ be an instance of s-t-path ATSP and let $\Delta \geq 0$. Now consider the instance $\mathcal{I}' = (G + e', c, s, t)$, where we add an edge $e' = (t, s)$ with $c(e') := \Delta$. Then $\text{LP}(\mathcal{I}) \geq \text{LP}(\mathcal{I}')$. Moreover, $\text{LP}(\mathcal{I}) = \text{LP}(\mathcal{I}')$ if and only if there exists an optimum solution (a, y) of (ATSPP DUAL) for the instance $\mathcal{I}$ with $a_s - a_t \leq \Delta$.*

Proof. Every feasible solution x of (ATSPP LP) for $\mathcal{I}$ can be extended to a feasible solution of (ATSPP LP) for $\mathcal{I}'$ by setting $x_{e'} := 0$. This shows $\text{LP}(\mathcal{I}) \geq \text{LP}(\mathcal{I}')$.

The dual LPs for the two instances are identical, except for the constraint corresponding to e', which is

$$\Delta = c(e') \geq a_s - a_t + \sum_{\emptyset \neq U \subseteq V \setminus \{s,t\}, e' \in \delta(U)} y_U = a_s - a_t. \tag{5.1}$$

Suppose $\text{LP}(\mathcal{I}) = \text{LP}(\mathcal{I}')$. Let (a, y) be an optimum dual solution for $\mathcal{I}'$. Then, (5.1) is satisfied and (a, y) is also feasible for the dual LP for the instance $\mathcal{I}$. Moreover, since $\text{LP}(\mathcal{I}) = \text{LP}(\mathcal{I}')$, the dual solution (a, y) is also optimum for the instance $\mathcal{I}$.

For the reverse direction, let (a, y) be an optimum solution to (ATSPP DUAL) for the instance $\mathcal{I}$ with $a_s - a_t \leq \Delta$. Then (a, y) satisfies (5.1) and thus is also feasible for (ATSPP DUAL) for $\mathcal{I}'$. Hence, $\text{LP}(\mathcal{I}') \geq \text{LP}(\mathcal{I})$. □

The next lemma describes an important property of optimum dual LP solutions with minimum $a_s - a_t$. We will later show that this property implies $a_s - a_t \leq \text{LP}$.

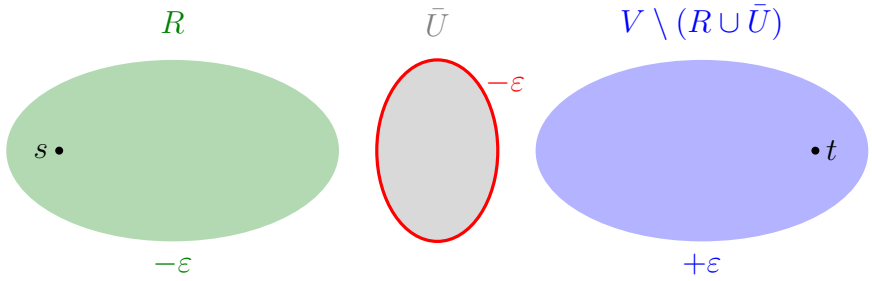

Figure 5.2: Modifying the dual solution in the proof of Lemma 5.5. The green and blue numbers in the bottom indicate the change of the dual node variables. In red the decrease of the variable $y_{\bar{U}}$ is indicated. There is no edge from R to $V \setminus (R \cup \bar{U})$.

Lemma 5.5. *Let (G, c, s, t) be an instance of s-t-path ATSP, where G is the support graph of an optimum solution to* (ATSPP LP)*. Moreover, let (a, y) be an optimum solution of* (ATSPP DUAL) *such that $a_s - a_t$ is minimum. Let $\bar{U} \subseteq V \setminus \{s, t\}$ such that every s-t-path in G enters (and leaves) $\bar{U}$ at least once. Then $y_{\bar{U}} = 0$.*

Proof. Suppose $y_{\bar{U}} > 0$ and let $\varepsilon := y_{\bar{U}}$. Let R be the set of vertices reachable from s in $G - \bar{U}$. We define a dual solution $(\bar{a}, \bar{y})$ as follows:

$$\bar{y}(U) := \begin{cases} y_U - \varepsilon & \text{if } U = \bar{U} \\ y_U & \text{else} \end{cases}$$

$$\bar{a}_v := \begin{cases} a_v - \varepsilon & \text{if } v \in R \\ a_v & \text{if } v \in \bar{U} \\ a_v + \varepsilon & \text{else.} \end{cases}$$

See Figure 5.2 for an illustration. We claim that $(\bar{a}, \bar{y})$ is an optimum (and feasible) solution to (ATSPP DUAL). Note that $t \in V \setminus \left(R \cup \bar{U}\right)$ and thus $\bar{a}_t = a_t + \varepsilon$. Since $s \in R$, we have $\bar{a}_s - \bar{a}_t < a_s - a_t$. Thus, if $(\bar{a}, \bar{y})$ is indeed optimum (and feasible), we obtain a contradiction to our choice of the dual solution (a, y).

First, we observe that $(\bar{a}, \bar{y})$ and (a, y) have the same objective value since

$$\bar{a}_t - \bar{a}_s + \sum_{\emptyset \neq U \subseteq V \setminus \{s,t\}} 2\bar{y}_U \;=\; (a_t + \varepsilon) - (a_s - \varepsilon) + \sum_{\emptyset \neq U \subseteq V \setminus \{s,t\}} 2y_U - 2\varepsilon.$$

By our choice of ε, the vector $\bar{y}$ will be nonnegative. Now consider an edge $e = (v, w) \in E(G)$. We need to show that

$$\bar{a}_w - \bar{a}_v + \sum_{U : e \in \delta(U)} \bar{y}_U \;\leq\; c(e). \tag{5.2}$$

To prove this we will show that

$$\bar{a}_w - a_w - \bar{a}_v + a_v + \sum_{U : e \in \delta(U)} (\bar{y}_U - y_U) \leq 0. \tag{5.3}$$

Since (a, y) is a feasible dual solution, this will imply (5.2). We have

$$\bar{a}_w - a_w := \begin{cases} -\varepsilon & \text{if } w \in R \\ 0 & \text{if } w \in \bar{U} \\ \varepsilon & \text{else,} \end{cases}$$

$$-\bar{a}_v + \bar{a}_v := \begin{cases} \varepsilon & \text{if } v \in R \\ 0 & \text{if } v \in \bar{U} \\ -\varepsilon & \text{else,} \end{cases}$$

$$\sum_{U : e \in \delta(U)} (\bar{y}_U - y_U) := \begin{cases} -\varepsilon & \text{if } (v, w) \in \delta(\bar{U}) \\ 0 & \text{else.} \end{cases}$$

Since $\bar{a}_w - a_w \leq \varepsilon$ and $\bar{\sum}_{U : e \in \delta(U)} (\bar{y}_U - y_U) \leq 0$, it suffices to consider the cases $v \in R$ and $v \in \bar{U}$. If $v \in R$, we have by definition of R, either $w \in R$ or $w \in \bar{U}$. In both cases (5.3) holds, because if $w \in \bar{U}$, we have $(v, w) \in \delta(\bar{U})$. Now let $v \in \bar{U}$. Then if $(v, w) \in \delta(\bar{U})$, we have $\sum_{U : e \in \delta(U)} (\bar{y}_U - y_U) = -\varepsilon$, implying (5.3). Otherwise, $w \in \bar{U}$ and $\bar{a}_w - a_w - \bar{a}_v + a_v = 0$.

This shows that $(\bar{a}, \bar{y})$ is an optimum dual solution and $\bar{a}_s - \bar{a}_t < a_s - a_t$, a contradiction. Hence, $y_{\bar{U}} = 0$. □

We will need the following variant of Menger's Theorem.

Lemma 5.6. *Let G be a directed graph and $s, t \in V(G)$ such that t is reachable from s in G. Let $U \subseteq V(G) \setminus \{s, t\}$ such that for every vertex $u \in U$, there exists an s-t-path in $G - u$. Then there exist two s-t-paths P_1 and P_2 in G such that no vertex $u \in U$ is contained in both P_1 and P_2.*

Proof. We construct a graph G' that arises from G as follows. We split every vertex $u \in U$ into two vertices u^- and u^+ that are connected by an edge $e_u := (u^-, u^+)$. Every edge (v, u) is replaced by an edge (v, u^-) and every edge (u, v) is replaced by an edge (u^+, v). In the graph G' we now define integral edge capacities. Every edge e_u for $u \in U$ has capacity one. All other edges, i.e. all edges corresponding to edges of G, have infinite capacity.

Since for every vertex $u \in U$ there exists an s-t-path in $G - u$, for every $u \in U$ there exists an s-t-path in $G' - e_u$. Thus, the minimum capacity of an s-t-cut in G' is at least two. Hence, by Theorem 2.21 there exists an integral s-t-flow of value two in G' with the defined edge capacities. This flow can be decomposed into two s-t-paths P_1' and P_2' and possibly some cycles. By the choice of the edge capacities, no edge e_u for $u \in U$ occurs in both paths. Since this edge e_u is the only outgoing edge of u^- and the only incoming edge of u^+, an s-t-path containing u^- or u^+ must contain e_u, and at most one of P_1' and P_2' can do so.

Hence, contracting the edges e_u (for $u \in U$) yields two s-t-paths P_1 and P_2 in G such that no vertex $u \in U$ is contained in both P_1 and P_2. □

We will now continue to work with a dual solution (a, y) that minimizes $a_s - a_t$. By Lemma 3.5, we can assume in addition that (a, y) has strongly laminar support. For an illustration of the following lemma, see Figure 5.3.

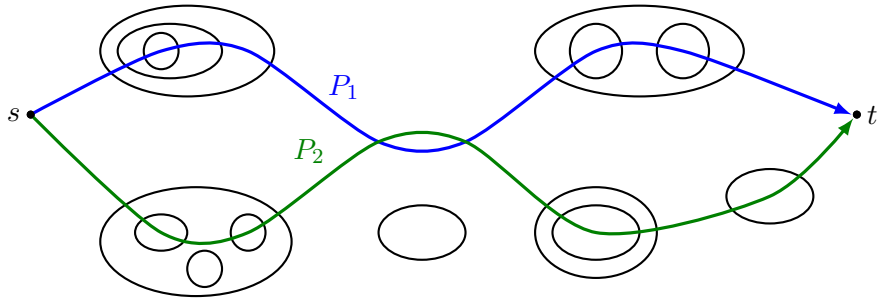

Figure 5.3: The paths P_1 and P_2 as in Lemma 5.7. In black the vertex sets $U \in \text{supp}(y)$ are shown. The paths P_1 and P_2 are not necesarily disjoint but they never both cross the same set U with $y_U > 0$.

Lemma 5.7. *Let (G, c, s, t) be an instance of s-t-path ATSP, where G is the support graph of an optimum solution to* (ATSPP LP)*. Moreover, let (a, y) be an optimum solution to* (ATSPP DUAL) *such that y has strongly laminar support and $a_s - a_t$ is minimum.*

Then G contains two s-t-paths P_1 and P_2 such that for every set $U \in \text{supp}(y)$ we have $|E(P_1) \cap \delta(U)| + |E(P_2) \cap \delta(U)| \leq 2$.

Proof. By Lemma 5.5, for every set $U \in \text{supp}(y)$ there is an s-t-path in G that visits no vertex in U. We contract all maximal sets $U \in \text{supp}(y)$. Using Lemma 5.6, we can find two s-t-paths such that each vertex arising from the contraction of a set $U \in \text{supp}(y)$ is visited by at most one of the two paths.

Now we revert the contraction of the sets $U \in \text{supp}(y)$. We complete the edge sets of the two s-t-paths we found before (which are not necessarily connected anymore after undoing the contraction), to paths P_1 and P_2 with the desired properties. To see that this is possible, let v be the end vertex of the edge e^{in} entering a contracted set $U \in \text{supp}(y)$ and let w be the start vertex of the edge e^{out} leaving U. By Lemma 3.6 there is a nice v-w-path $P_{v,w}$. This path is completely contained in $G[U]$ and enters and leaves every set $U' \in \text{supp}(y)$ with $U' \subsetneq U$ at most once. Moreover, if $e^{\mathsf{in}} \in \delta^-(U')$ for $U' \in \text{supp}(y)$, then $v \in U'$ and the nice v-u-path $P_{v,w}$ does not enter U'. Similarly if $e^{\mathsf{out}} \in \delta^+(U')$ for $U' \in \text{supp}(y)$, then $w \in U'$ and $P_{v,w}$ never leaves U' (and enters it at most once). □

We finally show the main lemma of this section, which we restate here for convenience. Proving this lemma will conclude the reduction to strongly laminar instances (Theorem 5.2).

Lemma 5.3. *Let $\mathcal{I} = (G, c, s, t)$ be an instance of s-t-path ATSP, where G is the support graph of an optimum solution to* (ATSPP LP)*. Then there is an optimum solution (a, y) of* (ATSPP DUAL) *with strongly laminar support and $a_s - a_t \leq \text{LP}$.*

Proof. Let (a, y) be an optimum solution to (ATSPP DUAL) that has strongly laminar support and minimum $a_s - a_t$. Note that such an optimum dual solution exists by Lemma 3.5.

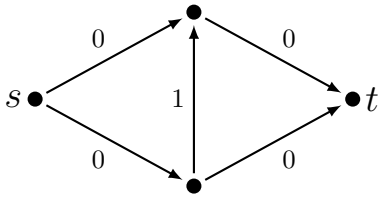

Figure 5.4: Example with no optimum dual solutions with $a_s - a_t < \text{LP}$: The numbers next to the arcs denote their cost. For this instance we have $\text{LP} = 1$. However adding an edge (t, s) with cost $\gamma < 1$ would result in an instance with $\text{LP} = \gamma$. By Lemma 5.4 there cannot be an optimum dual solution where $a_s - a_t < 1 = \text{LP}$.

We define the c^y cost of an edge $e = (v, w)$ to be

$$c^y(e) = \sum_{U: e \in \delta(U)} y_U \;=\; c(e) + a_v - a_w. \tag{5.4}$$

By Lemma 5.7, G contains two s-t-paths P_1 and P_2 such that

$$c^y(E(P_1)) + c^y(E(P_2)) \leq \sum_{\emptyset \neq U \subseteq V \setminus \{s,t\}} 2y_U.$$

Then, using (5.4),

$$\begin{aligned}
0 \;&\leq\; c(E(P_1)) + c(E(P_2)) \\
&=\; \sum_{e=(v,w) \in E(P_1)} (c^y(e) + a_w - a_v) + \sum_{e=(v,w) \in E(P_2)} (c^y(e) + a_w - a_v) \\
&=\; c^y(E(P_1)) - (a_s - a_t) + c^y(E(P_2)) - (a_s - a_t) \\
&\leq\; \sum_{\emptyset \neq U \subseteq V \setminus \{s,t\}} 2y_U - 2(a_s - a_t),
\end{aligned}$$

implying

$$a_s - a_t \;\leq\; \sum_{\emptyset \neq U \subseteq V \setminus \{s,t\}} 2y_U - (a_s - a_t) \;=\; \text{LP}.$$

□

We remark (although we will not need it) that Lemma 5.3 also holds for general instances of s-t-path ATSP. To adapt the proof, work with the subgraph G' of G that contains all edges of G for which the dual constraint is tight. Now G' plays the role of G in the proof, and by choosing ε small enough in the proof of Lemma 5.5 we maintain dual feasibility also for the edges that are not in G'.

By Lemma 5.4, this also shows that adding an edge (t, s) of cost equal to the LP value does not change the value of an optimum LP solution.

The instance in Figure 5.4 shows that the bound $a_s - a_t \leq \text{LP}$ is tight. Note that the bound is also tight for the instance in Figure 5.5 in which $x^*_e > 0$ for all edges e, and in which the integrality ratio is arbitrarily close to the best known lower bound of 2.

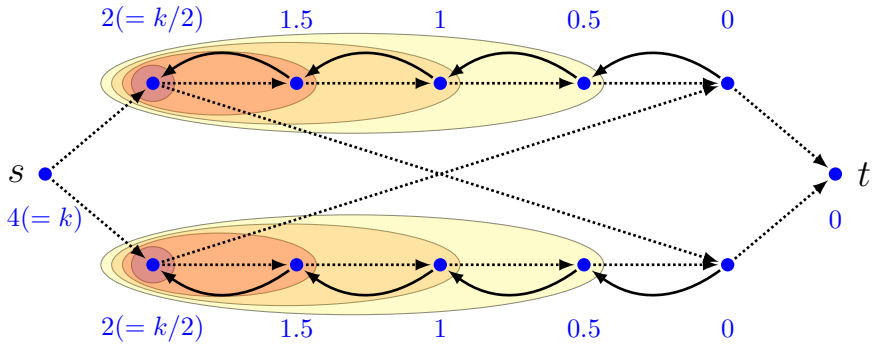

Figure 5.5: Example due to Friggstad, Gupta, and Singh [FGM16] with integrality ratio approaching 2 as the number of vertices increases. Setting $x_e := \frac{1}{2}$ for all shown edges defines a feasible solution of (ATSPP LP). If the $2k$ curved edges have cost 1 and the dotted edges have cost 0, we have LP $= c(x) = k$, but any s-t-tour costs at least $2k - 1$. (In the figure, $k = 4$.) Setting $y_U = \frac{1}{2}$ for the vertex sets indicated by the ellipses and a_v as shown in blue defines an optimum solution of (ATSPP DUAL).

5.2 Bounding the integrality ratio for strongly laminar instances

In this section we show that the integrality ratio of (ATSPP LP) is at most 43. The following lemma describes the structure of the strongly connected components of the support graph G of an optimum solution to (ATSPP LP).

Lemma 5.8. *Let (G, c, s, t) be an instance of s-t-path ATSP, where G is the support graph of an optimum solution x^* to* (ATSPP LP)*. Let (a, y) be an optimum solution of* (ATSPP DUAL)*.*

Then the strongly connected components of V can be numbered $V_1, \dots, V_l$ such that

- *$s \in V_1$ and $t \in V_l$,*
- *$\emptyset = \delta^-(V_1) = \delta^+(V_l)$, and*
- *$\delta^+(V_i) = \delta^-(V_{i+1})$ and $x^*(\delta^+(V_i)) = 1$ for all $i \in \{1, \dots, l-1\}$.*

Proof. Let $V_1, \dots, V_l$ be a topological order of the strongly connected components of G. Then $\emptyset = \delta^-(V_1) = \delta^+(V_l)$. Moreover, we have $s \in V_1$ and $t \in V_l$ because for every vertex v in G, v is reachable from s, and t is reachable from v.

We now show $\delta^+(V_i) = \delta^-(V_{i+1})$ and $x^*(\delta^+(V_i)) = 1$ for $i = 1, \dots, l-1$ by induction on i. For $i = 1$, we have $x^*(\delta^+(V_i)) = 1$, because $\delta^-(V_1) = \emptyset$, $s \in V_1$, and $t \notin V_1$. This implies $1 \le x^*(\delta^-(V_2)) \le x^*(\delta^+(V_1)) = 1$. Since, G is the support graph of x^*, this implies $\delta^+(V_1) = \delta^-(V_2)$.

Now let $i \in \{2, \dots, l-1\}$. We have $\delta^-(V_{i+1}) \subseteq \delta^+(V_1) \cup \dots \cup \delta^+(V_i)$ because $V_1, \dots, V_l$ is a topological order. Since, by induction, we have $\delta^+(V_j) = \delta^-(V_{j+1})$ for $j = 1, \dots, i-1$, this implies $\delta^-(V_{i+1}) \subseteq \delta^+(V_i)$. Then

$$1 = x^*(\delta^+(V_{i-1})) = x^*(\delta^-(V_i)) = x^*(\delta^+(V_i)) \ge x^*(\delta^-(V_{i+1})) \ge 1,$$

where we used $x^*(\delta^-(V_i)) = x^*(\delta^+(V_i))$ because V_i contains neither s nor t. Thus, $x^*(\delta^+(V_i)) = x^*(\delta^-(V_{i+1})) = 1$ and, since G is the support graph of x^*, this also implies $\delta^+(V_i) = \delta^-(V_{i+1})$. □

Theorem 5.9. *For every $\varepsilon > 0$ there is a polynomial-time algorithm that computes for every instance (G, c, s, t) of s-t-path ATSP a solution of cost at most $43 + \varepsilon$ times the cost of an optimum solution to* (ATSPP LP).

Proof. We show that for every $\varepsilon > 0$ there is a polynomial-time algorithm that computes for every strongly laminar instance $\mathcal{I} = (G, s, t, \mathcal{L}, x, y)$ of s-t-path ATSP a solution of cost at most $(22 + \varepsilon) \cdot \mathrm{LP}(\mathcal{I})$. This is sufficient by Theorem 5.2.

We define a strongly laminar ATSP instance $\mathcal{I}' = (G', \mathcal{L}', x', y')$. Let G' arise from G by adding a new vertex v and two edges (t, v) and (v, s). Then we set $x'_{(t,v)} = x'_{(v,s)} = 1$ and $x'_e = x_e$ for all $e \in E(G)$. Then x' is a feasible solution to (ATSP LP). Let $V_1, \ldots, V_l$ be the vertex sets of the strongly connected components of G numbered as in Lemma 5.8. We define $\mathcal{L}' = \mathcal{L} \cup \{V_1, \ldots, V_l\}$. Then we set $y'_{V_i} = 0$ for all $i \in \{1, \ldots, l\}$. For $L \in \mathcal{L}$ we set $y'_L = y_L$.

We claim that $\mathcal{I}'$ is a strongly laminar ATSP instance. First we observe that $G'[L]$ is strongly connected for all $L \in \mathcal{L}'$ by construction and G' is strongly connected by Lemma 5.8. Moreover, by Lemma 5.8 we have $x'(\delta(V_i)) = 2$ for all $i \in \{1, \ldots, l\}$. Furthermore, for $L \in \mathcal{L}$ we have $s, t \notin L$ and hence $x'(\delta(L)) = x(\delta(L)) = 2$. It remains to show that $\mathcal{L}'$ is a laminar family. We have that $\mathcal{L}$ is a laminar family and the sets $V_1, \ldots, V_l$ are pairwise disjoint. Suppose $\mathcal{L}'$ is not a laminar family. Then there is a set $L \in \mathcal{L}$ and an index $i \in \{1, \ldots, l\}$ such that $L \setminus V_i \neq \emptyset$, $V_i \setminus L \neq \emptyset$ and $V_i \cap L \neq \emptyset$. Let $u \in V_i \cap L$ and $w \in L \setminus V_i$. Because $G[L]$ is strongly connected, G contains both a u-w-path and a w-u-path, contradicting the fact that V_i is the vertex set of a strongly connected component of G. This shows that $\mathcal{I}'$ is a strongly laminar ATSP instance.

Theorem 4.9 yields a $(3, 2, 1)$-algorithm for Subtour Cover and by Theorem 4.28 this implies that there is a polynomial-time $(2, 14 + \varepsilon)$-algorithm for vertebrate pairs. By Lemma 4.3 and Lemma 4.5, we can compute in polynomial time a tour F_i in $G[V_i]$ of cost at most

$$c(F_i) \;\leq\; 6 \cdot \mathrm{value}(V_i) + (16 + \varepsilon) \cdot (\mathrm{value}(V_i) - D_{V_i}). \tag{5.5}$$

for every $i \in \{1, \ldots, l\}$. By Lemma 5.8, any s-t-path in G visits every strongly connected component $G[V_i]$ of G exactly once. Hence, the union of the edge set a shortest s-t-path P and $F_1 \,\dot\cup \ldots, \dot\cup\, F_l$ is an s-t-tour F. Using the definition of D_{V_i} (see Section 4.2), we get $c(E(P)) \leq \sum_{i=1}^{l} D_{V_i}$. Therefore, using (5.5) we obtain

$$\begin{aligned} c(F) &\leq \sum_{i=1}^{l} (D_{V_i} + 6 \cdot \mathrm{value}(V_i) + (16 + \varepsilon) \cdot (\mathrm{value}(V_i) - D_{V_i})) \\ &\leq \sum_{i=1}^{l} (22 + \varepsilon) \cdot \mathrm{value}(V_i) \\ &= (22 + \varepsilon) \cdot \mathrm{LP}(\mathcal{I}). \end{aligned}$$

Here we used that $\mathcal{I}$ is a strongly laminar s-t-path ATSP instance, implying that $(0, y)$ is an optimum solution to (ATSPP DUAL) and hence $\mathrm{LP}(\mathcal{I}) = \sum_{L \in \mathcal{L}} 2y_L = \sum_{i=1}^{l} \mathrm{value}(V_i)$. □

Corollary 5.10. *The integrality ratio of* (ATSPP LP) *is at most* 43.

Proof. Suppose there is an instance $\mathcal{I}$ of s-t-path ATSP where $\frac{\mathrm{OPT}(\mathcal{I})}{\mathrm{LP}(\mathcal{I})} > 43$. Then there exists $\varepsilon > 0$ such that $\frac{\mathrm{OPT}(\mathcal{I})}{\mathrm{LP}(\mathcal{I})} > 43 + \varepsilon$. By Theorem 5.9 we can compute an integral solution for $\mathcal{I}$ with cost at most $(43 + \varepsilon) \cdot \mathrm{LP}(\mathcal{I}) < \mathrm{OPT}(\mathcal{I})$, a contradiction. □

5.3 Blackbox reduction to ATSP

We first transform an instance and a solution to (ATSPP LP) to an instance and a solution to (ATSP LP) and work with an integral solution of this ATSP instance. The following lemma is essentially due to Feige and Singh [FS07]. For completeness, we prove it here again for our setting.

Lemma 5.11. *Let ρ be the integrality ratio of* (ATSP LP)*. Suppose the following condition holds for every strongly laminar instance $\mathcal{I} = (G, s, t, \mathcal{L}, x, y)$ of s-t-path ATSP: if there are two s-t-walks P_1 and P_2 of total cost L in G, there is a single s-t-walk P in G which has cost $c(P) \leq L + \mathrm{LP}$ and contains all vertices of P_1 and P_2.*

Then the integrality ratio of (ATSPP LP) *restricted to strongly laminar instances is at most $2\rho - 1$.*

Proof. Let $\mathcal{I} = (G, s, t, \mathcal{L}, x, y)$ be a strongly laminar instance of s-t-path ATSP. Consider the strongly laminar instance $\mathcal{I}' = (G', \mathcal{L}', x, y)$ of ATSP that arises from $\mathcal{I}$ as follows. We add a new vertex v to G and two edges (t, v) and (v, s); we set $x_{(t,v)} = x_{(v,s)} = 1$. Moreover we add the set $\{v\}$ to $\mathcal{L}$ and set $y_{\{v\}} = \frac{1}{2} \cdot \mathrm{LP}(\mathcal{I})$. Then $\mathrm{LP}(\mathcal{I}') = 2 \cdot \mathrm{LP}(\mathcal{I})$. Hence there is an ATSP solution for $\mathcal{I}'$ with cost at most $2\rho \cdot \mathrm{LP}(\mathcal{I})$. Let R be such a solution. Then R has to use (t, v) and (v, s) at least once, since it has to visit v. By deleting all copies of (t, v) and (v, s) from R, we get $k > 0$ s-t-walks in G with total cost at most $2\rho \cdot \mathrm{LP}(\mathcal{I}) - k \cdot \mathrm{LP}(\mathcal{I})$ such that every vertex of G is visited by at least one of them. As long as $k > 1$, by our assumption we can replace two of the s-t-walks by a single one, increasing the cost by at most $\mathrm{LP}(\mathcal{I})$ and decreasing k by one. We end up with a single s-t-walk P with cost $c(P) \leq 2\rho \cdot \mathrm{LP}(\mathcal{I}) - \mathrm{LP}(\mathcal{I})$ in G, which contains every vertex of G. This walk is an s-t-path ATSP solution for $\mathcal{I}$ and thus the integrality ratio of (ATSPP LP) restricted to strongly laminar instances is at most $2\rho - 1$ as proposed. □

The following procedure is similar to one step ("inducing on a tight set") of the approximation algorithm for ATSP by Svensson, Tarnavski, and Végh [STV18a].

Lemma 5.12. *Let $(G, s, t, \mathcal{L}, x, y)$ be a strongly laminar instance of s-t-path ATSP. Let P_1 and P_2 be s-t-walks in G with total cost L. Then there is a single s-t-walk P in G which contains every vertex of P_1 and P_2 and has cost at most $L + \mathrm{LP}$.*

Proof. Let $V_1, \ldots, V_l$ be the vertex sets of the strongly connected components of G in their topological order, which is unique by Lemma 5.8. Let P_i^j be the section of P_i that visits vertices in V_j (for $i = 1, 2$ and $j = 1, \ldots, l$). By Lemma 5.8, none of these sections of P_i is empty. (Such a section might consist of a single vertex and no edges, but it has to contain at least one vertex.)

Because $G[L]$ is strongly connected for every $L \in \mathcal{L}$ and the sets V_i for $i \in \{1, \ldots, l\}$ are the vertex sets of the strongly connected components of G, we have that $\mathcal{L}' \coloneqq$

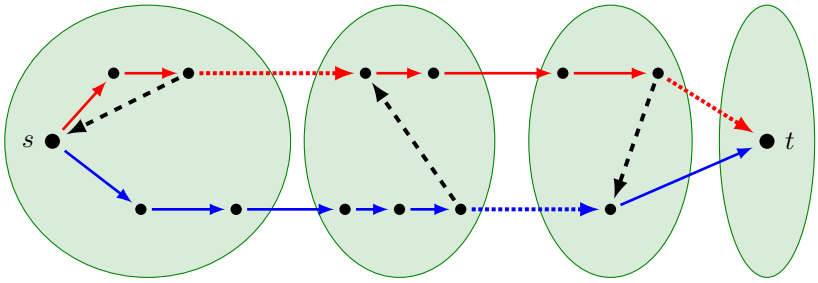

Figure 5.6: Construction of P. The s-t-walks P_1 and P_2 are shown with solid and dotted lines. (Here, P_1 is the upper red walk and P_2 is shown in blue at the bottom.) The vertex sets $V_1, \dots, V_l$ of the strongly connected components are indicated by the green ellipses. The red and blue solid edges of the walks P_i that are those that are used in the walk P. The dashed black arrows indicate the paths R^j.

$\mathcal{L} \cup \{V_1, \dots, V_l\}$ is a laminar family. Moreover, $G[L]$ is strongly connected for every set $L \in \mathcal{L}'$. Therefore, we can apply Lemma 3.6 to the laminar family $\mathcal{L}'$.

We consider nice paths R^j in G for $j = 1, \dots, l$ that we will use to connect the walks P_1^j and P_2^j to a single walk visiting all vertices in V_j. See Figure 5.6. If j is odd, let R^j be a nice path from the last vertex of P_1^j to the first vertex of P_2^j. If j is even, let R^j be a nice path from the last vertex of P_2^j to the first vertex of P_1^j. (Such paths exist by Lemma 3.6.)

We now construct our s-t-walk P that will visit every vertex of P_1 and P_2. We start by setting $P = s$ and then add for $j = 1, \dots, l$ all the vertices in V_j to P as follows. If j is odd, we append P_1^j and R^j and then P_2^j. If j is even, we append P_2^j and R^j and then P_1^j. Note that when moving from one connected component V_j to the next component V_{j+1}, we use an edge from either P_1 (if j is even) or P_2 (if j is odd). Then P is, indeed, an s-t-walk in G and contains every vertex of P_1 and P_2.

We now bound the cost of the walk P. It is constructed from pieces of P_1 and P_2 and the paths R^j. Each of the paths R^j can only contain vertices of V_j. Two paths R^j and $R^{j'}$, such that $j \neq j'$, can never both enter or both leave the same element of the laminar family $\mathcal{L}$: otherwise they would contain vertices of the same strongly connected component of G. Thus every element of $\mathcal{L}$ is entered at most once and left at most once on all the paths R^j used in the construction of P, and the total cost of these paths is at most $\sum_U 2y_U = \text{LP}$. Consequently, we have $c(P) \leq L + \text{LP}$ as claimed. □

We will now prove the main result of this section.

Theorem 5.13. *Let ρ be the integrality ratio of* (ATSP LP)*. Then the integrality ratio of* (ATSPP LP) *is at most* $4\rho - 3$.

Proof. By Lemma 5.12, the condition of Lemma 5.11 is satisfied. Hence, the integrality ratio of (ATSPP LP) restricted to strongly laminar s-t-path ATSP instances is at most $2\rho - 1$. By Theorem 5.2 this implies that the integrality ratio of (ATSPP LP) is at most $4\rho - 3$. □

Finally, we will prove a better bound for unit-weight instances.

Theorem 5.14. *Let ρ be the integrality ratio of* (ATSP LP) *for graph ATSP. Then the integrality ratio of* (ATSPP LP) *for s-t-path graph ATSP is at most $2\rho - 1$.*

Suppose there exists an α-approximation algorithm for graph ATSP. Then there is an $(2\alpha - 1)$-approximation algorithm for s-t-path graph ATSP.

Proof. First we show how to modify the proof of Lemma 5.11 for unit-weighted instances. For an instance $\mathcal{I} = (G, s, t)$ of s-t-path graph ATSP, let the instance $\mathcal{I}' = G'$ of graph ATSP result from $\mathcal{I}$ by adding a t-s-path of length $n - 1$, where all inner vertices of this path are new vertices not contained in G; here $n := |V(G)|$. Then $\mathrm{LP}(\mathcal{I}') \leq \mathrm{LP}(\mathcal{I}) + (n-1)$ and $\mathrm{OPT}(\mathcal{I}') \leq \mathrm{OPT}(\mathcal{I}) + (n-1)$. Continuing with the instance $\mathcal{I}'$ as in the proof of Lemma 5.11 yields the following: It suffices to show that for instances of s-t-path graph ATSP, we can merge any two s-t-walks to a single s-t-walk P as in Lemma 5.12, but with $c(P) \leq L + (n-1)$.

We construct P as in the proof of Lemma 5.12, where we choose the paths R^j to be shortest paths in $G[V_j]$. Again we first bound the cost of the paths R^j. For $1 \leq j \leq l$ each vertex in V_j can only be contained in the path R^j (and not in a path $R^{j'}$ for $j' \neq j$). Hence every vertex can be used in at most one path R^j. Therefore, the total cost of all paths R^j can be bounded from above by $n-1$. This shows $c(P) \leq L+(n-1)$ and completes the proof. □

Corollary 5.15. *The integrality ratio of* (ATSPP LP) *for s-t-path graph ATSP is at most 25. For every $\varepsilon > 0$ there is a $(25 + \varepsilon)$-approximation algorithm for the s-t-path graph ATSP.*

Proof. By Theorem 4.43, there is a $(13 + \frac{\varepsilon}{2})$-approximation algorithm for graph ATSP. Moreover, Svensson [Sve15] showed that the integrality ratio of (ATSP LP) for graph ATSP is at most 13. Applying Theorem 5.14 completes the proof. □

Part II

The symmetric traveling salesman problem and its path version

Chapter 6

TSP and s-t-path TSP

In this chapter we provide an introduction to TSP and s-t-path TSP and their unit-weight special cases graph TSP and s-t-path graph TSP. We introduce the classical LP relaxations and give an overview of prior work and known approximation algorithms.

6.1 Problem definitions

An instance of TSP consists of a connected undirected graph $G = (V, E)$ and nonnegative edge costs $c : E \to \mathbb{R}_{\geq 0}$. The task is to find a closed walk in G that visits every vertex at least once and has minimum cost. A *closed walk* in G is a sequence $v_0, v_1, \ldots, v_k$ with $v_0 = v_k$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \{1, \ldots, k\}$. We are looking for such a sequence that contains every vertex at least once and minimizes $\sum_{i=1}^{k} c(v_{i-1}, v_i)$. Equivalently, we can require that the sequence contains every vertex exactly once, assume that G is a complete graph, and assume that the edge costs c fulfill the triangle inequality, i.e. $c(u, v) + c(v, w) \geq c(u, w)$ for all $u, v, w \in V$.

To see that this is equivalent, we consider the metric closure of (G, c). In the metric closure, we have a complete graph and the edge cost fulfill the triangle inequality. Every closed walk in G is a closed walk in the metric closure of at most the same cost. Moreover, having a closed walk in the metric closure that visits every vertex at least once, we can replace every edge by a shortest path between its endpoints to obtain such a walk in G without increasing the cost. In the metric closure, we can require that every vertex is visited at exactly once because we can otherwise shortcut the tour, i.e. we can skip all but one of the visits; due to the triangle inequality this does not increase the cost.

Another equivalent definition of TSP is the following. Given a connected undirected graph $G = (V, E)$ and nonnegative edge cost $c : E \to \mathbb{R}_{\geq 0}$, find a minimum cost multi-set F of edges such that (V, F) is connected and Eulerian, i.e. $|\delta_F(v)|$ is even for every vertex $v \in V$. We call such a multi-set F a *tour*. Clearly, if F is the multi-set of edges of a closed walk that visits all vertices, then (V, F) is connected and Eulerian. On the other hand, if F is a tour, we can find a closed Eulerian walk in (V, F) that visits all vertices by Theorem 2.2. Therefore, this is indeed an equivalent definition of TSP, which is often more convenient when designing approximation algorithms for TSP.

The s-t-path TSP is the generalization of TSP where the start and the end of the tour are given, but are not necessarily identical. More precisely, we are given a connected undirected graph $G = (V, E)$, nonnegative edge cost $c : E \to \mathbb{R}_{\geq 0}$, a start

vertex $s \in V$ and an end vertex $t \in V$. The task is to find an s-t-walk that visits all the vertices and has minimum cost. In other words, we want to find a sequence $v_0, v_1, \ldots, v_k$ with $s = v_0$, $t = v_k$, and $\{v_{i-1}, v_i\} \in E$ for all $i \in \{1, \ldots, k\}$ such that every vertex appears at least once in this sequence and we minimize $\sum_{i=1}^{k} c(v_{i-1}, v_i)$. TSP is the special case where $s = t$.

Similar to TSP, we can also in the path version assume that G is the complete graph on the vertex set V and that the edge costs fulfill the triangle inequality; then we can require the sequence to visit every vertex exactly once. Another equivalent formulation is to ask for a multi-set F of edges of the (not necessarily complete) graph G such that (V, F) is connected and all vertex degrees have "the correct parity", i.e. for $s \neq t$ the vertices s and t should have odd degree while all other vertices should have even degree. We call such a multi-set F where (V, F) is connected and $\text{odd}(F) = \{s\} \triangle \{t\}$ an *s-t-tour*.

6.2 The standard LP relaxations

The standard LP relaxation for TSP is the following:

$$\begin{array}{lrcll} \min & c(x) & & & \\ \text{s.t.} & x(\delta(U)) & \geq & 2 & \text{for } \emptyset \neq U \subsetneq V \\ & x_e & \geq & 0 & \text{for } e \in E. \end{array} \tag{TSP LP}$$

If we consider the formulation of TSP where G is a complete graph and the edge cost fulfill the triangle inequality (which we can always achieve by taking the metric closure), we can also add degree constraints for all the vertices:

$$\begin{array}{lrcll} \min & c(x) & & & \\ \text{s.t.} & x(\delta(v)) & = & 2 & \text{for } v \in V \\ & x(\delta(U)) & \geq & 2 & \text{for } \emptyset \neq U \subsetneq V \\ & x_e & \geq & 0 & \text{for } e \in E = \binom{V}{2}. \end{array} \tag{6.1}$$

However, adding degree constraints does not strengthen the relaxation. Using the splitting-off technique due to Lovász' [Lov76], one can show that the LP (6.1) is equivalent to (TSP LP) in the following sense. For every feasible solution x of (TSP LP), there is a feasible solution x' of (6.1) with $c(x') \leq c(x)$. Of course, every feasible solution of (6.1) is also feasible for (TSP LP). Hence, optimum solutions of (TSP LP) and (6.1) have the same cost and both linear programs have the same integrality ratio. These observations are due to Cunningham [MMP90] and Goemans and Bertsimas [GB93].

The linear programming relaxation (6.1) was first proposed by Dantzig, Fulkerson, and Johnson [DFJ54] and is often called *Subtour Elimination LP* or *Held-Karp relaxation*. Despite the exponential number of constraints (TSP LP) and (6.1) can be solved in polynomial time. This can be done either via an extended formulation of polynomial size or by using that the separation problem is equivalent to a minimum-cut problem.

The best known lower bound on the integrality ratio of (TSP LP) is $\frac{4}{3}$. Figure 6.1 shows a well-known family of instances with integrality ratio arbitrarily close to $\frac{4}{3}$. In

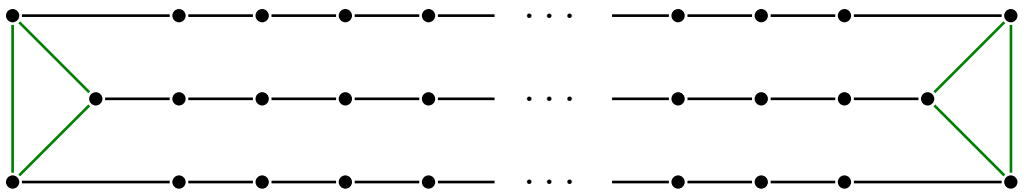

Figure 6.1: A family of instances showing that the integrality ratio of (TSP LP) is at least $\frac{4}{3}$: here the cost of every edge is one. In an optimum LP solution we have $x_e = 1$ for every black edge e and $x_e = \frac{1}{2}$ for every green edge e. Then LP $= n := |V|$, but an optimum tour has $\frac{4}{3}n - 2$ edges.

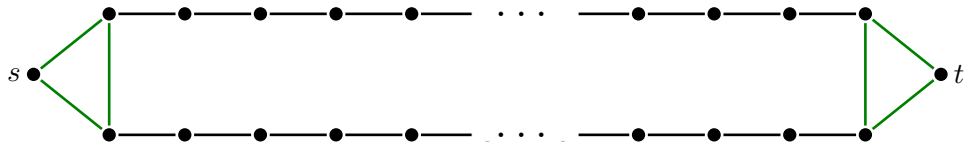

Figure 6.2: A family of instances showing that the integrality ratio of (TSPP LP) is at least $\frac{3}{2}$: here the cost of every edge is one. In an optimum LP solution we have $x_e = 1$ for every black edge e and $x_e = \frac{1}{2}$ for every green edge e. Then LP $= n - 1$, but an optimum s-t-tour has $\frac{3}{2}n - 4$ edges.

these instances the edges have unit weights, i.e. $c(e) = 1$ for all $e \in E$. This is one important reason for studying the unit-weight special case of TSP, the *graph TSP*.

For the s-t-path TSP the standard LP relaxation is the following:

$$\begin{aligned} \min\ & c(x) \\ \text{s.t.}\quad & x(\delta(U)) \geq 2 \quad \text{for } \emptyset \neq U \subseteq V \setminus \{s,t\} \\ & x(\delta(U)) \geq 1 \quad \text{for } \{s\} \subseteq U \subseteq V \setminus \{t\} \\ & x_e \geq 0 \quad \text{for } e \in E \end{aligned} \tag{TSPP LP}$$

We can solve this LP in polynomial time similar to (TSP LP). If G is a complete graph and c fulfills the triangle inequality, we can add degree constraints:

$$\begin{aligned} \min\ & c(x) \\ \text{s.t.}\quad & x(\delta(v)) = \begin{cases} 1, & \text{if } v \in \{s,t\} \text{ and } s \neq t \\ 2, & \text{if } v \in V \setminus \{s,t\} \text{ or } v = s = t \end{cases} \\ & x(\delta(U)) \geq 2 \quad \text{for } \emptyset \neq U \subseteq V \setminus \{s,t\} \\ & x(\delta(U)) \geq 1 \quad \text{for } \{s\} \subseteq U \subseteq V \setminus \{t\} \\ & x_e \geq 0 \quad \text{for } e \in E = \tbinom{V}{2}. \end{aligned} \tag{6.2}$$

However, as for (TSP LP) adding degree constraints does not lead to a stronger relaxation.

The best known lower bound on the integrality ratio of (TSPP LP) is $\frac{3}{2}$, which can be shown by the well-known family of examples shown in Figure 6.2. In this family of instances we have $c(e) = 1$ for every edge $e \in E$. This raised attention also for the unit-weight special case of the s-t-path TSP, the *s-t-path graph TSP*.

6.3 Christofides' algorithm

The best known approximation algorithm for TSP is Christofides' algorithm [Chr76] from 1976, which was independently found by Serdjukov [Ser78]. The algorithm works as follows. First we compute a minimum cost spanning tree (V, S). Now let $T_S \coloneqq \text{odd}(S)$ be the set of vertices whose degree in S is odd and therefore has the wrong parity. To correct these parities, we compute a minimum cost T_S-join J, i.e. a set J such that $T_S = \text{odd}(J)$. Then $(V, S \dot\cup J)$ is connected and Eulerian. Hence we can find a closed Eulerian walk that visits all vertices.

This algorithm has approximation ratio $\frac{3}{2}$. In fact it computes a solution of cost at most $\frac{3}{2}$ times the value of (TSP LP), or equivalently the value of (6.1) in the metric closure, as shown by Wolsey [Wol80].

Theorem 6.1. *For every instance (G, c) of TSP, Christofides' algorithm computes a solution of cost at most $\frac{3}{2}$ times the value of* (TSP LP)*. In particular, the integrality ratio of* (TSP LP) *is at most $\frac{3}{2}$.*

Proof. Consider an optimum solution x^* of the linear program (6.1) in the metric closure. We first bound the cost of the spanning tree S. Let $G = (V, E)$ and let $n := |V|$ denote the number of vertices. The vector $\frac{n-1}{n} \cdot x^*$ is contained in the spanning tree polytope of the complete graph $(V, \bar E)$

$$\left\{x \in \mathbb{R}^{\bar E}_{\geq 0} : x(\bar E) = n - 1,\ x(\bar E[U]) \leq |U| - 1 \ \text{ for } \emptyset \neq U \subseteq V\right\} \tag{6.3}$$

because for every set $\emptyset \neq U \subsetneq V$,

$$2 \cdot x^*(\bar E[U]) = \sum_{v \in U} x^*(\delta(v)) - x^*(\delta(U)) = 2|U| - x^*(\delta(U)) \leq 2(|U| - 1)$$

and similarly $x^*(\bar E) = n$, implying $\frac{n-1}{n} \cdot x^*(\bar E) = n - 1$. Therefore, $\frac{n-1}{n} \cdot x^*$ is a convex combination of incidence vectors of spanning trees (in the complete graph $(V, \bar E)$), implying $c(S) \leq \frac{n-1}{n} \cdot c(x^*) \leq c(x^*)$. Here we used that a minimum cost spanning tree in G also is a minimum cost spanning tree in the metric closure.

To bound the cost of the T_S-join J, we observe that the vector $\frac{1}{2}x^*$ is contained in the T_S-join polyhedron

$$\left\{x \in \mathbb{R}^{\bar E}_{\geq 0} : x(\delta(U)) \geq 1 \ \text{ for } U \subseteq V \text{ with } |U \cap T_S| \text{ odd}\right\}; \tag{6.4}$$

see Theorem 2.16. Therefore, we have $c(J) \leq \frac{1}{2}c(x^*)$, where we used that every minimum cost T-join in G also is a minimum cost T-join in the metric closure.

We conclude that the cost of the tour computed by Christofides' algorithm is at most $c(S) + c(J) \leq \frac{3}{2} \cdot c(x^*)$. Because the two LP relaxations with and without degree constraints are equivalent, $c(x^*)$ is the optimum value of (TSP LP). □

Figure 6.3: A family of instances due to Hoogeveen [Hoo91] showing that the approximation ratio of Christofides' algorithm for the s-t-path TSP is no better than $\frac{5}{3}$. The instances have unit weights, i.e. $c(e) = 1$ for every edge e. The thick red edges show a minimum spanning tree S of cost $n-1$. Squares show the set T_S. Then the cost of a minimum T_S-join is $\frac{2}{3}(n-1)$. Therefore, the solution computed by Christofides' algorithm has cost $\frac{5}{3}(n-1)$, whereas an optimum s-t-tour has cost $n-1$.

See [SW90] for an alternate proof of Theorem 6.1.

Christofides' algorithm can be generalized to the s-t-path TSP as follows. We again compute a minimum spanning tree (V, S) and denote by T_S the set of vertices whose degree has the "wrong parity". More precisely, for $s \neq t$ we set $T_S := \text{odd}(S) \triangle \{s, t\}$. Then we compute a cheapest T_S-join J and we can find an Eulerian s-t-walk in $(V, S \dot\cup J)$ that visits all vertices. Hoogeveen [Hoo91] showed that this algorithm has approximation ratio $\frac{5}{3}$. In fact it computes a solution of cost at most $\frac{5}{3}$ times the optimum value of (TSPP LP) as shown by An, Kleinberg, and Shmoys [AKS15].

Theorem 6.2. *For very instance (G, c, s, t) of the s-t-path TSP, Christofides' algorithm computes a solution of cost at most $\frac{5}{3}$ times the value of* (TSPP LP)*. In particular, the integrality ratio of* (TSPP LP) *is at most $\frac{5}{3}$.*

Proof. We may assume $s \neq t$. Otherwise, the claimed statement follows from Theorem 6.1. Let x^* be an optimum solution of (6.2) in the metric closure.

Then x^* is contained in the spanning tree polytope (6.3) of the complete graph $(V, \bar{E})$. To see this consider a set $\emptyset \neq U \subsetneq V$. Then $2 \cdot x^*(\bar{E}[U]) = \sum_{v \in U} x^*(\delta(v)) - x^*(\delta(U))$. If $\delta(U)$ is an s-t-cut, we have $\sum_{v \in U} x^*(\delta(v)) = 2|U| - 1$ and $x^*(\delta(U)) \geq 1$. Otherwise, $\sum_{v \in U} x^*(\delta(v)) \leq 2|U|$ and $x^*(\delta(U)) \geq 2$. In any case we conclude $x^*(\bar{E}[U]) \leq |U| - 1$. Moreover, $x^*(E) = n - 1$. Therefore x^* is contained in the spanning tree polytope of the complete graph $(V, \bar{E})$. This implies that a cheapest spanning tree in the metric closure, and hence also in G, has cost at most $c(x^*)$. So $c(S) \leq c(x^*)$.

To bound the cost of the T_S-join J, we show that the vector $y := \frac{1}{3} \cdot x^* + \frac{1}{3} \cdot \chi^S$ is contained in the T_S-join polyhedron (6.4). Then the cost of the tour computed by Christofides' algorithm is $c(S) + c(J) \leq c(S) + c(y) \leq \frac{4}{3} c(S) + \frac{1}{3} \cdot c(x^*) \leq \frac{5}{3} c(x^*)$. Since $c(x^*)$ is the value of (6.2) in the metric closure and therefore also the value of (TSPP LP), this implies the statement of the theorem.

It remains to show that the vector y is contained in the T_S-join polyhedron. To prove this let $\emptyset \neq U \subsetneq V$ with $y(\delta(U)) < 1$. We show that $|T_S \cap U|$ is even. Since $|S \cap \delta(U)| \geq 1$ and $x^*(\delta(U)) \geq 1$, we have $|S \cap \delta(U)| = 1$ and $x^*(\delta(U)) < 2$. Therefore, $\sum_{v \in U} |S \cap \delta(v)| = 2|S[U]| + |S \cap \delta(U)|$ is odd, which implies that $|\text{odd}(S) \cap U|$ is odd. Moreover, $\delta(U)$ is an s-t-cut because $x^*(\delta(U)) < 2$. Hence, $|T_S \cap U| = |(\text{odd}(S) \triangle \{s, t\}) \cap U|$ is even. □

Figure 6.3 shows a family of instances that prove that the approximation ratio of Christofides' algorithm is no better than $\frac{5}{3}$. In particular, Christofides' algorithm has a worse approximation ratio for the s-t-path TSP than for its special case TSP.

Until 2010 Christofides' algorithm remained to be the best known approximation algorithm for TSP, graph TSP, s-t-path TSP, and s-t-path graph TSP. While for TSP we still don't know how to achieve any better approximation ratio, a lot of progress has been made for s-t-path TSP, graph TSP, and s-t-path graph TSP. See Figure 6.4 and the following sections.

6.4 Approximation algorithms for s-t-path TSP

The first improvement over Christofides' algorithm for the s-t-path TSP was the Best-of-many Christofides algorithm by An, Kleinberg, and Shmoys [AKS15]. For $s \neq t$, they start by computing an optimum solution x^* to the linear program (6.2). Then they write x^* as a convex combination of incidence vectors of spanning trees $(V, S_1), \ldots, (V, S_k)$, i.e. $x^* = \sum_{i=1}^{k} p_i \cdot \chi^{S_i}$ for some coefficients $p_1, \ldots, p_k > 0$ with $\sum_{i=1}^{k} p_i = 1$. This is possible because x^* is contained in the spanning tree polytope (6.3) as shown above.

Now they apply parity correction as in Christofides' algorithm to each of the spanning trees in the convex combination, i.e. for each $i \in \{1, \ldots, k\}$ they compute a cheapest T_{S_i}-join J_i. Finally, they simply return the cheapest of the resulting s-t-tours $S_1 \mathbin{\dot\cup} J_1, \ldots, S_k \mathbin{\dot\cup} J_k$. An, Kleinberg, and Shmoys proved that this algorithm is a $\frac{1+\sqrt{5}}{2}$-approximation algorithm [AKS15]. Sebő [Seb13] gave an improved analysis and showed that the Best-of-many Christofides algorithm is a 1.6-approximation algorithm for the s-t-path TSP.

Vygen [Vyg16] introduced the idea to work not with an arbitrary decomposition of the LP solution x^* into incidence vectors of spanning trees, but one with particular properties. He gave a 1.599-approximation algorithm. Then Gottschalk and Vygen [GV18] showed how to find a decomposition of x^* into spanning trees that provides a lot more structure (see Theorem 7.1). They showed that proceeding as in the Best-of-many Christofides algorithm, but starting from their structured decomposition of x^* into spanning trees instead of an arbitrary one, yields a 1.566-approximation algorithm.

Sebő and van Zuylen [SvZ19] proposed the Best-of-many Christofides algorithm with lonely edge deletion. They start with the same decomposition of x^* into incidence vectors of spanning trees as Gottschalk and Vygen [GV18], but then they delete some edges of these trees before doing parity correction on the resulting forest. In the end two copies of some of the edges are added to ensure connectivity. The main idea is that parity correction will often help connectivity; so one saves more by deleting edges than one pays for reconnection at the end. Sebő and van Zuylen [SvZ19] proved that their algorithm yields a $(\frac{3}{2} + \frac{1}{34})$-approximation. We will discuss this algorithm in more detail in Chapter 7 and give an improved analysis.

All approximation algorithms for s-t-path TSP that we mentioned so far have been analyzed with respect to the optimum value of (TSPP LP). Thus their analysis also yields an upper bound on the integrality ratio of (TSPP LP). In fact the best known upper bound of 1.5284 on the integrality ratio results from the improved analysis of the algorithm by Sebő and van Zuylen [SvZ19] that we will present in Chapter 7.

The approximation ratio however has been improved further. Traub and Vygen [TV19a] gave a $(\frac{3}{2} + \varepsilon)$-approximation algorithm for every fixed $\varepsilon > 0$. They used a different approach than the previous algorithms, based on dynamic programming. Zenklusen [Zen19] simplified their algorithm and improved the approximation ratio to $\frac{3}{2}$, matching the best known approximation ratio for the special case TSP.

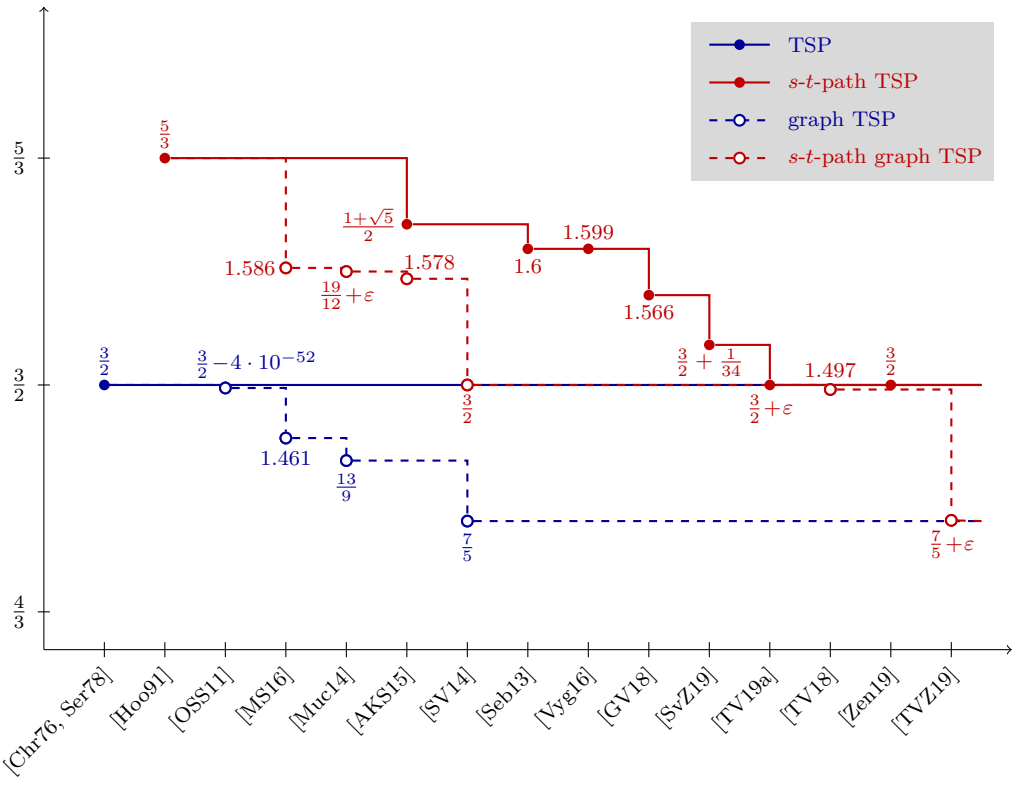

Figure 6.4: Development of the approximation ratios obtained for TSP, graph TSP and their path versions.
The circles show improved approximation ratios proved in the publications indicated at the bottom (from left to right in the order of discovery). Blue points refer to (graph) TSP and red points to the more general s-t-path (graph) TSP. Filled circles show results that apply to the weighted case, whereas circles with white interior indicate results that apply only to the graph case (with unit weights). The solid lines indicate the development of the approximation ratios for TSP (blue) and s-t-path TSP (red); the dashed lines refer to the graph versions.
Until 2011 Christofides' algorithm remained the best known approximation algorithm for all four of these variants of TSP. By now, there is a $\frac{3}{2}$-approximation algorithm for the s-t-path TSP, matching the approximation ratio of $\frac{3}{2}$ that Christofides' algorithm yields for TSP. Moreover, we have a $\frac{7}{5}$-approximation algorithm for graph TSP and a $(\frac{7}{5}+\varepsilon)$-approximation algorithms for its path version (for any fixed $\varepsilon > 0$). Only for TSP the approximation ratio achieved by Christofides' algorithm has not been improved.

Therefore, currently the best known approximation ratios for TSP and s-t-path TSP are the same, but the $\frac{3}{2}$-approximation algorithm for the s-t-path TSP by Zenklusen was found only very recently, more than 40 years after Christofides' $\frac{3}{2}$-approximation algorithm for TSP. In Chapter 9 we show that the s-t-path TSP is not much harder to approximate than TSP: we show that the existence of an α-approximation algorithm for TSP implies that there is an $(\alpha+\varepsilon)$-approximation algorithm for the s-t-path TSP (for any fixed $\varepsilon > 0$). This result avoids future differences between the approximation ratios for TSP and s-t-path TSP up to an arbitrarily small $\varepsilon > 0$.

6.5 Approximation algorithms for graph TSP and s-t-path graph TSP

The graph TSP and the s-t-path graph TSP are the unit-weight special cases of TSP and s-t-path TSP, respectively where we have $c(e) = 1$ for all $e \in E$, i.e. the objective is to minimize the number of edges of our tour. One reason for studying these natural special cases is that the well-known examples that yield the best known lower bounds for the integrality ratio are graph instances (see Figure 6.1 and Figure 6.2). Another motivation is that some techniques developed for the graph case led to progress for general weights later on. For example the algorithm by Gottschalk and Vygen [GV18] used ideas from Gao's [Gao13] $\frac{3}{2}$-approximation algorithm for the s-t-path graph TSP.

The standard LP relaxations for graph TSP and its path version are

$$\begin{aligned} \min\ & x(E) \\ \text{s.t.}\quad & x(\delta(U)) \geq 2 \quad \text{for } \emptyset \neq U \subsetneq V \\ & x_e \geq 0 \quad \text{for } e \in E \end{aligned} \tag{Graph TSP LP}$$

for the graph TSP, and

$$\begin{aligned} \min\ & x(E) \\ \text{s.t.}\quad & x(\delta(U)) \geq 2 \quad \text{for } \emptyset \neq U \subseteq V \setminus \{s,t\} \\ & x(\delta(U)) \geq 1 \quad \text{for } \{s\} \subseteq U \subseteq V \setminus \{t\} \\ & x_e \geq 0 \quad \text{for } e \in E \end{aligned} \tag{Graph TSPP LP}$$

for the s-t-path graph TSP.

In contrast to the more general TSP, for graph TSP we know how to achieve a better approximation ratio than $\frac{3}{2}$. The first improvement over Christofides' algorithm for graph TSP was achieved by Oveis Gharan, Saberi, and Singh [OSS11]. They gave a $(\frac{3}{2}-\varepsilon)$-approximation for some very small $\varepsilon > 0$.

Next, Mömke and Svensson [MS16] improved the approximation ratio for graph TSP to 1.461. They also gave a 1.586-approximation for s-t-path graph TSP, improving on the approximation ratio of $\frac{5}{3}$ achieved by Christofides' algorithm [Hoo91]. They introduced the so-called *removable-pairings technique*. We will see an application of this technique in Section 8.6. Mucha [Muc14] gave an improved analysis of the algorithm by Mömke and Svensson and improved the approximation ratios to $\frac{13}{9}$ for graph TSP and $\frac{19}{12}+\varepsilon$ for the path version.

The next improvement was by An, Kleinberg, and Shmoys [AKS15] who gave a 1.578-approximation algorithm for s-t-path graph TSP. Then Sebő and Vygen [SV14] improved the approximation ratios to $\frac{7}{5}$ for graph TSP and to $\frac{3}{2}$ for the s-t-path graph TSP. Their algorithm uses ear-decompositions that have particular properties and are optimized using matroid intersection. We will describe their approach in more detail in Section 8.1.4.

Because the algorithms we mentioned so far were all analyzed with respect to the values of (Graph TSP LP) and (Graph TSPP LP), the results also imply upper bounds on the integrality ratio of these LP relaxations. In particular, the analysis of the $\frac{3}{2}$-approximation algorithm for s-t-path graph TSP by Sebő and Vygen [SV14] implies that the integrality ratio of (Graph TSPP LP) is equal to $\frac{3}{2}$. Later Gao [Gao13] gave a simpler proof of this result. Among all the variants of TSP and its path version that we discuss here, the s-t-path graph TSP is the only variant for which we know the integrality ratio of the standard LP relaxation exactly.

If one analyzes an algorithm solely with respect to the value of (Graph TSPP LP), one cannot achieve an approximation ratio better than $\frac{3}{2}$ for the s-t-path graph TSP. However, in Chapter 8 we present an algorithm that gets below this threshold and achieves an approximation ratio of 1.497. We also show that the instances from Figure 6.2 are essentially "the only examples" (up to small local differences) where the integrality ratio converges to $\frac{3}{2}$.

The most recent improvement for the s-t-path graph TSP is a black-box reduction to graph TSP that we will present in Chapter 9: if there exists an α-approximation algorithm for graph TSP for some $\alpha > 1$, then there is an $(\alpha + \varepsilon)$-approximation algorithm for the s-t-path graph TSP for any fixed $\varepsilon > 0$. Applying this result to the $\frac{7}{5}$-approximation algorithm by Sebő and Vygen [SV14], yields a $(\frac{7}{5} + \varepsilon)$-approximation algorithm for the s-t-path graph TSP for any fixed $\varepsilon > 0$.

Chapter 7

An improved upper bound on the integrality ratio for *s*-*t*-path TSP

In this chapter we give an improved analysis of the Best-of-many Christofides algorithm with lonely edge deletion that was proposed by Sebő and van Zuylen [SvZ19]. Our analysis implies a better upper bound on the integrality ratio of the standard LP relaxation (TSPP LP) for the s-t-path TSP. This chapter is based on [TV19b], which is joint work with Jens Vygen.

7.1 Best-of-many Christofides with lonely edge deletion

In this section we explain the algorithm by Sebő and van Zuylen [SvZ19] that we will analyze. To simplify notation, we will only consider instances of the s-t-path TSP with $s \neq t$. If $s = t$, Christofides' algorithm already yields a $\frac{3}{2}$-approximation with respect to the value of (TSP LP), which is equivalent to (TSPP LP) for $s = t$. We use the formulation of the s-t-path TSP in which $G = (V, E)$ is a complete graph, the edge costs c fulfill the triangle inequality, and we are required to compute an s-t-tour that visits every vertex exactly once.

First, recall the Best-of-many Christofides algorithm that was proposed and analyzed by An, Kleinberg and Shmoys [AKS15]: it starts by computing an optimum solution x^* to the LP

$$\begin{aligned} &\min c(x) \\ &\text{s.t.} \quad x(\delta(U)) \geq 2 \quad \text{for } \emptyset \subsetneq U \subseteq V \setminus \{s,t\}, \\ &\qquad x(\delta(U)) \geq 1 \quad \text{for } \{s\} \subseteq U \subseteq V \setminus \{t\}, \\ &\qquad x(\delta(v)) = 2 \quad \text{for } v \in V \setminus \{s,t\}, \\ &\qquad x(\delta(v)) = 1 \quad \text{for } v \in \{s,t\}, \\ &\qquad x(e) \geq 0 \quad \text{for } e \in E. \end{aligned} \tag{7.1}$$

We write x^* as a convex combination of incidence vectors of spanning trees, i.e. $x^* = \sum_{j=1}^{k} p_j \chi^{S_j}$ for spanning trees $(V, S_1), \ldots, (V, S_k)$ and nonnegative coefficients $p_1, \ldots, p_k$ with $\sum_{j=1}^{k} p_j = 1$. Then parity correction as in Christofides' algorithm is applied to each of the k spanning trees; finally the best of the resulting s-t-tours is selected.

A key observation of An, Kleinberg and Shmoys [AKS15], used in all subsequent works, was that the set $\mathcal{N} \coloneqq \{\delta(U) : \{s\} \subseteq U \subseteq V \setminus \{t\},\ x^*(\delta(U)) < 2\}$ of *narrow* cuts is induced by a chain.

The algorithm can be further improved by using a convex combination with certain properties [Vyg16, GV18]. In particular, Gottschalk and Vygen [GV18] showed:

Theorem 7.1. *Let x^* be an optimum solution to the LP (7.1) and $\mathcal{N}$ the set of narrow cuts. Then there exist spanning trees $(V, S_1), \dots, (V, S_k)$ and nonnegative coefficients $p_1, \dots, p_k$ with $\sum_{j=1}^k p_j = 1$ such that*

- *$x^* = \sum_{j=1}^k p_j \chi^{S_j}$ and*
- *for every $C \in \mathcal{N}$ there exists an $r \in \{1, \dots, k\}$ with $\sum_{j=1}^r p_j = 2 - x^*(C)$ and $|C \cap S_j| = 1$ for all $j = 1, \dots, r$.*

Schalekamp et al. [SSTvZ18] found a simpler proof of this theorem. We will work with such a convex combination henceforth.

Sebő and van Zuylen [SvZ19] had the brilliant idea to delete some of the edges in each spanning tree and do parity correction on the resulting forest. This can save cost because parity correction will often reconnect the connected components of the forest anyway.

Call an edge e and a cut $C \in \mathcal{N}$ *lonely* in tree S_j if $\{e\} = C \cap S_j$ and $\sum_{i=1}^j p_i \leq 2 - x^*(C)$. Then we also say that e is lonely at C. We denote the lonely cuts in S_j by $\mathcal{L}(S_j)$. Let F_j be the edge set of the forest that results from S_j by deleting its lonely edges. The algorithm by Sebő and van Zuylen [SvZ19] does parity correction on each forest (V, F_j). Let $T_j \coloneqq \mathrm{odd}(F_j) \mathbin{\triangle} \{s, t\}$ denote the set of vertices whose degree in F_j has the wrong parity.

Every T_j-join J must contain an edge (in fact, an odd number of edges) in every lonely cut of S_j (because they are all T_j-cuts, i.e. cuts $\delta(U)$ for a vertex set U with $|U \cap T_j|$ odd). However, this does not imply that $F_j \mathbin{\dot\cup} J$ is connected, because an edge of J can belong to several lonely cuts of S_j. In this case we can, for all but one of these cuts, add two copies of the lonely edge of S_j in this cut (to ensure connectivity without changing parities).

If we choose a T_j-join J for parity correction, we will pay a total of at most $\sum_{e \in J} c^j(e)$, where

$$c^j(e) \ \coloneqq\ c(e) \ + \sum_{C \in \mathcal{L}(S_j),\, e \in C} 2c(C \cap S_j) \ - \ \max\left\{0, \max_{C \in \mathcal{L}(S_j),\, e \in C} 2c(C \cap S_j)\right\};$$

here the second and third terms account for the reconnection cost.

Algorithm 3 formally describes the Best-of-many Christofides algorithm with lonely edge deletion due to Sebő and van Zuylen [SvZ19]. This is the algorithm that we will analyze.

Algorithm 3: Best-of-many Christofides algorithm with lonely edge deletion

1. Compute an optimum solution x^* to the LP (7.1).
2. Compute $x^* = \sum_{j=1}^{k} p_j \chi^{S_j}$ as in Theorem 7.1
3. Do the following for each $j = 1, \dots, k$:
   (a) Compute a T_j-join J_j with minimum c^j-cost.
   (b) Compute a minimum c-cost subset $R_j \subseteq S_j \setminus F_j$ of the lonely edges such that $F_j \dot\cup J_j \dot\cup R_j$ is connected.
   (c) Find an Eulerian s-t-walk in $H_j \coloneqq F_j \dot\cup J_j \dot\cup R_j \dot\cup R_j$ and shortcut whenever a vertex is visited more than once.
4. Return the cheapest of these k s-t-tours.

7.2 Outline of the new analysis

Let us first outline the main ideas of our new analysis. In Section 7.3 and Section 7.4 we will then give a formal analysis of Algorithm 3 and prove our new upper bound on the integrality ratio of (TSPP LP).

By definition of c^j, the cost of the tour H_j is at most $c(F_j) + c^j(J_j)$. The cost of the T_j-join J_j is the minimum cost of a vector y in the T_j-join polyhedron

$$\Big\{ y \in \mathbb{R}^E_{\geq 0} : y(\delta(U)) \geq 1 \text{ for } U \subseteq V \text{ with } |U \cap T_j| \text{ odd} \Big\}; \tag{7.2}$$

see Theorem 2.16. We call a vector y in (7.2) a *parity correction vector*. Note that every parity correction vector yields an upper bound on the cost of J_j. A first attempt to design a parity correction vector could be the vector $\beta x^* + (1 - 2\beta)\chi^{S_j}$ for some $0 \leq \beta \leq \frac{1}{2}$. This vector has value at least one on all cuts except the narrow cuts. The narrow cuts can be repaired by adding fractions of incidence vectors of lonely edges (not necessarily from the same tree). We will pay all this and the reconnection cost by what we gain by deleting the lonely edges. Then our total cost is

$$\begin{aligned} \min_{j=1}^{k} c(H_j) &\leq \sum_{j=1}^{k} p_j \cdot c(H_j) \\ &\leq \sum_{j=1}^{k} p_j \cdot \big(c(S_j) + \beta c(x^*) + (1 - 2\beta) c(S_j)\big) \\ &= (2 - \beta) \cdot c(x^*). \end{aligned} \tag{7.3}$$

Hence we would like to choose β as large as possible. Unfortunately, for $\beta = \frac{1}{2}$ we need too much from the lonely edges. By reducing β, we can increase the value of $\beta x^* + (1 - 2\beta)\chi^{S_j}$ on the narrow cuts and thus decrease the required amount of lonely edges. Choosing $\beta = \frac{8}{17}$ is sufficient and this is essentially what Sebő and van Zuylen did.

In our parity correction vector for a forest F_j we will use lonely edges of S_j and of earlier trees (i.e. from trees S_i with $i \leq j$). If we increase β for the early trees and decrease β for the late trees, we need more from the lonely edges in the early trees, but less in the late trees. This will improve our bound if the late trees are cheaper (and this is indeed true in the worst case).

The algorithm computes k tours $H_1, \ldots, H_k$. All previous analyses, like (7.3), computed an upper bound on $\sum_{j=1}^{k} p_j c(H_j)$. Instead, we will compute a weighted average with different weights q_j, giving a higher weight to tours resulting from early trees.

We choose values $\beta_j \in [0, \frac{1}{2}]$ and weights $q_j > 0$ with $\sum_j q_j = 1$ and such that $q_j \cdot (2 - 2\beta_j) = M \cdot p_j$ for some constant $M > 0$. Such a choice allows to bound the cost of our tour against the LP value $c(x^*)$:

$$\sum_{j=1}^{k} q_j \cdot c(H_j) \;\leq\; \sum_{j=1}^{k} q_j \cdot (\beta_j c(x^*) + (2 - 2\beta_j) c(S_j)) \;=\; \left(\sum_{j=1}^{k} q_j \beta_j + M \right) c(x^*).$$

Intuitively, choosing $\frac{q_j}{p_j}$ (and thus β_j) larger for the early trees is good, because for the early trees we delete more lonely edges (cf. Theorem 7.1). This allows us to choose the average value of β larger and thus improves our upper bound.

We first analyze the cost of a tour resulting from a single tree S_j. Later, we will take a weighted average.

7.3 Analyzing one tree

Let $j \in \{1, \ldots, k\}$. To bound the cost of parity correction of the forest F_j, we follow Wolsey's approach [Wol80] and use a vector in the T_j-join polyhedron (7.2).

Let $0 \leq \beta \leq \frac{1}{2}$ and $\alpha := 1 - 2\beta \geq 0$. Moreover, for $C \in \mathcal{N}$, let $v^C \in \mathbb{R}^E_{\geq 0}$ be a vector with $v^C(C) = 1$ and $v^C(e) = 0$ unless e is lonely at C in some tree (not necessarily in S_j). We will choose v^C later. We define

$$y^j_\beta := \beta x^* + \alpha \chi^{S_j} + \sum_{C \in \mathcal{L}(S_j)} \beta(2 - x^*(C)) \chi^{S_j \cap C} + \sum_{C \in \mathcal{N} \setminus \mathcal{L}(S_j)} \max\{0,\, \beta(2 - x^*(C)) - \alpha\}\, v^C.$$

The first sum is the contribution from lonely edges of the tree S_j itself, in order to repair the lonely cuts of S_j. The second sum is the contribution from lonely edges of earlier trees, in order to repair the other narrow cuts.

Obviously, y^j_β is a nonnegative vector. We show that y^j_β is a parity correction vector, i.e. a vector in (7.2).

Lemma 7.2. *For every T_j-cut C we have $y^j_\beta(C) \geq 1$.*

Proof. Let $C = \delta(U)$ be a T_j-cut. Since $|U \cap \mathrm{odd}(F_j)|$ is odd if and only if $|F_j \cap \delta(U)|$ is odd, we conclude that $|U \cap \{s, t\}| + |F_j \cap C|$ is odd. We now distinguish several cases.

Case 1: $|U \cap \{s, t\}|$ is odd (i.e., C is an s-t-cut).

Then $|F_j \cap C|$ is even. We now consider two subcases.

Case 1a: $C \in \mathcal{L}(S_j)$.

Then $y^j_\beta(C) \geq \beta x^*(C) + \alpha + \beta(2 - x^*(C)) = \alpha + 2\beta = 1$.

Case 1b: $C \notin \mathcal{L}(S_j)$.
Since $|F_j \cap C|$ is even, we have $|S_j \cap C| \geq |F_j \cap C| \geq 2$ or $|F_j \cap C| = 0$. Since $C \notin \mathcal{L}(S_j)$, if $F_j \cap C$ is empty, the cut C must contain at least two edges that are lonely in S_j. So we have also in this case $|S_j \cap C| \geq 2$. Thus $y^j_\beta(C) \geq \beta x^*(C) + 2\alpha + \max\{0, \beta(2 - x^*(C)) - \alpha\}$ (note that the last term is zero if $C \notin \mathcal{N}$). We conclude $y^j_\beta(C) \geq \beta x^*(C) + 2\alpha + \beta(2 - x^*(C)) - \alpha = \alpha + 2\beta = 1$.

Case 2: $|U \cap \{s, t\}|$ is even.
Then $x^*(C) \geq 2$. Hence, $y^j_\beta(C) \geq \beta x^*(C) + \alpha \geq 2\beta + \alpha = 1$. □

Moreover,we have $y^j_\beta \geq 0$. Thus y^j_β is contained in the T_j-join polyhedron (7.2), and so $\min\{c^j(J) : J \text{ a } T_j\text{-join}\} \leq c^j(y^j_\beta)$.

A key observation of Sebő and van Zuylen [SvZ19] was that the need for reconnection is unlikely. Only bad edges can result in reconnection, where an edge is called *bad* (for S_j) if it is contained in more than one lonely cut. The edges in S_j are never bad for S_j, nor are the lonely edges of trees that come earlier in the list $S_1, \ldots, S_r$. Therefore, an edge e with $v^C(e) > 0$ for some $C \in \mathcal{N} \setminus \mathcal{L}(S_j)$ is not bad for S_j. At this point we use the particular choice of the decomposition of x^* into incidence vectors of spanning trees. For every edge e that is not bad we have $c^j(e) = c(e)$. Hence

$$\begin{aligned} c^j(y^j_\beta) \;&=\; \beta\, c^j(x^*) \;+\; \alpha\, c(S_j) + \sum_{C \in \mathcal{L}(S_j)} \beta(2 - x^*(C))\, c(S_j \cap C) \\ &\quad + \sum_{C \in \mathcal{N} \setminus \mathcal{L}(S_j)} \max\Big\{0,\, \beta(2 - x^*(C)) - \alpha\Big\} c(v^C). \end{aligned}$$

Moreover, Sebő and van Zuylen [SvZ19] showed:

Lemma 7.3.

$$c^j(x^*) \;\leq\; c(x^*) + \sum_{C \in \mathcal{L}(S_j)} 2(x^*(C) - 1) c(S_j \cap C).$$

Proof. We consider a directed auxiliary graph with vertex set $E \,\dot\cup\, \mathcal{L}(S_j)$ and edge set $\{(e, C) : e \in C,\ C \in \mathcal{L}(S_j)\}$. We show that there exists a flow f in this auxiliary graph with $f(\delta^+(e)) \leq x^*_e$ for all $e \in E$ and $f(\delta^-(C)) \geq 1$ for all $C \in \mathcal{L}(S_j)$. Then

$$\begin{aligned} c^j(x^*) - c(x^*) \;&\leq\; \sum_{e \in E} \left(\sum_{C \in \mathcal{L}(S_j),\, e \in C} (x^*_e - f(e, C)) \cdot 2 \cdot c(C \cap S_j) \right) \\ &\leq \sum_{C \in \mathcal{L}(S_j)} 2 \cdot c(C \cap S_j) \sum_{e \in C} (x^*_e - f(e, C)) \\ &\leq \sum_{C \in \mathcal{L}(S_j)} 2 \cdot c(C \cap S_j) \cdot (x^*(C) - 1). \end{aligned}$$

To show that a flow f as above exists, we need to show that for every set $\mathcal{L}' \subseteq \mathcal{L}(S_j)$ we have $x^*\left(\cup_{C\in\mathcal{L}'} C\right) \geq |\mathcal{L}'|$. Let $\{s\} \subseteq L_1 \subseteq \cdots \subseteq L_k \subseteq V \setminus \{t\}$ such that $\mathcal{L}' = \{\delta(L_1), \ldots, \delta(L_k)\}$. Then

$$\begin{aligned} 2 \cdot x^*\left(\cup_{C\in\mathcal{L}'} C\right) &= x^*(\delta(L_1)) + x^*(\delta(L_k)) + \sum_{i=1}^{k-1} x^*(\delta(L_{i+1} \setminus L_i)) \\ &\geq 1 + 1 + 2 \cdot (k-1) \;=\; 2 \cdot |\mathcal{L}'|. \end{aligned}$$

□

Therefore, the cost of the tour that results from the tree S_j is at most

$$\begin{aligned} & c(F_j) + \min\{c^j(J) : J \text{ a } T_j\text{-join}\} \\ \leq\; & c(S_j) - \sum_{C\in\mathcal{L}(S_j)} c(S_j \cap C) + c^j(y^j_\beta) \\ \leq\; & (1+\alpha)c(S_j) + \beta c(x^*) + \sum_{C\in\mathcal{L}(S_j)} \left(2\beta(x^*(C)-1) - 1 + \beta(2 - x^*(C))\right) c(S_j \cap C) \\ & + \sum_{C\in\mathcal{N}\setminus\mathcal{L}(S_j)} \max\left\{0,\, \beta(2-x^*(C)) - \alpha\right\} c(v^C) \\ =\; & (1+\alpha)c(S_j) + \beta c(x^*) - \sum_{C\in\mathcal{L}(S_j)} \left(\alpha + \beta(2 - x^*(C))\right) c(S_j \cap C) \\ & + \sum_{C\in\mathcal{N}\setminus\mathcal{L}(S_j)} \max\left\{0,\, \beta(2-x^*(C)) - \alpha\right\} c(v^C), \end{aligned} \tag{7.4}$$

since $\alpha = 1 - 2\beta$.

7.4 Average cost

It will be useful to index the trees by a continuum and define S_σ for all $0 < \sigma \leq 1$, where $S_\sigma = S_j$ if $\sum_{i=1}^{j-1} p_i < \sigma \leq \sum_{i=1}^{j} p_i$.

Let $h\colon [0,1] \to [0,1]$ be an integrable function to be chosen later. The weight of the tour resulting from S_σ will be proportional to $1 + h(\sigma)$. Also α and β depend on σ, namely as follows:

$$\alpha_\sigma = \frac{1 - h(\sigma)}{1 + h(\sigma)} \qquad \text{and} \qquad \beta_\sigma = \frac{h(\sigma)}{1 + h(\sigma)}.$$

Note that indeed $0 \leq \beta_\sigma \leq \frac{1}{2}$ and $\alpha_\sigma + 2\beta_\sigma = 1$ for all σ.

Moreover, we set

$$v^C \;:=\; \frac{1}{\int_0^z (1 - h(\sigma) + z h(\sigma))\,\mathrm{d}\sigma} \int_0^z (1 - h(\sigma) + z h(\sigma)) \cdot \chi^{S_\sigma \cap C}\,\mathrm{d}\sigma,$$

where we abbreviated $z := 2 - x^*(C)$. Then indeed $v^C(C) = 1$ for all $C \in \mathcal{N}$, and $v^C(e) = 0$ unless e is lonely at C.

We will now show under which condition the last two terms in (7.4) vanish:

Lemma 7.4. *Suppose*

$$\int_z^1 \max\{0, h(\sigma) - 1 + zh(\sigma)\}\, d\sigma + \int_0^z \big(h(\sigma) - 1 - zh(\sigma)\big)\, d\sigma \;\leq\; 0 \tag{7.5}$$

for all $z \in [0,1]$. *Then*

$$\begin{aligned}\int_0^1 (1+h(\sigma))\Bigg(&- \sum_{C\in\mathcal{L}(S_\sigma)} \big(\alpha_\sigma + \beta_\sigma(2 - x^*(C))\big)\, c(S_\sigma \cap C) \\ &+ \sum_{C\in\mathcal{N}\setminus\mathcal{L}(S_j)} \max\left\{0,\, \beta_\sigma(2-x^*(C)) - \alpha_\sigma\right\} c(v^C)\Bigg)\, \mathrm{d}\sigma\end{aligned} \tag{7.6}$$

is nonpositive.

Proof. Again writing $z \coloneqq 2 - x^*(C)$, using

$$(1+h(\sigma))\big(\alpha_\sigma + \beta_\sigma(2-x^*(C))\big) \;=\; 1 - h(\sigma) + zh(\sigma)$$

and

$$(1+h(\sigma)) \max\left\{0,\, \beta_\sigma(2-x^*(C)) - \alpha_\sigma\right\} \;=\; \max\left\{0,\; h(\sigma) - 1 + zh(\sigma)\right\},$$

and changing the order of summation, we can rewrite (7.6) as

$$\begin{aligned}&- \sum_{C\in\mathcal{N}} \int_0^z \big(1 - h(\sigma) + zh(\sigma)\big)\, c(S_\sigma \cap C)\, \mathrm{d}\sigma \\ &+ \sum_{C\in\mathcal{N}} \int_z^1 \max\left\{0,\, h(\sigma) - 1 + zh(\sigma)\right\} c(v^C)\, \mathrm{d}\sigma.\end{aligned}$$

Hence (plugging in the definition of v^C) and using $1 - h(\sigma) + z \cdot h(\sigma) > 0$, it suffices to show that, for every $z \in (0,1]$,

$$-1 + \frac{1}{\int_0^z \big(1 - h(\sigma) + zh(\sigma)\big)\, \mathrm{d}\sigma} \int_z^1 \max\left\{0,\, h(\sigma) - 1 + zh(\sigma)\right\} d\sigma \;\leq\; 0.$$

which follows directly from (7.5). □

Lemma 7.5. *Let* $h\colon [0,1] \to [0,1]$ *be an integrable function with* (7.5) *for all* $z \in [0,1]$. *Then the Best-of-many Christofides algorithm with lonely edge deletion computes a solution of cost at most* $\rho^* \cdot c(x^*)$, *where*

$$\rho^* \;=\; 1 + \frac{1}{1 + \int_0^1 h(\sigma)\, \mathrm{d}\sigma}.$$

Proof. Combining (7.4) and Lemma 7.4, we get the following upper bound on the total cost of the Best-of-many Christofides algorithm with lonely edge deletion:

$$\begin{aligned}
&\tfrac{1}{\int_0^1 (1+h(\sigma))\,\mathrm{d}\sigma} \int_0^1 (1+h(\sigma)) \left(\beta_\sigma c(x^*) + (1+\alpha_\sigma) c(S_\sigma)\right) \mathrm{d}\sigma \\
&= \tfrac{1}{\int_0^1 (1+h(\sigma))\,\mathrm{d}\sigma} \int_0^1 \left(h(\sigma) c(x^*) + 2c(S_\sigma)\right) \mathrm{d}\sigma \\
&= \tfrac{1}{\int_0^1 (1+h(\sigma))\,\mathrm{d}\sigma} \left(\int_0^1 h(\sigma)\,\mathrm{d}\sigma + 2\right) c(x^*) \\
&= \tfrac{1}{\int_0^1 (1+h(\sigma))\,\mathrm{d}\sigma} \left(\int_0^1 (1+h(\sigma))\,\mathrm{d}\sigma + 1\right) c(x^*) \\
&= \left(1 + \tfrac{1}{1+\int_0^1 h(\sigma)\,\mathrm{d}\sigma}\right) c(x^*)
\end{aligned}$$

□

Now we can prove the main result:

Theorem 7.6. *Let*

$$\rho^* \;:=\; 1 + \frac{1}{1 + 4\ln(\frac{5}{4})}.$$

Then the Best-of-many Christofides algorithm with lonely edge deletion computes a solution of cost at most $\rho^* \cdot c(x^*)$.

Proof. We set $h(\sigma) = \frac{4}{4+\sigma}$ for $0 \le \sigma \le 1$. Then $\int_0^1 \frac{4}{4+\sigma}\,\mathrm{d}\sigma = 4\ln(\frac{5}{4})$. We need to check (7.5).

Note that $h(\sigma) - 1 + z h(\sigma) > 0$ if and only if $\frac{4}{4+\sigma} = h(\sigma) > \frac{1}{1+z}$, i.e., $\sigma < 4z$. Hence to prove (7.5) it suffices to show

$$\int_z^{4z} \big(h(\sigma) - 1 + z h(\sigma)\big)\,\mathrm{d}\sigma + \int_0^z \big(h(\sigma) - 1 - z h(\sigma)\big)\,\mathrm{d}\sigma \;\le\; 0$$

The left-hand side is

$$4(1+z)(\ln(4z+4) - \ln(z+4)) + 4(1-z)(\ln(z+4) - \ln(4)) - 4z,$$

so (dividing by 4) we need to check

$$(1+z)\ln\tfrac{4z+4}{z+4} + (1-z)\ln\tfrac{z+4}{4} - z \;\le\; 0.$$

This is true for $z = 0$, moreover the derivative of the left-hand side is

$$\ln\tfrac{16(z+1)}{(z+4)^2} - \tfrac{2z}{z+4}.$$

Using $\ln x \le x - 1$ for all $x > 0$ this is at most

$$\tfrac{16(z+1)}{(z+4)^2} - 1 - \tfrac{2z}{z+4} \;=\; \tfrac{16(z+1) - (z+4)^2 - 2z(z+4)}{(z+4)^2} \;=\; \tfrac{-3z^2}{(z+4)^2} \;\le\; 0.$$

□

Theorem 7.6 immediately implies that the integrality ratio of (7.1) is at most ρ^*. Because the LP relaxations with and without degree constraints are equivalent, also the integrality ratio of (TSPP LP) is at most ρ^*. Note that $\rho^* < 1.5284$, which is strictly less than the bound $\frac{3}{2} + \frac{1}{34}$ of Sebő and van Zuylen [SvZ19]. We see that (7.5) is tight only for $z = 0$ with our choice of h. A better choice would lead to a better upper bound on the integrality ratio. However, we do not know how to find the best h. Numerical computations indicate that the best value that can be obtained in this way is approximately 1.5273.

Chapter 8

Beating the integrality ratio for s-t-tours in graphs

In this chapter, we characterize instances of the s-t-path TSP (with many vertices) where the ratio $\frac{\mathrm{OPT}}{\mathrm{LP}}$ of the length OPT of an optimum s-t-tour and the optimum value LP of (Graph TSPP LP) is close to the integrality ratio $\frac{3}{2}$. We show that the classical example of such a worst case instance is "essentially the only such example".

Moreover, we prove that the s-t-path graph TSP has a 1.497-approximation algorithm, achieving for the first time an approximation ratio below the integrality ratio of (Graph TSPP LP).

Most parts of this chapter have been previously published in [TV18], which is joint work with Jens Vygen.

8.1 Introduction and preliminaries

Almost all the algorithms for the graph TSP, the ATSP and their path versions work with the classical linear programming relaxations, with an LP solution as starting point of the algorithm or at least for the analysis. Although these LPs have been studied intensively for decades, both for the symmetric and the asymmetric TSP, we still do not know their integrality ratios. For the (symmetric) s-t-path TSP we are quite close: the integrality ratio is between 1.5 and 1.5284 (cf. Chapter 7), and in the s-t-path graph TSP, it is indeed $\frac{3}{2}$, and an approximation guarantee matching the integrality ratio is known [SV14].

In this chapter we go below this threshold: we show that there is a 1.497-approximation algorithm for the s-t-path TSP. Moreover, we show that all instances with $\frac{\mathrm{OPT}}{\mathrm{LP}}$ close to $\frac{3}{2}$ (and many vertices) are very similar to the classical examples (Figure 6.2). Our proof is based on the $\frac{3}{2}$-approximation algorithm by Sebő and Vygen [SV14], but we introduce several new techniques. These include a new type of ear-decomposition, an enhanced ear induction that reveals a novel connection to matroid union, and a stronger lower bound.

We give an upper bound on the integrality ratio that depends on the distance of the vertices s and t. This upper bound is equal to $\frac{3}{2}$ only if the distance of s and t equals half the length of an optimum s-t-tour and decreases for smaller s-t-distance. Also if the distance of s and t is larger half the length of an optimum s-t-tour, the

integrality ratio is less than $\frac{3}{2}$. This is much easier to prove and can show by a well-known argument; see Section 8.7. Our proof of the upper bound on the integrality ratio also yields a polynomial-time algorithm that computes an s-t-tour with significantly less than $\frac{3}{2}\mathrm{OPT}$ edges if the distance of s and t is relatively small.

In order to handle instances with large distance of s and t, we use a result from [Tra17] that reduces the s-t-path graph TSP to the special case where s and t have distance at most $\left(\frac{1}{3}+\varepsilon\right)\cdot\mathrm{OPT}$. We prove this reduction via a dynamic programming technique that will be further developed in Chapter 9, where we will prove an even better approximation ratio for the s-t-path graph TSP.

The rest of this section is organized as follows. We start with some short preliminaries on T-tours (Section 8.1.1) and ear-decompositions (Section 8.1.2). Then we explain a simple way of constructing s-t-tours from ear-decompositions using a technique called ear-induction (Section 8.1.3) and give an overview of the approach by Sebő and Vygen [SV14] (Section 8.1.4). In Section 8.1.5, we introduce the new notion of well-oriented ear-decompositions. Finally, in Section 8.1.6 we provide an overview of our new algorithm and outline the remaining part of this chapter.

8.1.1 Preliminaries on T-tours

A *T-tour* in an undirected graph G is a T-join $J \subseteq 2E(G)$ such that $(V(G), J)$ is connected; here $2E(G)$ denotes the multi-set of edges containing two copies of every edge of G. We allow taking two copies of an edge, but more than two are never useful. Henceforth we speak of edge sets and graphs even if they contain parallel edges. Moreover, in this chapter we will only consider undirected graphs and do not state this explicitly.

The notion of T-tours generalizes s-t-tours: an s-t-tour is a T-tour for $T = \{s\} \triangle \{t\}$. It is easy to see that a T-tour exists if and only if G is connected and $|T|$ is even. As an obvious lower bound, note that every T-tour has at least $n-1$ edges (for any T); here and throughout this chapter we denote $n := |V(G)|$.

The T-tour problem (in graphs or in general) has been introduced by Sebő and Vygen [SV14]; they gave a $\frac{3}{2}$-approximation algorithm for finding a smallest T-tour in a graph, and a $\frac{7}{5}$-approximation algorithm if $T = \emptyset$. Sebő [Seb13] showed an approximation ratio of $\frac{8}{5}$ for finding a minimum-weight T-tour in a weighted graph. These approximation ratios have not been improved since then. Section 8.2 of this chapter works for general T-tours, but later parts do not seem to extend beyond constant $|T|$. In particular, the reduction technique in Section 8.8 fails to generalize.

For a graph G and a set $T \subseteq V(G)$ with $|T|$ even and a set $W \subseteq V(G)$, let $(G,T)/W$ be the instance of the T-tour problem arising by *contraction* of W. More precisely, we define $(G,T)/W$ to be the instance of the T-tour problem where we are looking for a T'-tour in the graph G/W and T' contains all elements of $T \setminus W$ and contains in addition the vertex arising from the contraction if $|T \cap W|$ is odd.

Without loss of generality we may assume that the input graph is 2-vertex-connected because if $G = G[W_1] \cup G[W_2]$, where W_1 and W_2 share a single vertex, then it suffices to solve the instances $(G,T)/W_1$ and $(G,T)/W_2$ (cf. [SV14]).

8.1.2 Preliminaries on ear-decompositions

For a finite sequence $P_0, P_1, \dots, P_l$ of graphs, let $V_i = V(P_0) \cup V(P_1) \cup \dots \cup V(P_i)$ and $G_i = (V_i, E(P_1) \cup \dots \cup E(P_i))$. If P_0 has a single vertex, and each P_i is either a circuit with $|V(P_i) \cap V_{i-1}| = 1$ or a path such that exactly its endpoints belong to V_{i-1} $(i = 1, \dots, l)$, then $P_0, P_1, \dots, P_l$ is an *ear-decomposition* of G_l.

The graphs $P_1, \dots, P_l$ are called *ears*. The vertices of $V(P_i) \cap V_{i-1}$ are called *endpoints* of P_i, the other vertices of P_i are its *internal vertices*. We denote the set of internal vertices of P_i by $\mathrm{in}(P_i)$. We always have $|\mathrm{in}(P_i)| = |E(P_i)| - 1$. An ear is *open* if it is P_1 or it is a path. Other ears are *closed*. If all ears are open, the ear-decomposition is *open*. An ear is called an *r-ear* if it has exactly r edges. We denote the number of r-ears of a fixed ear-decomposition by k_r. The 1-ears are also called *trivial* ears; they have no internal vertices. A *short* ear is a 2-ear or 3-ear. Ears with more than three edges are called *long*. An ear is *odd* if the number of its edges is odd, otherwise *even*.

Figure 8.1(a) shows an ear-decomposition with P_0 colored black, a 6-ear P_1 colored brown, a closed 6-ear P_2 colored blue, an open 5-ear P_3 colored green, an open 6-ear colored cyan, and four 2-ears (gray, dotted). Every vertex is an internal vertex of exactly one ear (except for the vertex of P_0) and is colored accordingly in this figure.

We say that an ear P is *attached to* an ear Q (at v) if v is an internal vertex of Q and an endpoint of P. A vertex is *pendant* if it is not an endpoint of any nontrivial ear, and an ear is *pendant* if it is nontrivial and all its internal vertices are pendant. Having a fixed vertex set T, we call an ear P *clean* if it is short and $|T \cap \mathrm{in}(P)| = \emptyset$, i.e. none of its internal vertices is contained in T.

8.1.3 Simple ear induction

The following lemma tells how to construct a T-tour by considering the nontrivial ears in reverse order. For any nontrivial ear P_i, note that G_i/V_{i-1} is a circuit with $|E(G_i/V_{i-1})| = |E(P_i)|$.

Lemma 8.1 (Sebő and Vygen [SV14])**.** *Let P be a circuit and $T_P \subseteq V(P)$ with $|T_P|$ even. Then there exists a T_P-join $F \subseteq 2E(P)$ such that the graph $(V(P), F)$ is connected and*

$$|F| \;\leq\; \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2} + \gamma, \tag{8.1}$$

where $\gamma = 1$ if $|E(P)| \leq 3$ and $T_P = \emptyset$, and $\gamma = 0$ otherwise.

Proof. If $T_P = \emptyset$, then $F = E(P)$ does the job because $|E(P)| - 1 \geq 3$ or $\gamma = 1$.

Let now $T_P \neq \emptyset$. The vertices of T_P subdivide P into subpaths. Color these paths alternatingly red and blue. Let E_R and E_B denote the set of edges of red and blue subpaths, respectively. Without loss of generality $|E_R| \leq |E_B|$. Then we take two copies of each edge in E_R and one copy of each edge in E_B. Note that $E_R \neq \emptyset$, and remove one pair of parallel edges. This yields $F \subseteq 2E(P)$ with

$$|F| \;=\; |E_B| + 2|E_R| - 2 \;\leq\; \tfrac{3}{2}|E(P)| - 2 \;=\; \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}. \qquad \square$$

This can be used to construct a T-tour as follows (see Figure 8.1 (b)). Let $P_1, \dots, P_l$ be the nontrivial ears of an ear-decomposition of G (trivial ears can be deleted beforehand). Starting with $T_l := T$ and $F := \emptyset$, we do the following for $i = l, \dots, 1$. Apply

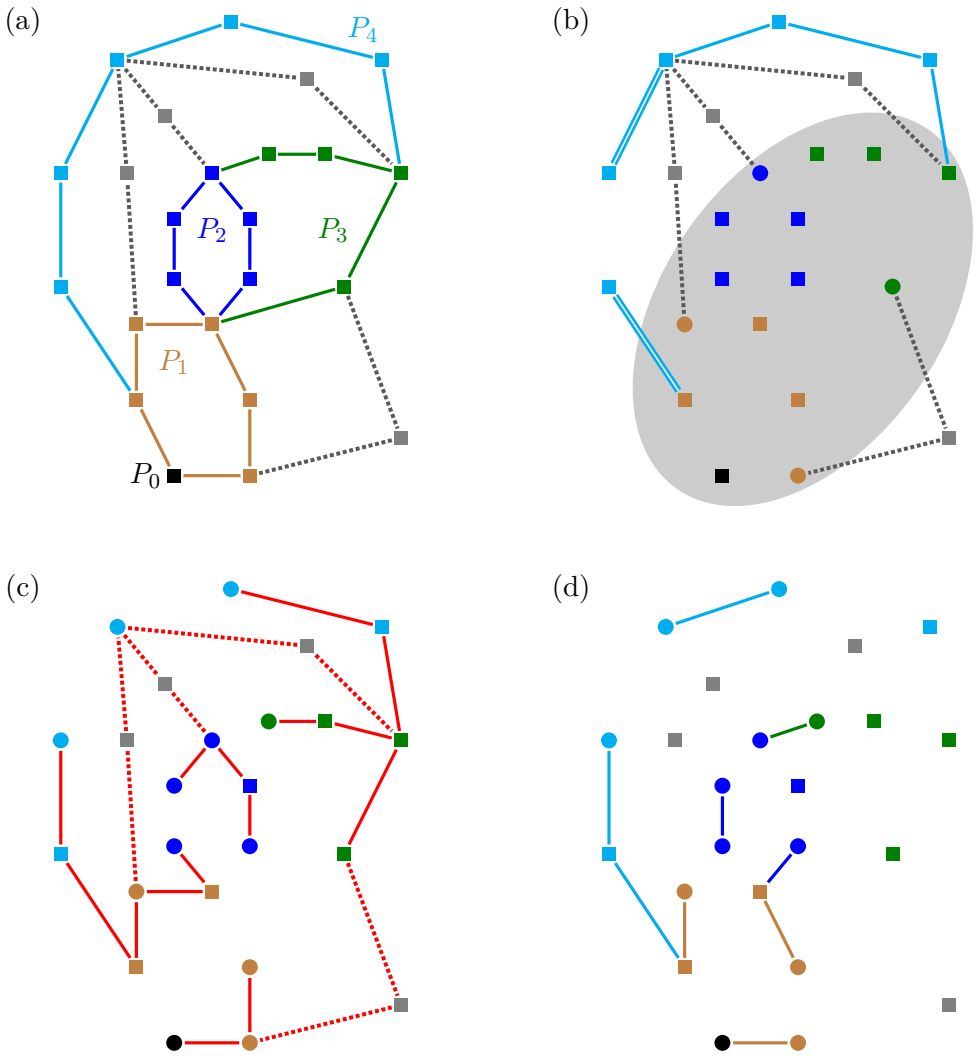

Figure 8.1: (a) An ear-decomposition with four non-pendant long ears $P_1, \ldots, P_4$ and four pendant 2-ears (gray, dotted). No trivial ears are shown in this picture. (b) Simple ear induction for $T = \emptyset$: for the four pendant ears we can just take all their edges. Then we deal with P_4, for this we contract the gray set and require odd parity at the middle vertex of P_4. One possible solution is shown in cyan. The set T_3 of vertices of vertices whose degree has the wrong parity after this step is indicated by circles; they are all contained in the gray set. (c) A spanning tree S containing the forest of short ears. Circles denote the set T_S of vertices whose degree has the wrong parity in S. (d) Parity correction by ear induction in the long ears constructs a T_S-join. The disjoint union of the edge sets in (c) and (d) is a tour.

Lemma 8.1 to $(G_i, T_i)/V_{i-1}$ and obtain a set $F_i \subseteq 2E(P_i)$. Set $F := F \cup F_i$ and $T_{i-1} := T_i \triangle \text{odd}(F_i)$. Then the union of F_i and any T_{i-1}-tour in G_{i-1} is a T_i-tour in G_i. By induction, $F_1 \dot\cup \ldots \dot\cup F_l$ is a T-tour in G. Since

$$|F_i| \;\leq\; \tfrac{3}{2}(|E(P_i)| - 1) - \tfrac{1}{2} + \gamma_i \;=\; \tfrac{3}{2}|\text{in}(P_i)| - \tfrac{1}{2} + \gamma_i,$$

this T-tour has at most $\frac{3}{2}(n-1)$ edges if $\gamma_i = 0$ for at least half of the nontrivial ears, in particular if most ears are long.

8.1.4 Outline of the Sebő–Vygen algorithm

The previously best approximation algorithm for the s-t-path graph TSP, due to Sebő and Vygen [SV14], is the basis of our work. Let us briefly review this algorithm before we explain how to improve on it. The previous section shows already why short ears (of length 2 and 3) need special attention.

The first step is to compute a *nice* ear-decomposition: one with minimum number of even ears, in which all short ears are pendant, and internal vertices of distinct short ears are non-adjacent. We will present a strengthening of this in Section 8.3.

The second step is to re-design the short ears so that as many of them as possible are part of a forest (i.e., help connecting vertices that are not internal vertices of short ears). Re-designing a short ear means changing its endpoints by replacing its first and/or last edge by an edge of a trivial ear. This can be reduced to a matroid intersection problem (with a graphic matroid and a partition matroid). For every short ear that is not part of this forest, we can raise the lower bound. We will present a refinement of this step in Section 8.4.

Finally, two simple algorithms are applied to the resulting ear-decomposition. If at least half of the nontrivial ears are long, simple ear induction (Lemma 8.1) yields a short tour. Otherwise one can obtain a cheap tour by a similar approach as Christofides' algorithm, computing a spanning tree and doing parity correction: first the forest of short ears is completed to a spanning tree without using trivial ears; see Figure 8.1 (c). Then the internal vertices of short ears in the forest (which are pendant) have still degree two. Therefore it is sufficient to do parity correction in the subgraph consisting of the other non-trivial ears. If the forest contains many short ears, this subgraph is relatively small. One can show that then parity correction is cheap, using a similar ear induction as in Section 8.1.3 (but without the connectivity requirement); see Figure 8.1 (d).

In this chapter we will combine the two algorithms in the last step to a single new step, which will be described in Section 8.2.

The critical case, when the Sebő–Vygen algorithm has no better approximation ratio than $\frac{3}{2}$, is when (essentially) all ears are even (note that we save $\frac{1}{2}$ more for odd ears in (8.1) by rounding down the right-hand side), half of the ears are 2-ears, and the 2-ears form a forest.

8.1.5 Well-oriented ear-decompositions

Given an ear-decomposition, let $\mathcal{F}$ be a subset of the pendant ears that form a forest. Let $ear(v)$ denote the index of the ear that contains v as an internal vertex. A *rooted orientation* of $\mathcal{F}$ is an orientation of the edges of the ears in $\mathcal{F}$ such that each connected component is an arborescence whose root is a vertex v with $ear(v)$ minimum. Then

every ear of $\mathcal{F}$ is a directed path. A *well-oriented ear-decomposition* consists of an ear-decomposition and a rooted orientation of a subset of pendant ears that form a forest. See Figure 8.4 (a) for an example; the dotted, gray 2-ears are pendant and have a rooted orientation.

We denote by $r(w)$ the root of the connected component of the branching of oriented edges that contains w. We say that an ear Q *enters* another ear P if $Q \in \mathcal{F}$ and there is an oriented edge (v, w) of Q such that $w \in \text{in}(P)$. If any ear enters P, we call P *entered*; other nontrivial ears are called *non-entered*. In particular, all oriented ears are pendant and hence non-entered.

8.1.6 Summary of new techniques and structure of this chapter

First, we will show that we can compute an s-t-tour that is short with respect to the LP value in the case where the distance of the vertices s and t is relatively small; this proof will constitute most of this chapter (Section 8.2 – Section 8.6). Then we use this to characterize instances with $\frac{\text{OPT}}{\text{LP}}$ close to $\frac{3}{2}$ (Section 8.7). Finally, we reduce the s-t-path graph TSP to the special case where the distance of s and t is relatively small and obtain a 1.497-approximation algorithm (Section 8.8).

Although our proof can be viewed as a refined version of [SV14], we need many new ideas, some of which may be of independent interest or have further applications.

In Section 8.2, we describe a more sophisticated ear induction, and we assume that we have a well-oriented ear-decomposition in which the oriented ears are precisely the short ears. In particular, we assume the short ears to be pendant and form a forest. If we take all edges of the short ears before doing ear-induction on the long ears, some internal vertices of the long ears will already be connected via short ears. To use this connectivity service of the short ears, we exploit their orientation and reveal a novel connection to matroid union (Section 8.2.2). This saves in many cases but does not always help. Therefore, we also propose a second new way to benefit from the 2-ears: instead of taking a 2-ear as it is, one can also double one edge and discard the other, changing the parity at the endpoints of the 2-ear. Combining those two different possibilities of exploiting the short ears, either for connectivity or for parity, we obtain the main result of Section 8.2. Our ear induction algorithm saves at least $\frac{1}{26}$ for every non-entered ear, compared to $\frac{3}{2}(n-1)$, unless most of the long ears are 4-ears (Theorem 8.22).

Therefore, in addition to the properties of short ears, we need an ear-decomposition with extra properties of 4-ears. In Section 8.3 we show that one can always obtain such an ear-decomposition in polynomial time. In particular there will be only four types of 4-ears: pendant, blocked (with a closed ear attached to it), horizontal, or vertical (to be defined later), and at most one third of the long ears can be blocked 4-ears.

Then, in Section 8.4 we re-design short ears but also vertical 4-ears. Again we can use matroid intersection: one matroid is again graphic, but the other one is now a laminar matroid (instead of a partition matroid as in [SV14]). We can raise the lower bound not only for short ears that are not part of the forest, but also for horizontal and vertical 4-ears.

By this we remove the assumptions that the short ears form a forest and there are not too many 4-ears. See Section 8.5. We are done unless there are only few non-entered ears. Then there are few nontrivial ears at all because every entered ear is entered by a non-entered (short) ear. But then it is quite easy to obtain a better approximation

ratio than $\frac{3}{2}$ if in addition there is a short s-t-path P in G. To see this, let G' result from G by deleting the trivial ears (note that $|E(G')|$ is $n-1$ plus the number of nontrivial ears), and let the vector $x \in \mathbb{R}^{E(G)}$ be the sum of the incidence vectors of P and G', both multiplied by $\frac{1}{3}$. Then one can easily show that x is in the convex hull of T-joins for $T = \{s\} \triangle \{t\} \triangle \operatorname{odd}(E(G'))$, and hence adding a minimum T-join to G' results in an s-t-tour with $\frac{4}{3}|E(G')| + \frac{1}{3}|E(P)|$ edges. One can do a bit better by applying the removable-pairing technique of Mömke and Svensson [MS16] in a slightly novel way; see Section 8.6. We prove a variant of their lemma for well-oriented ear-decompositions that works without the 2-vertex-connectivity assumption.

At this point we have an algorithm that has an approximation ratio better than $\frac{3}{2}$ if the distance of s and t is relatively small compared to the length OPT of an optimum s-t-tour. Moreover, our algorithm computes a solution of cost significantly less that $\frac{3}{2} \cdot \mathrm{LP}$ if the distance of s and t is small not only compared to OPT, but also to LP. We use this to prove necessary conditions for instances with a large integrality ratio $\frac{\mathrm{OPT}}{\mathrm{LP}}$; see Section 8.7.

In order to obtain our 1.497-approximation algorithm, the only remaining case is when the distance from s to t in G is large, i.e. close to $\frac{\mathrm{OPT}}{2}$. To handle this case we use the following general statement from [Tra17] about approximation algorithms for the s-t-path TSP, not only applying to the graph case: for finding an α-approximation algorithm for some constant $\alpha > 1$, it is sufficient to consider the special case where the distance of the vertices s and t is at most $\frac{1}{3} + \varepsilon$ times the cost of an optimum solution for some arbitrary constant $\varepsilon > 0$. The proof of this reduction uses recursive dynamic programming similar to Blum et al. [BCK$^+$07] and Traub and Vygen [TV19a].

The case in which our ear-decomposition has only very few non-entered ears is the only case in which we do not compare our solution to the optimum LP value but to the optimum s-t-tour. Here, the dynamic programming algorithm allows us to bound the number of edges of our s-t-tour with respect to OPT (rather than the LP value), which we need to obtain an approximation ratio below the integrality ratio of the LP.

The different sections of this chapter can be read mostly independently of each other. Later sections make use only of the main result of previous sections; summarized in one theorem each: Theorem 8.22 states the result of our ear induction algorithm described in Section 8.2, Theorem 8.24 states the properties of the initial ear-decomposition that we construct in Section 8.3, and Theorem 8.34 gives the optimized and well-oriented ear-decomposition and the raised lower bound, as shown in Section 8.4. The rest of the chapter will be relatively short. In Section 8.5 we combine the previous sections to obtain a good bound if we have many non-entered ears in our well-oriented ear-decomposition (Theorem 8.35). In Section 8.6 we use the removable-pairing technique for the case where there are few non-entered ears and the distance of s and t is small. Combining these results we then obtain a 1.497-approximation on instances in which the distance of s and t is small (Theorem 8.38).

In Section 8.7, we characterize instances with integrality ratio close to $\frac{3}{2}$ (and many vertices). Finally, in Section 8.8, we give a 1.497-approximation for the general case of the s-t-path graph TSP. Here we use dynamic programming to reduce to the case where the distance of s and t is small (Theorem 8.42).

Let us review the overall 1.497-approximation algorithm: first use the reduction to the case where s and t have small distance (Theorem 8.42). To solve this case,

first compute an initial ear-decomposition as in Theorem 8.24. Based on this, compute an optimized and well-oriented ear-decomposition as in Theorem 8.34. If there are many non-entered ears, we get a short tour by enhanced ear induction (Theorem 8.22). If there are few non-entered ears, we obtain a short tour by the removable-pairing technique; see Lemma 8.37. Our presentation follows a different order because each section is motivated by the previous ones.

8.2 Enhanced ear induction

8.2.1 Outline of our ear induction algorithm

In this section we describe an ear induction algorithm that computes a T-tour, where $T \subseteq V(G)$ is a given even-cardinality set. Our goal is to obtain an upper bound on the number of edges where we gain some constant amount per non-entered ear, compared to $\frac{3}{2}(n-1)$.

Recall that the clean ears are the short ears P with $\mathrm{in}(P) \cap T = \emptyset$. For the entire Section 8.2 we will assume that we are given a well-oriented ear-decomposition in which all short ears are clean and the oriented ears are precisely the clean ears. In particular, the clean ears are all pendant and form a forest. Later (in Section 8.4) we consider the general case.

Let $G_\gamma = (V(G), E_\gamma)$ be the spanning subgraph of G that contains only the edges of clean ears. Due to the rooted orientation this is a branching. Every connected component of G_γ will be used either for connectivity or for parity correction. If we use a connected component of G_γ for connectivity, we add all edges of the component to our T-tour. However, we can instead use a component C of G_γ consisting of only 2-ears for parity correction as follows. Let T' be a set of vertices that are contained in the component C, but are not internal vertices of short ears (see Figure 8.2). If $|T'|$ is even, we can change the parity of exactly the vertices in T' by "flipping" the 2-ears that are part of a T'-join in C, i.e. we take two copies of one edge instead of one copy of each of the edges of the "flipped" 2-ears (see Figure 8.2). As a consequence, we can choose the parity of all vertices that have an entering clean ear in C and then fix the parity at the root of the component C such that the set T' of vertices where we need to change the parity of the degree has even cardinality. We can also flip 3-ears, but then we need four instead of three edges.

If we could bound the number of edges that we need during ear induction (as in Section 8.1.3) for every ear P by $\frac{3}{2}|\mathrm{in}(P)|$, we would obtain a T-tour with at most $\frac{3}{2}(n-1)$ edges. Lemma 8.1 yields an even better bound for long ears (of length at least four); so we can gain some constant amount per long ear. However, for (clean) 2-ears we need two edges, which is $\frac{1}{2}$ more than $\frac{3}{2}|\mathrm{in}(P)|$, and also for (clean) 3-ears we can not improve over $\frac{3}{2}|\mathrm{in}(P)| = 3$. To make up for this, we would like to improve over $\frac{3}{2}|\mathrm{in}(P)|$ for long ears by some constant amount for each short (oriented) ear entering the ear P. In order to gain from a short ear entering P at some vertex w, we then either exploit that the clean ears connect w to the root $r(w)$ (if the component of G_γ containing w is used for connectivity) or make use of the fact that we can choose the parity of the vertex w (by possibly changing the parity at $r(w)$).

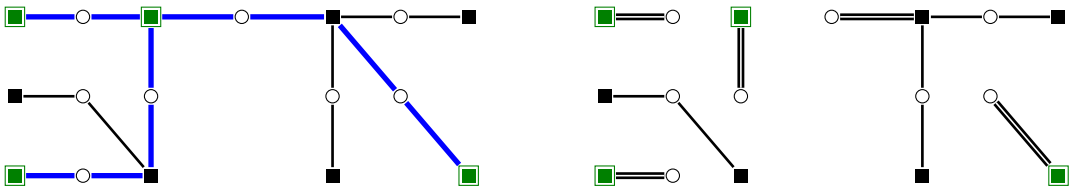

Figure 8.2: Using 2-ears for parity correction: The filled squares denote internal vertices of long ears, the circles internal vertices of 2-ears. Edges of long ears and the orientation of short ears are not shown here. The vertex set T' is shown in green with frames. The thick blue edges (in the left picture) show a T'-join in this component of G_γ. The right picture shows the same component after "flipping" the T'-join. Compared to the left picture, the parity of the degree is changed precisely for the vertices in T' .

8.2.2 Using clean ears for connectivity via matroid union

In this section we use the matroid union theorem (Theorem 2.14) to prove our main lemma for enhanced ear induction. We obtain a better bound than Lemma 8.1 in many cases by making use of the contribution of the clean ears to connectivity. In the proof we will need a statement about matroids that follows from the matroid union theorem. Recall the contraction operation in matroids: if $\mathcal{M} = (E, \mathcal{F})$ is a matroid and $F \in \mathcal{F}$ is an independent set, then $\mathcal{M}/F := (E \setminus F, \{Z \subseteq E \setminus F : Z \cup F \in \mathcal{F}\})$ is well-known to be a matroid.

Lemma 8.2. *Let $\mathcal{M} = (E, \mathcal{F})$ be a matroid with rank function r, and a partition of E into sets R, B, and U (red, blue, and uncolored). Then there is a partition of U into sets X and Y such that $r(R \cup X) + r(B \cup Y) \geq r(R \cup B) + r(U)$. Such a partition can be found in polynomial time.*

Proof. Let $R' \subseteq R$, $B' \subseteq B$, and $U' \subseteq U$ such that $r(R \cup B) = r(R' \cup B') = |R' \cup B'|$ and $r(U) = r(U') = |U'|$.

For every $S \subseteq U'$ we have

$$\begin{aligned} |U' \setminus S| + r_{\mathcal{M}/R'}(S) + r_{\mathcal{M}/B'}(S) &= |U' \setminus S| + r(S \cup R') - |R'| + r(S \cup B') - |B'| \\ &\geq |U'| - |S| + r(S) + r(S \cup R' \cup B') - |R'| - |B'| \\ &= |U'| + r(S \cup R' \cup B') - |R' \cup B'| \\ &\geq |U'|. \end{aligned}$$

We used submodularity of r in the first inequality. By the matroid union theorem ([Edm68], Theorem 2.14), the minimum of the left-hand side over all $S \subseteq U'$ is the rank of U' in the union of $\mathcal{M}/R'$ and $\mathcal{M}/B'$ (which is also a matroid). Hence there is a partition $U' = X' \dot\cup Y'$ such that X' is independent in $\mathcal{M}/R'$ and Y' is independent in $\mathcal{M}/B'$. Such a partition can be found by a matroid union algorithm; see e.g. Section 42.3 in Schrijver's book [Sch03].

Then any partition $U = X \dot\cup Y$ with $X' \subseteq X$ and $Y' \subseteq Y$ satisfies

$$\begin{aligned} r(R \cup X) + r(B \cup Y) &\geq r(R' \cup X') + r(B' \cup Y') \\ &= |X'| + |R'| + |Y'| + |B'| \\ &= |R' \cup B'| + |U'| \\ &= r(R \cup B) + r(U). \end{aligned}$$

□

The special case when $R \cup B$ and U are bases of $\mathcal{M}$ is a well-known theorem by Brylawski [Bry73], Greene [Gre73], and Woodall [Woo74]; see (42.13) in Schrijver's book [Sch03].

We apply the lemma in the following context:

Lemma 8.3. *Let (V, E) be a graph (possibly with parallel edges) and a partition of E into nonempty sets R, B, U such that (V, U) is a forest and $(V, R \cup B)$ is a circuit. Then there is a partition of U into sets U_R, U_B, and Z such that $(V, R \cup U_R)$ is a forest, $(V, B \cup U_B)$ is a forest, and Z contains at most one element. Such a partition can be found in polynomial time.*

Proof. Apply Lemma 8.2 to the cycle matroid of (V, E); for its rank function r we have $r(R \cup B) + r(U) = |R| + |B| + |U| - 1$. We get a partition $U = X \dot\cup Y$ with $r(R \cup X) + r(B \cup Y) \geq |R \cup X| + |B \cup Y| - 1$.

If $r(R \cup X) = |R \cup X|$, set $U_R := X$. If $r(R \cup X) = |R \cup X| - 1$, there is an element $z \in X$ such that $R \cup (X \setminus \{z\})$ is independent because (V, R) is a forest; then set $U_R := X \setminus \{z\}$.

Set U_B analogously, and $Z := U \setminus (U_R \cup U_B)$. □

An *antipodal pair* in a circuit P is a set of two vertices of P that have distance $\frac{|E(P)|}{2}$ in P. Obviously, only even circuits have antipodal pairs. We use the following lemma to exploit the connectivity service of clean ears during ear induction. The clean ears entering P are represented by the edge set U in this lemma. A subset C of U will be used for connectivity.

Lemma 8.4. *Let P be a circuit with at least four edges and $T_P \subseteq V(P)$ with $|T_P|$ even. Let $(V(P), U)$ be a forest. Then one of the following is true:*

(i) *$|E(P)| = 4$ and $|U| = 1$ and $T_P = \emptyset$;*

(ii) *T_P is an antipodal pair, and U consists of a single edge with endpoints T_P;*

(iii) *There exists a T_P-join $F \subseteq 2E(P)$ and a subset $C \subseteq U$ such that the graph $(V(P), F \cup C)$ is connected,*

$$|F| \leq \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}|U| - \tfrac{1}{2}\max\{1, |U| - 1\}, \tag{8.2}$$

and $|C| \leq 2\left(\frac{3}{2}(|E(P)| - 1) - \frac{1}{2}|U| - |F|\right)$. Such an F can be found in polynomial time.

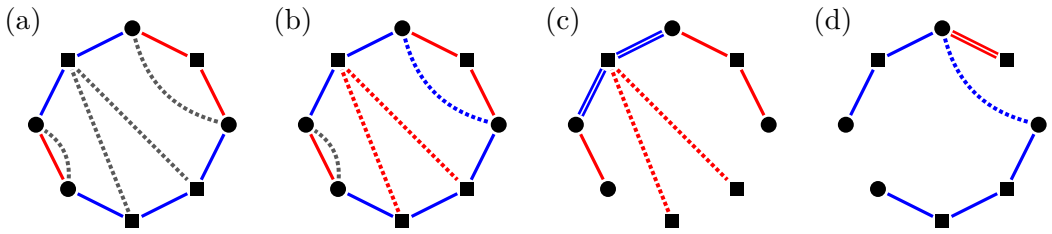

Figure 8.3: Enhanced ear induction in Lemma 8.4: (a) ear P is drawn with solid lines, dotted lines indicate the edges in U, black squares are elements of $V(P) \setminus T_P$; black circles are elements of T_P; (b) coloring the edges of U red and blue; one edge remains uncolored; (c) the red solution F_R; (d) the blue solution F_B.

Proof. Similarly to the proof of Lemma 8.1, we distinguish two cases.

Case 1: $T_P \neq \emptyset$.

The vertices of T_P subdivide P into subpaths, alternatingly colored red and blue. Let E_R and E_B denote the set of edges of red and blue subpaths, respectively. Let $T_R := \mathrm{odd}(E_R)$ and $T_B := \mathrm{odd}(E_B)$ be the set of vertices having odd degree in $(V(P), E_R)$ and $(V(P), E_B)$, respectively. Note that $\{E_R, E_B\}$ is a partition of $E(P)$, both sets are nonempty, and $T_R = T_B = T_P$. Color the edges in U red and blue according to Lemma 8.3; one edge may remain uncolored. Let C be the set of colored edges in U. See Figure 8.3.

We consider two solutions: To construct F_R, we take E_R plus two copies of some edges of E_B. Since E_R plus the red elements of U form a forest, the number of blue edges needed for connectivity (with two copies each) is at most $|E_B| - 1$ minus the number of red elements of U. To construct F_B, we exchange the roles of red and blue.

The smaller of the two has at most

$$\begin{aligned} \tfrac{1}{2}(3|E(P)| - 4 - 2|C|) &\leq \tfrac{1}{2}(3|E(P)| - 4 - (|U| - 1) - |C|) \\ &= \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}|U| - \tfrac{1}{2}|C| \end{aligned}$$

edges. Since $|C| \geq |U| - 1$, we are done if $|U| > 1$ or $|U| = |C| = 1$.

Now let $|U| \leq 1$. If $U = \emptyset$, then the smaller of the sets F_B and F_R has size at most

$$\tfrac{1}{2}(3|E(P)| - 4) = \tfrac{3}{2}(|E(P)| - 1) - \frac{1}{2}.$$

Now consider the remaining case that $|U| = 1$ and $C = \emptyset$, i.e., the only edge in U cannot be colored. This means that the endpoints of this edge are connected by a path in E_R and a path in E_B, and hence T_P consists of exactly these two elements. If (ii) does not hold, T_P is not an antipodal pair, and hence $|E_R| \neq |E_B|$. Then the smaller of the sets F_B and F_R has size at most

$$\begin{aligned} |E(P)| + \min\{|E_R|, |E_B|\} - 2 &\leq \tfrac{3}{2}|E(P)| - \tfrac{1}{2} - 2 \\ &= \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}|U| - \tfrac{1}{2}. \end{aligned}$$

Case 2: $T_P = \emptyset$.

We can set $F = E(P)$ and $C = \emptyset$, but instead we can also set $C = U$ and take all but $|U| + 1$ edges, each with two copies, making $2(|E(P)| - |U| - 1)$ edges. The smaller of the two choices for F has at most

$$\tfrac{1}{2}(3|E(P)| - 2|U| - 2) \;=\; \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}|U| - \tfrac{1}{2}(|U| - 1)$$

edges. If $|U| > 1$, this implies (8.2). If $|U| > 1$ and both solutions have the same number of edges, $F = E(P)$ and $C = \emptyset$ fulfills (iii). If $|U| > 1$ and one of the two solutions for F has fewer edges, the smaller of the two choices for F has at most

$$\tfrac{1}{2}(3|E(P)| - 2|U| - 3) \;=\; \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}|U| - \tfrac{1}{2}|U|$$

edges. Then we also have

$$|C| \;\leq\; |U| \;=\; 2\left(\tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}|U| - |F|\right).$$

If $|U| \leq 1$, we have (i) or $|E(P)| - 1 \geq 3 + |U|$. In the latter case,

$$|E(P)| \;\leq\; \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}|U| - \tfrac{1}{2}.$$

Hence, we can set $F = E(P)$ and $C = \emptyset$. □

8.2.3 Bad, special, and good ears

In this section we describe our ear induction algorithm that makes use of the short ears for connectivity. We may assume that the short ears come last in the ear-decomposition because they are all pendant. Let $P_1, \ldots, P_l$ be the long ears of our ear-decomposition, i.e., the ears of length at least four. Recall that G_i is the graph composed of the first i ears, and V_i denotes its vertex set.

Roughly speaking, a long ear is good if we can apply Lemma 8.4 (iii) to it. Let us consider the exceptions. We again refer to our orientation of the clean ears.

Definition 8.5. *Call a pair (P, T_P) for an ear P and a set $T_P \subseteq \mathrm{in}(P)$ a* bad pair *if exactly one clean ear Q enters P, $|E(P)|$ is even and P fulfills one of the following properties:*

(a) $|E(P)| = 4$ *and* $T_P = \emptyset$,

(b) $|E(P)| > 4$, *Q enters P at its middle internal vertex w, $T_P = \{w\}$, and $r(w)$ is not an internal vertex of P.*

Definition 8.6. *Let P be an even ear with at least six edges such that exactly one clean ear Q enters P. Denote by w the internal vertex of P where Q enters P. If $r(w) \in \mathrm{in}(P)$ and if $r(w)$ and w have distance $\frac{|E(P)|}{2}$ in P, we call the pair $(P, \{r(w), w\})$* special.

Call a pair (P, T_P) for a long ear P and a set $T_P \subseteq \mathrm{in}(P)$ a good pair *if it is neither bad nor special.*

We now describe an algorithm that computes a short T-tour if we have many ears that are good or special. To this end we first use ear induction to obtain a short T-tour

if many ears are good and we can thus apply Lemma 8.4 (iii) often. We then show that we can afterwards improve the resulting T-tour if there are many special ears (cf. the proof of Lemma 8.11).

For a long ear P_i (with $i \in \{1, \dots, l\}$) let h_i denote the number of clean ears entering P_i. For a multi-set $F \subseteq 2E(P_i)$ let

$$\text{gain}_i(F) \;:=\; \tfrac{3}{2}|\text{in}(P_i)| - \tfrac{1}{2}h_i - |F|.$$

Lemma 8.7. *Let $F_i \subseteq 2E(P_i)$ be a multi-set for every $i = 1, \dots l$. Then*

$$|E_\gamma| + \sum_{i=1}^{l} |F_i| \;=\; \tfrac{3}{2}(n-1) - \tfrac{1}{2}k_3 - \sum_{i=1}^{l} \text{gain}_i(F_i).$$

Proof.

$$\begin{aligned}
|E_\gamma| + \sum_{i=1}^{l} |F_i| \;&=\; 2 \cdot k_2 + 3 \cdot k_3 + \sum_{i=1}^{l} |F_i| \\
&=\; 2 \cdot k_2 + 3 \cdot k_3 + \sum_{i=1}^{l} \Big(\tfrac{3}{2}|\text{in}(P_i)| - \tfrac{1}{2}h_i - \text{gain}_i(F_i) \Big) \\
&=\; \sum_{P \text{ short ear}} \tfrac{3}{2}|\text{in}(P)| + \tfrac{1}{2}k_2 + \sum_{i=1}^{l} \Big(\tfrac{3}{2}|\text{in}(P_i)| - \tfrac{1}{2}h_i \Big) - \sum_{i=1}^{l} \text{gain}_i(F_i) \\
&=\; \tfrac{3}{2}(n-1) - \tfrac{1}{2}k_3 - \sum_{i=1}^{l} \text{gain}_i(F_i).
\end{aligned}$$

□

For a subset C of the clean ears, let $V_\gamma(C)$ and $E_\gamma(C)$ be the union of the vertex sets and edge sets, respectively, of all connected components of G_γ that contain a clean ear in C. Let $T_l := T \triangle \text{odd}(E_\gamma) \subseteq V_l$. If we take all the edges of clean ears, then T_l is the set of vertices for which we still need an odd number of incident edges to obtain a T-tour. We consider the long ears in reverse order, starting from P_l and apply the following lemma to obtain a multi-set $F_i \subseteq 2E(P_i)$ and a set C_i of clean ears. The set C_i contains those clean ears whose connectivity service we use.

Lemma 8.8. *Given a set $T_i \subseteq V_i$ with $|T_i|$ even, we can construct a multi-set $F_i \subseteq 2E(P_i)$ and a set C_i of clean ears such that*

(i) $\text{odd}(F_i) \cap \text{in}(P_i) = T_i \cap \text{in}(P_i)$, *and*

(ii) $(V_i \cup V_\gamma(C_i), F_i \cup E_\gamma(C_i))/V_{i-1}$ *is connected, and*

(iii) $\text{gain}_i(F_i) \geq 0$, *and*

(iv) $|C_i| \leq 2 \cdot \text{gain}_i(F_i)$.

(v) *Moreover, we have:*

- *if $(P_i, T_i \cap \text{in}(P_i))$ is good, then $\text{gain}_i(F_i) \geq \frac{1}{2}\max\{1, h_i - 1\}$;*
- *if $(P_i, T_i \cap \text{in}(P_i))$ is special, then $\text{gain}_i(F_i) = 0$, $C_i = \emptyset$, and the two edges of P_i incident to an endpoint of P_i are contained exactly once in F_i.*

With such an F_i we define

$$T_{i-1} \;:=\; T_i \triangle \operatorname{odd}(F_i).$$

Note that $T_{i-1} \subseteq V_{i-1}$ due to (i). Moreover, since the symmetric difference of two even-cardinality sets is even, $|T_{i-1}|$ is even. We call an ear P_i good/special/bad if $(P_i, T_i \cap \operatorname{in}(P_i))$ is a good/special/bad pair.

We keep track of the clean ears C_i used for connectivity, since we will later make other use of the connected components of G_γ that we did not use for connectivity. Some of those components can be used to help parity correction and save edges in special ears. This will happen in a post-processing step after ear induction, in the proof of Lemma 8.11.

Proof of Lemma 8.8: If $(P_i, T_i \cap \operatorname{in}(P_i))$ is a bad pair, we set $C_i := \emptyset$. To obtain F_i we then apply Lemma 8.1 to $(G_i, T_i)/V_{i-1}$. (Then the circuit P is G_i/V_{i-1}.)

If $(P_i, T_i \cap \operatorname{in}(P_i))$ is special, $T_i \cap \operatorname{in}(P_i)$ contains exactly two vertices w, r which have distance $\frac{|E(P_i)|}{2}$ in P_i. To construct F_i we take $E(P_i)$, double all edges of the w-r-path in P_i and remove both copies of one duplicated edge. If P_i is closed, we take the w-r-path that does not contain the endpoint of P_i. Then $|F_i| = \frac{3}{2}|\operatorname{in}(P_i)| - \frac{1}{2} = \frac{3}{2}|\operatorname{in}(P_i)| - \frac{1}{2}h_i$, implying $\operatorname{gain}_i(F_i) = 0$. See the brown ear in Figure 8.4 (b) for an example.

If $(P_i, T_i \cap \operatorname{in}(P_i))$ is good, we will apply Lemma 8.4. To this end, let (P, T_P) be the instance $(G_i, T_i)/V_{i-1}$. (Then again, the circuit P is G_i/V_{i-1}.) We now define the set U: Let $Q_1, \dots, Q_h$ be the clean ears entering internal vertices $w_1, \dots, w_h$ of P_i.

We then set U to be the set of edges resulting from $\{\{r(w_1), w_1\}, \dots, \{r(w_h), w_h\}\}$ by contracting V_{i-1}. Note that the vertex set of the circuit P consists of $\operatorname{in}(P_i)$ and the vertex arising from the contraction of V_{i-1}. Since the vertices $r(w_1), \dots, r(w_h)$ are all contained in V_i (by choice of the orientation of G_γ), all endpoints of the edges in U are vertices of $P = G_i/V_{i-1}$. See Figure 8.4 (c) for an example.

In order to apply Lemma 8.4, we need that $(V(P), U)$ is a forest. We orient every edge in U resulting from $\{r(w_j), w_j\}$ from $r(w_j)$ to w_j. Since all vertices w_j for $j \in \{1, \dots, h\}$ are internal vertices of the ear P_i, the vertex in P arising from the contraction of V_{i-1} has no incoming edge. Since G_γ is a branching, no vertex in $P = G_i/V_{i-1}$ has more than one incoming edge in the oriented edge set U. Now suppose $(V(P), U)$ contains an undirected circuit. Since every vertex in $(V(P), U)$ has at most one entering (directed) edge, the undirected circuit in $(V(P), U)$ must be also a directed circuit.

Now note that for $j, j' \in \{1, \dots, h\}$, the vertex $w_j \in \operatorname{in}(P_i)$ can not be the root $r(w_{j'})$ of the connected component of G_γ containing $w_{j'}$ since w_j has an entering clean ear Q_j in G_γ. Thus, a vertex in $(V(P), U)$ that has an entering (directed) edge cannot have an outgoing edge. This contradicts the fact that $(V(P), U)$ contains a directed circuit. Hence, $(V(P), U)$ must be a forest and we can apply Lemma 8.4 to obtain a set $F \subseteq 2E(P) = 2E(P_i)$ and a set $C \subseteq U$.

We set $F_i := F$ and set $C_i := \{Q_j : \{r(w_j), w_j\} \in C\}$. Then by Lemma 8.4, we have

- $\operatorname{odd}(F_i) \cap \operatorname{in}(P_i) = T_i \cap \operatorname{in}(P_i)$,
- $\operatorname{gain}_i(F_i) \geq \frac{1}{2} \max\{1, h_i - 1\}$, and
- $|C_i| = |C| \leq 2\left(\frac{3}{2}(|E(P)| - 1) - \frac{1}{2}h_i - |F_i|\right) = 2 \cdot \operatorname{gain}_i(F_i)$.

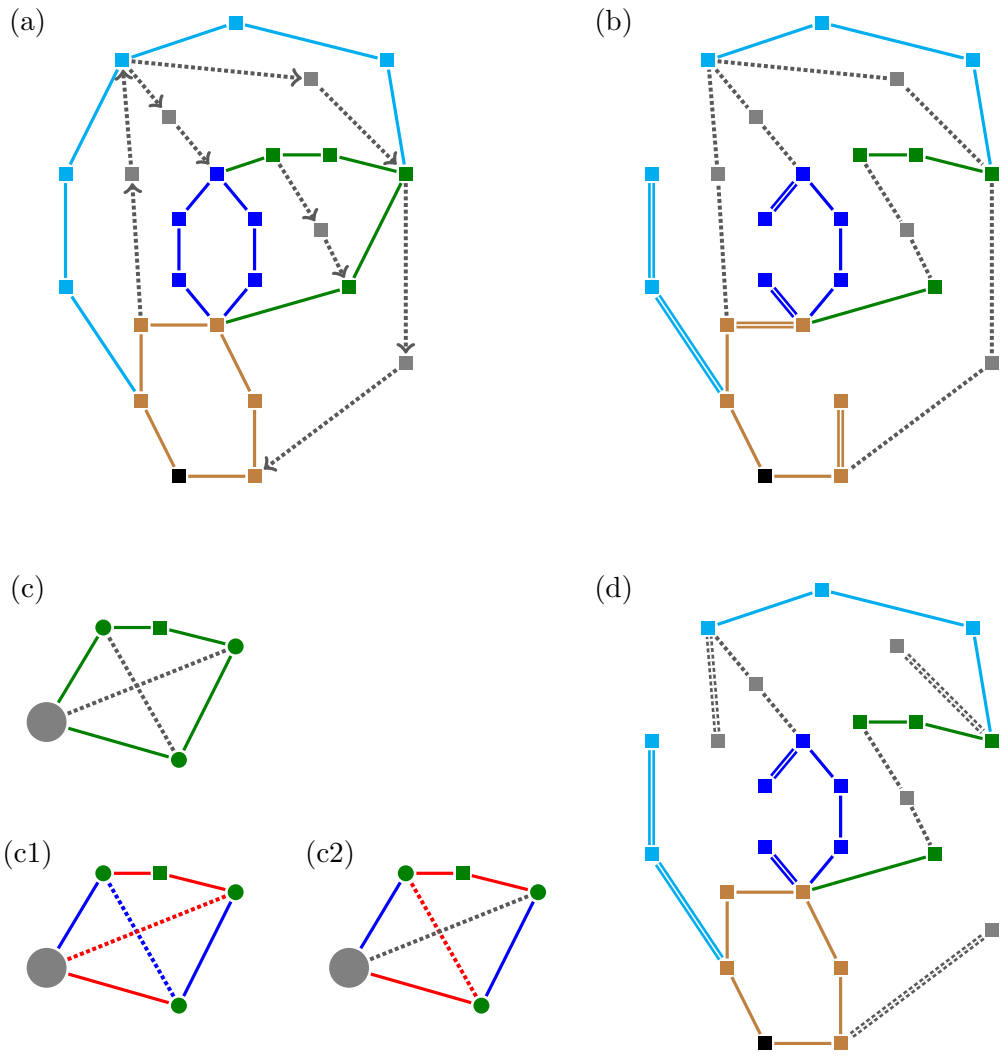

Figure 8.4: An example with $T = \emptyset$. (a): ears with oriented short ears (gray, dotted). (b): result of enhanced ear induction, using short ears for connectivity. The cyan ear is bad, the green ear is good, the blue ear is bad, and with the choices made as in the figure, the brown ear is special. (c): Applying Lemma 8.4 to the good (green) ear. (c1) and (c2) show two different ways to color the edges in U. In (c2) one edge remains uncolored (although it would be possible to color it). (d): The T-tour after applying the modification described in the proof of Lemma 8.11 if the coloring as in (c2) is chosen. If the coloring of the set U is the one shown in (c1), the algorithm described in the proof of Lemma 8.11 won't modify the T-tour.

It remains to prove that $(V_i \cup V_\gamma(C_i), F_i \cup E_\gamma(C_i))/V_{i-1}$ is connected. Recall that $V_\gamma(C_i)$ and $E_\gamma(C_i)$ contain the vertex set and the edge set, respectively, of all connected components of G_γ that contain a clean ear in C_i. Hence, for every edge $\{r(w_j), w_j\} \in C$, the set $E_\gamma(C_i)$ contains the edge set of the $r(w_j)$-w_j-path in G_γ. Since by Lemma 8.4, $(V(P), F \cup C)$ is connected, also $(V_i \cup V_\gamma(C_i), F_i \cup E_\gamma(C_i))/V_{i-1}$ is connected. □

Lemma 8.9. *$E_\gamma \dot\cup F_1 \dot\cup \ldots \dot\cup F_l$ is a T-join.*

Proof. The set T_{i-1} is defined such that F_i is a $(T_i \triangle T_{i-1})$-join. By induction on $l-i$, the set $F_i \dot\cup \ldots \dot\cup F_l$ is a $(T_{i-1} \triangle T_l)$-join. For $i = 1$, this implies that $F_1 \dot\cup \ldots \dot\cup F_l$ is a $(T_0 \triangle T_l)$-join. As F_i is constructed such that $T_i \subseteq V_{i-1}$ for all $i \in \{1, \ldots, l+1\}$, we have $T_0 \subseteq V_0$.

The set $T_0 \triangle T_l$ contains an even number of vertices (since a $(T_0 \triangle T_l)$-join exists), and $|T_l|$ is even. Hence, $|T_0|$ must be even. Since V_0 has exactly one element, T_0 has at most one element. As $|T_0|$ is even, $T_0 = \emptyset$ and $F_1 \dot\cup \ldots \dot\cup F_l$ is a T_l-join. By definition of T_l, adding the edges E_γ to $F_1 \dot\cup \ldots \dot\cup F_l$ results in a T-join. □

Lemma 8.10. *Let*

$$\bar{V}_\gamma := \bigcup_{i=1}^{l} V_\gamma(C_i) \quad \text{and} \quad \bar{E}_\gamma := \bigcup_{i=1}^{l} E_\gamma(C_i).$$

Then $(V_l \cup \bar{V}_\gamma, F_1 \dot\cup \ldots \dot\cup F_l \dot\cup \bar{E}_\gamma)$ is connected.

Proof. We prove by induction on $l-i$ that for every $i = 1, \ldots, l+1$, the graph resulting from $(V_l \cup \bar{V}_\gamma, F_i \dot\cup \ldots \dot\cup F_l \dot\cup \bar{E}_\gamma)$ by contracting V_{i-1} is connected. For $i = 1$ the set $V_{i-1} = V_0$ contains only one element, hence this completes the proof.

For $i = l+1$ the set $V_{i-1} = V_l$ contains the roots of all connected components of G_γ, hence $(V_l \cup \bar{V}_\gamma, \bar{E}_\gamma)/V_l$ is connected. Now let $i \leq l$. By induction hypothesis, for every vertex $v' \in V_l \setminus V_i$ the set $F_{i+1} \cup \cdots \cup F_l \cup \bar{E}_\gamma$ contains the edge set of a v'-w'-path for some $w' \in V_i$. Hence, it suffices to show that for every $v \in \mathrm{in}(P_i)$ the set $F_i \dot\cup \ldots \dot\cup F_l \dot\cup \bar{E}_\gamma$ contains a path from v to a vertex in V_{i-1}. This is the case since the graph $(V_i \cup V_\gamma(C_i), F_i \cup E_\gamma(C_i))/V_{i-1}$ is connected by Lemma 8.8 (ii). □

We now describe a post-processing step that deals with special ears. Let P be a special ear. If the connected component of G_γ that contains the clean ear entering P at w has not been used for connectivity, we will use it to save edges in P by "flipping" the $r(w)$-w-path in G_γ. See Figure 8.5.

Lemma 8.11. *Given a well-oriented ear-decomposition with long ears $P_1, \ldots, P_l$ where all short ears are clean and the oriented ears are precisely the clean ears, we can compute a T-tour with at most*

$$\tfrac{3}{2}(n-1) - \tfrac{1}{2}k_3 - \sum_{i=1}^{l} \mathrm{gain}_i(F_i) - \max\left\{0, k_{\text{special}} - k_3 - \sum_{i=1}^{l} 2 \cdot \mathrm{gain}_i(F_i)\right\}$$

edges, where k_{special} denotes the number of special pairs $(P_i, T_i \cap \mathrm{in}(P_i))$.

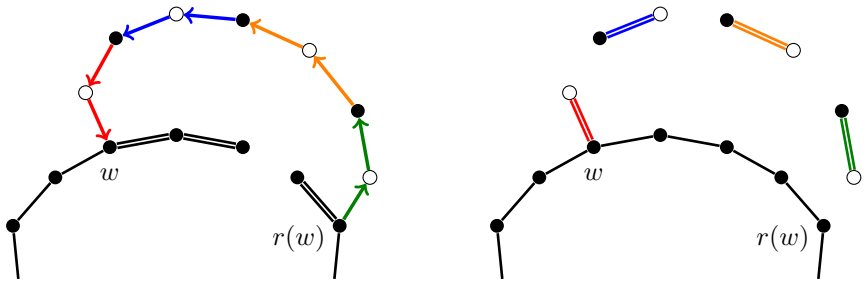

Figure 8.5: Modifying the T-tour for special ears as in the proof of Lemma 8.11. The edges shown in black are edges in $E(P_i)$, the colored edges are edges of (pendant) 2-ears that are part of the $r(w)$-w-path. The edges of different 2-ears are shown in different colors. The filled vertices are internal vertices of long ears, i.e. these vertices are contained in V_l. The non-filled vertices are internal vertices of (pendant) 2-ears. The left picture shows the edges of $E(P)$ and the $r(w)$-w-path used in the T-tour before modifying it, and the right picture shows the used edges after the modification.

Proof. By Lemma 8.9 and Lemma 8.10, $E_\gamma \dot\cup F_1 \dot\cup \ldots \dot\cup F_l$ is a T-tour and by Lemma 8.7 we can bound the number of edges by

$$|E_\gamma| + \sum_{i=1}^{l} |F_i| \;=\; \tfrac{3}{2}(n-1) - \tfrac{1}{2}k_3 - \sum_{i=1}^{l} \mathrm{gain}_i(F_i).$$

Note, that for any special ear P and vertex $w \in \mathrm{in}(P)$, the root $r(w)$ of the connected component of G_γ containing w is contained in $\mathrm{in}(P)$. (This follows from the definition of a special ear.) Thus, any connected component of G_γ contains internal vertices of at most one special ear.

We now modify the T-tour for every special ear P_i as follows: The ear P_i has exactly one entering clean ear Q. Let w be the internal vertex of P_i where Q enters P_i. By the definition of a special ear, we have $r(w) \in \mathrm{in}(P_i)$ and w and $r(w)$ have distance $\frac{|E(P_i)|}{2}$ in P_i. The connected component of G_γ containing w (and $r(w)$) contains no internal vertex of any special ear distinct from P_i since any connected component of G_γ contains internal vertices of at most one special ear. If the connected component of G_γ containing w (and $r(w)$) contains neither an edge in $\bar{E}_\gamma$ nor the edge set of a 3-ear, we modify our T-tour by replacing F_i by $E(P_i)$ and replacing the $r(w)$-w-path in E_γ by two copies of every second edge on this path. Note that the edge set of this path is the union of edge sets of 2-ears. (See Figure 8.5.)

We now show that modifying the tour results in a T-tour with at least one edge less than before. Replacing the $r(w)$-w-path in G_γ by two copies of every second edge on this path does not change the parity of the vertex degrees at internal vertices of this path, but changes the parity of the vertex degrees of $r(w)$ and w. Recall that we constructed F_i such that the edges incident to endpoints of P_i are contained exactly once. Thus, replacing F_i by $E(P_i)$ does not change the parity of the degree of vertices not in $\mathrm{in}(P_i)$. By definition of a special ear $T_i \cap \mathrm{in}(P_i) = \{w, r(w)\}$, hence $\mathrm{odd}(F_i) \cap \mathrm{in}(P_i) = \{w, r(w)\}$. This shows that replacing F_i by $E(P_i)$ changes the parity of the vertex degree exactly at the vertices $r(w)$ and w. After replacing both F_i and the $r(w)$-w-path in G_γ the parity of all vertex degrees is the same as before, hence the resulting (multi-)edge set is a T-join.

Using Lemma 8.10, we thus get that after replacing the $r(w)$-w-path in G_γ all vertices in $V_l \cup \bar{V}_\gamma$ are still part of the same connected component. Moreover, we use at least one edge from every 2-ear and all edges from every 3-ear. Thus, we have indeed constructed a T-tour.

Note that replacing the $r(w)$-w-path in G_γ as described above does not change the total number of edges used from $2E_\gamma$. But we replaced F_i by $E(P_i)$, and since $|F_i| = \frac{3}{2}|\mathrm{in}(P_i)| - \frac{1}{2}$ (as we had $\mathrm{gain}_i(F_i) = 0$) and $|E(P_i)| \geq 6$ (because P_i is special), we decreased the number of edges by at least one.

This shows that we can decrease the number of edges of the T-tour by one for the special ear P_i, unless the connected component of G_γ containing w contains a 3-ear or an edge from $\bar{E}_\gamma$. Since every connected component of G_γ contains internal vertices of at most one special ear, we can modify our T-tour (in the way described above) for at least $k_{\text{special}} - k_3 - \sum_{i=1}^{l} |C_i|$ special ears, where k_{special} denotes the number of special ears. Thus, we obtain a T-tour with at most

$$\begin{aligned}
&\tfrac{3}{2}(n-1) - \tfrac{1}{2}k_3 - \sum_{i=1}^{l} \mathrm{gain}_i(F_i) - \max\left\{0,\ k_{\text{special}} - k_3 - \sum_{i=1}^{l} |C_i|\right\} \\
\leq\ &\tfrac{3}{2}(n-1) - \tfrac{1}{2}k_3 - \sum_{i=1}^{l} \mathrm{gain}_i(F_i) - \max\left\{0,\ k_{\text{special}} - k_3 - \sum_{i=1}^{l} 2 \cdot \mathrm{gain}_i(F_i)\right\}
\end{aligned}$$

edges; here we used Lemma 8.8 (iv). □

Lemma 8.12. *Given a well-oriented ear-decomposition with long ears $P_1, \ldots, P_l$ where all short ears are clean and the oriented ears are precisely the clean ears, we can compute a T-tour with at most*

$$\tfrac{3}{2}(n-1) - \tfrac{7}{20}k_3 - \sum_{i \in I} \max\left\{\tfrac{7}{20}(h_i - 1),\ \tfrac{3}{20}\right\}$$

edges, where $I := \{i \in \{1, \ldots, l\} : (P_i, T_i \cap \mathrm{in}(P_i))$ is good or special$\}$.

Proof.

$$\begin{aligned}
&\tfrac{1}{2}k_3 + \sum_{i=1}^{l} \mathrm{gain}_i(F_i) + \max\left\{0,\ k_{\text{special}} - k_3 - \sum_{i=1}^{l} 2 \cdot \mathrm{gain}_i(F_i)\right\} \\
\geq\ &\tfrac{7}{10}\left(\tfrac{1}{2}k_3 + \sum_{i=1}^{l} \mathrm{gain}_i(F_i)\right) + \tfrac{3}{20}\left(k_3 + \sum_{i=1}^{l} 2 \cdot \mathrm{gain}_i(F_i)\right) \\
&+ \tfrac{3}{20}\max\left\{0,\ k_{\text{special}} - k_3 - \sum_{i=1}^{l} 2 \cdot \mathrm{gain}_i(F_i)\right\} \\
\geq\ &\tfrac{7}{10}\left(\tfrac{1}{2}k_3 + \sum_{i=1}^{l} \mathrm{gain}_i(F_i)\right) + \tfrac{3}{20}k_{\text{special}} \\
\geq\ &\tfrac{7}{20}k_3 + \sum_{i \in I} \max\left\{\tfrac{7}{20}(h_i - 1),\ \tfrac{3}{20}\right\},
\end{aligned}$$

where we used $\text{gain}_i(F_i) \geq \max\{1, h_i - 1\}$ if $(P_i, T_i \cap \text{in}(P_i))$ is good and $h_i = 1$ if $(P_i, T_i \cap \text{in}(P_i))$ is special. Together with Lemma 8.11, this implies that we can compute a T-tour with at most

$$\tfrac{3}{2}(n-1) - \tfrac{7}{20}k_3 - \sum_{i \in I} \max\left\{\tfrac{7}{20}(h_i - 1),\ \tfrac{3}{20}\right\}$$

edges. □

Compared to $\frac{3}{2}(n-1)$, we have now gained for every good and special ear. More precisely, the resulting T-tour is short if there are many clean ears entering good or special ears (or many non-entered long ears; these are good ears). Next, we will deal with the bad ears.

8.2.4 Using clean ears for parity correction

If there are many bad ears, the T-tour resulting from Section 8.2.3 has too many edges. To deal with this case we compute a second T-tour, by a different kind of ear induction. In contrast to the ear induction in Section 8.2.3, we now use all components of G_γ for parity correction. We will gain a constant amount for every clean ear entering a bad ear of length at least six. Taking the better of the two T-tours constructed in the previous section and in this section, we will improve upon $\frac{3}{2}(n-1)$ by $\frac{1}{26}$ per non-entered ear, unless a large fraction of the long ears are 4-ears (remember that all bad ears are even and hence there are no bad 5-ears).

We now describe a new kind of ear induction. For 4-ears, good and special ears we essentially use Lemma 8.1. For bad long ears, however, we will now gain something by using the only entering clean ear in a different way: if this clean ear enters at w, we can "flip" all clean ears on the path from $r(w)$ to w, changing the parity at w and at $r(w)$, similar to the treatment of special ears above. This flexibility allows to save something for each bad ear of length at least six.

We again consider the long ears in reverse order, starting from P_l. Let $T'_l := T_l = T \triangle \text{odd}(E_\gamma)$ and $T^\gamma_l := \emptyset$. The sets T^γ_i record the vertices whose parity we will change by flipping clean ears (see the green framed vertices in Figure 8.2), and T'_i contains the vertices requiring odd degree in the first i (long) ears after these flips.

If $(P_i, T_i \cap \text{in}(P_i))$ is a bad pair and P_i is not a 4-ear, P_i has exactly one entering clean ear Q that enters P_i at the middle vertex w (and $r(w) \in V_{i-1}$); then let $W_i := \{w\}$. Otherwise let $W_i := \emptyset$. We construct a multi-set $F'_i \subseteq 2E(P_i)$ such that all internal vertices except possibly w have the correct parity, i.e.,

$$\text{odd}(F'_i) \cap (\text{in}(P_i) \setminus W_i) \ = \ T'_i \cap (\text{in}(P_i) \setminus W_i) \tag{8.3}$$

and $(V_i, F'_i)/V_{i-1}$ is connected. (We will later describe in detail how we construct F'_i; see Lemma 8.20.) If $W_i = \{w\}$ and $w \in (\text{in}(P_i) \cap \text{odd}(F'_i)) \triangle T'_i$, we define

$$T'_{i-1} \ := \ T'_i \triangle \text{odd}(F'_i) \triangle \{w, r(w)\}$$

and $T^\gamma_{i-1} := T^\gamma_i \triangle \{w, r(w)\}$; this is because we will later correct the parity at w by flipping the clean ears on the $r(w)$-w-path in G_γ. Otherwise, let $T^\gamma_{i-1} := T^\gamma_i$ and

$$T'_{i-1} \ := \ T'_i \triangle \text{odd}(F'_i).$$

We have $T'_{i-1} \subseteq V_{i-1}$ by (8.3) and since $r(w) \in V_{i-1}$ in the first case. Note that $|T_{i-1}|$ is even because both $|T'_i|$ and $|\mathrm{odd}(F'_i)|$ are even.

Lemma 8.13. *$(V_l, F'_1 \dot\cup \ldots \dot\cup F'_l)$ is connected.*

Proof. Since $(V_i, F'_i)/V_{i-1}$ is connected for every $i \in \{1, \ldots, l\}$, the multi-set F'_i contains the edge set of a path from every internal vertex of P_i to a vertex in V_{i-1}. By induction this implies that for every $i \in \{1, \ldots, l\}$ the multi-set $F'_1 \dot\cup \ldots \dot\cup F'_i$ contains the edge set of a path from every internal vertex of P_i to the unique element of V_0. This proves that $(V_l, F'_1 \dot\cup \ldots \dot\cup F'_l)$ is connected. □

Lemma 8.14. *For every $i \in \{1, \ldots, l+1\}$ the multi-set $F'_i \dot\cup \ldots \dot\cup F'_l$ is a $(T_l \triangle T^\gamma_{i-1} \triangle T'_{i-1})$-join.*

Proof. We prove this by induction on $l - i$. For $i = l + 1$ we have $T_l = T'_l$ and $T^\gamma_l = \emptyset$. Hence, $T_l \triangle T^\gamma_{i-1} \triangle T'_{i-1} = \emptyset$. Now let $i \leq l$. By induction hypothesis $F'_{i+1} \dot\cup \ldots \dot\cup F'_l$ is a $(T_l \triangle T^\gamma_i \triangle T'_i)$-join. Thus, proving that $F'_i \dot\cup \ldots \dot\cup F'_l$ is a $(T_l \triangle T^\gamma_{i-1} \triangle T'_{i-1})$-join is equivalent to proving that F'_i is a $((T_l \triangle T^\gamma_i \triangle T'_i) \triangle (T_l \triangle T^\gamma_{i-1} \triangle T'_{i-1}))$-join. We have either

- $T'_{i-1} := T'_i \triangle \mathrm{odd}(F'_i) \triangle \{w, r(w)\}$ and $T^\gamma_{i-1} := T^\gamma_i \triangle \{w, r(w)\}$, or
- $T'_{i-1} := T'_i \triangle \mathrm{odd}(F'_i)$ and $T^\gamma_{i-1} := T^\gamma_i$.

In any of the two cases we have

$$\left(T_l \triangle T^\gamma_i \triangle T'_i\right) \triangle \left(T_l \triangle T^\gamma_{i-1} \triangle T'_{i-1}\right) \;=\; \left(T^\gamma_i \triangle T^\gamma_{i-1}\right) \triangle \left(T'_i \triangle T'_{i-1}\right) \;=\; \mathrm{odd}(F'_i).$$

□

Lemma 8.15. *For every $i \in \{0, \ldots, l\}$ the edge set E_γ contains a T^γ_i-join.*

Proof. We again use induction on $l - i$. For $i = l$, we have $T^\gamma_i = \emptyset$ and the statement clearly holds. Let now $i < l$. We either have $T^\gamma_i = T^\gamma_{i+1}$ or $T^\gamma_i = T^\gamma_{i+1} \triangle \{w, r(w)\}$ for some vertex w. If $T^\gamma_i = T^\gamma_{i+1}$, the set E_γ contains a T^γ_i-join by induction hypothesis. If $T^\gamma_i = T^\gamma_{i+1} \triangle \{w, r(w)\}$, the set E_γ contains the edge set of a w-$r(w)$-path since $r(w)$ is the root of the connected component of $G_\gamma = (V, E_\gamma)$. Moreover, by induction hypothesis, E_γ contains a T^γ_{i+1}-join. The symmetric difference of such a T^γ_{i+1}-join and a w-$r(w)$-path in G_γ is a T^γ_i-join. □

For $i \in \{1, \ldots, l\}$ and a multi-set $F \subseteq 2E(P_i)$ let

$$\mathrm{gain}'_i(F) \;:=\; \tfrac{3}{2}|\mathrm{in}(P_i)| - |F|.$$

Lemma 8.16. *We can construct a T-tour in G with at most*

$$\tfrac{3}{2}(n-1) + \tfrac{1}{2}k_2 + k_3 - \sum_{i=1}^{l} \mathrm{gain}'_i(F'_i)$$

edges.

Proof. By Lemma 8.15 the edge set E_γ contains a T_0^γ-join J. Note that we never add any internal vertex of a short ear to a set T_i^γ for any i. Hence, T_0^γ contains no internal vertex of any short ear. This shows that for every short ear Q either all edges of Q are contained in J or none of these edges. We now define an edge set $H \subseteq 2E_\gamma$. If the edge set of a short ear Q is contained in J, we add two copies of all elements of $E(Q)$ except one to H. Otherwise, we have $E(Q) \cap J = \emptyset$ and we add $E(Q)$ to H. See Figure 8.2.

Since H and $E_\gamma \triangle J$ are identical up to pairs of parallel edges, $|\delta_H(v)|$ is odd if and only if $|\delta_{E_\gamma \triangle J}(v)|$ is odd for every vertex v, and this holds if and only if $|\delta_{E_\gamma}(v)| + |\{v\} \cap T_0^\gamma|$ is odd. Recall that $T_l = T \triangle \operatorname{odd}(E_\gamma) \subseteq V_l$ and thus $T_0^\gamma \triangle \operatorname{odd}(E_\gamma) = T_l \triangle T \triangle T_0^\gamma$. Hence, H is a $(T \triangle T_l \triangle T_0^\gamma)$-join and we have $|H| \leq 2k_2 + 4k_3$. Moreover, for every short ear Q and every vertex $v \in \operatorname{in}(Q)$, the edge set of some path from v to a vertex in V_l is contained in H. Together with Lemma 8.13 this shows that $(V, H \dot\cup F_1' \dot\cup \ldots \dot\cup F_l')$ is connected.

Since $|T_i'|$ is even for every $i \in \{0, \ldots, l\}$, $T_0' \subseteq V_0$,and $|V_0| = 1$, we have $T_0' = \emptyset$. Hence, by Lemma 8.14, $F_1' \dot\cup \ldots \dot\cup F_l'$ is a $(T_l \triangle T_0^\gamma)$-join. As H is a $(T \triangle T_l \triangle T_0^\gamma)$-join, $H \dot\cup F_1' \dot\cup \ldots \dot\cup F_l'$ is a T-join and thus a T-tour with

$$|H| + \sum_{i=1}^{l} |F_i'| \;\leq\; 2k_2 + 4k_3 + \sum_{i=1}^{l} |F_i'|$$

edges. By definition of gain_i' we have $|F_i'| = \frac{3}{2}|\operatorname{in}(P_i)| - \operatorname{gain}_i'(F_i')$. Moreover, the number of internal vertices of short ears is $k_2 + 2k_3$. □

We aim at $\operatorname{gain}_i'(F_i') \geq 1$ for the ears P_i that were bad in the previous section. This will be easy to achieve if $(P_i, T_i' \cap \operatorname{in}(P_i))$ is a bad pair and P_i has at least six edges. However, T_i (from the ear induction in Section 8.2.3) and T_i' are not always the same. Hence $(P_i, T_i' \cap \operatorname{in}(P_i))$ is not necessarily a bad pair even if $(P_i, T_i \cap \operatorname{in}(P_i))$ is. Therefore we also aim at minimizing $|T_i \triangle T_i'|$ whenever we have the choice between two solutions with the same number of edges. This will allow us to achieve a sufficiently large $\operatorname{gain}_i'(F_i')$ on average.

Lemma 8.17. *Let $i \in \{1, \ldots, l+1\}$. Then for every $v \in T_{i-1} \triangle T_{i-1}'$ we have one of the following three properties:*

(a) The vertex v is contained in $(T_i \triangle T_i') \cap V_{i-1}$.

(b) v is an endpoint of P_i and $|\delta_{F_i}(v)| + |\delta_{F_i'}(v)|$ is odd.

(c) We have $h_i = 1$ and the unique clean ear entering P_i enters P_i at a vertex w with $v = r(w)$. Moreover, $|\delta_{F_i}(w)| + |\delta_{F_i'}(w)|$ is odd.

Proof. We have by construction of the sets T_{i-1} and T_{i-1}' that both of these sets are subsets of V_{i-1}. Now let $v \in T_{i-1} \triangle T_{i-1}'$ such that v fulfills neither (a) nor (c). Then we have either $v \in T_i \triangle T_{i-1}$ or $v \in T_i' \triangle T_{i-1}'$. If $v \in T_i \triangle T_{i-1}$ but $v \notin T_i' \triangle T_{i-1}'$, we have $|\delta_{F_i}(v)|$ odd and $|\delta_{F_i'}(v)|$ even. Similarly, if $v \in T_i' \triangle T_{i-1}'$ but $v \notin T_i \triangle T_{i-1}$, we have $|\delta_{F_i'}(v)|$ odd and $|\delta_{F_i}(v)|$ even. In any of the two cases (b) holds. □

For $i \in \{1, \ldots, l\}$ let

$$\Delta_i \;:=\; |T_{i-1} \triangle T_{i-1}'| - |T_i \triangle T_i'|.$$

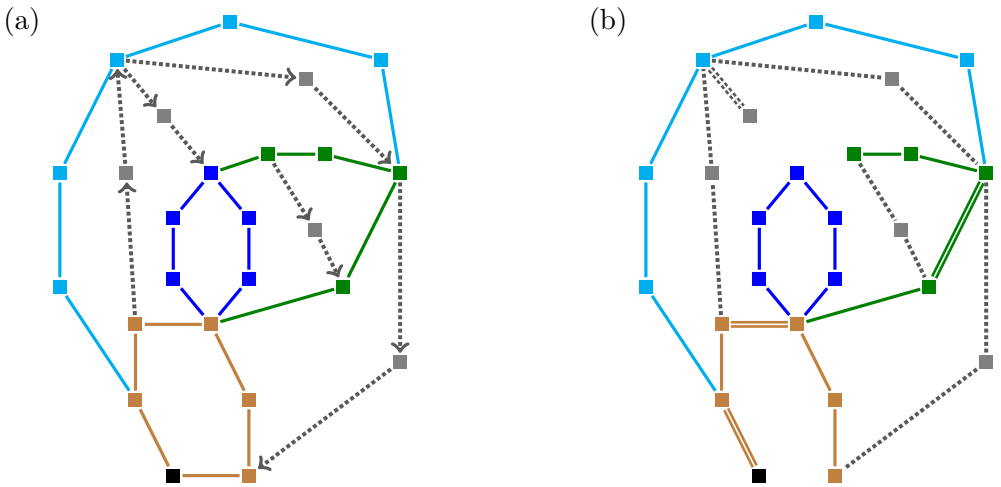

Figure 8.6: The example from Figure 8.4 with $T = \emptyset$. (a): ears with oriented short ears (gray, dotted). (b): result of using short ears for parity.

Lemma 8.18. *For every $i \in \{1, \dots, l\}$, we have $\Delta_i \leq 2$ and Δ_i is even.*

Proof. By Lemma 8.17, $|T_{i-1} \triangle T'_{i-1}| \leq |T_i \triangle T'_i| + 3$. As the symmetric difference of two even-cardinality sets has always even cardinality, we have Δ_i even and $|T_{i-1} \triangle T'_{i-1}| \leq |T_i \triangle T'_i| + 2$. □

We now show how to construct the F'_i. For good or special ears we can simply apply Lemma 8.1, but we will need a slightly refined version: in case the bound is tight, we want the parity at the endpoints to match the previous construction.

Lemma 8.19. *Let P be a circuit with at least four edges and $T_P \subseteq V(P)$ with $|T_P|$ even. Then there exists a T_P-join $F \subseteq 2E(P)$ such that the graph $(V(P), F)$ is connected and*

$$|F| \;\leq\; \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}. \tag{8.4}$$

Moreover, if this bound is tight and $|E(P)| + |T_P| \geq 5$, then there is another T_P-join $F' \subseteq 2E(P)$ such that the graph $(V(P), F')$ is connected and $|F'| = |F|$ and the number of copies of any edge in $F \dot\cup F'$ is odd.

Proof. If $T_P = \emptyset$, then $F = E(P)$ does the job because $|E(P)| \leq \frac{3}{2}|E(P)| - 2$, and the bound is tight only if $|E(P)| = 4$.

Let now $T_P \neq \emptyset$. The vertices of T_P subdivide P into subpaths. Color these paths alternatingly red and blue. Let E_R and E_B denote the set of edges of red and blue subpaths, respectively. We define two T_P-joins, F_R and F_B. For the red solution F_R, take two copies of each edge in E_R and one copy of each edge in E_B. Note that $E_R \neq \emptyset$, and remove one pair of parallel edges. F_B is formed by switching the roles of red and blue. This yields $F_R, F_B \subseteq 2E(P)$ with

$$\tfrac{1}{2}\left(|F_R| + |F_B|\right) \;=\; \tfrac{1}{2}\left(3|E_B| + 3|E_R| - 4\right) \;\leq\; \tfrac{1}{2}\left(3|E(P)| - 4\right) \;=\; \tfrac{3}{2}(|E(P)| - 1) - \tfrac{1}{2}.$$

Note that the number of copies of any edge in $F_R \dot\cup F_B$ is odd. Moreover, $|F_R| = |F_B|$ or the smaller one has fewer than $\frac{3}{2}(|E(P)| - 1) - \frac{1}{2}$ edges. □

Recall that $W_i = \emptyset$ unless $(P_i, T_i \cap \text{in}(P_i))$ is a bad pair and P_i is not a 4-ear. If $W_i \neq \emptyset$, the ear P_i has exactly one entering clean ear, entering P_i at w and $r(w) \in V_{i-1}$; then $W_i = \{w\}$.

Lemma 8.20. *Let $i \in \{1, \dots, l\}$ and $T_i \subseteq V_i$ with $|T_i|$ even. Then we can construct a multi-set $F_i' \subseteq 2E(P_i)$ such that*

- $\text{odd}(F_i') \cap (\text{in}(P_i) \setminus W_i) = T_i' \cap (\text{in}(P_i) \setminus W_i)$,
- *$(V_i, F_i')/V_{i-1}$ is connected, and*

$$\text{gain}_i'(F_i') - \tfrac{1}{4}\Delta_i \;\geq\; \begin{cases} 1 & \text{if } |E(P_i)| \geq 5 \text{ and } (P_i, T_i \cap \text{in}(P_i)) \text{ is bad} \quad \text{(a)} \\ \frac{1}{2} & \text{if either } |E(P_i)| \geq 5 \text{ or } (P_i, T_i \cap \text{in}(P_i)) \text{ is bad} \quad \text{(b)} \\ 0 & \text{if } |E(P_i)| = 4 \text{ and } (P_i, T_i \cap \text{in}(P_i)) \text{ is not bad} \quad \text{(c)} \end{cases}$$

Proof. (a) The vertices of $\text{in}(P_i) \cap T_i'$ subdivide P_i into subpaths, alternatingly colored red and blue. Let E_R and E_B denote the set of edges of red and blue subpaths, respectively. Let Q be the unique clean ear entering P_i at $w \in \text{in}(P_i)$. Let E_1, E_2 be the edge sets of the two paths in P_i from w to an endpoint of P_i. Then $E(P_i) = E_1 \dot\cup E_2$. By definition of a bad pair, $|E_1| = |E_2|$ and $r(w) \notin \text{in}(P_i)$. Hence, by the choice of the orientation of G_γ, we have $r(w) \in V_{i-1}$. Note that $W_i = \{w\}$. We now distinguish two cases:

Case 1: $|E_R \cap E_1| \neq |E_B \cap E_1|$ or $|E_R \cap E_2| \neq |E_B \cap E_2|$

Without loss of generality we may assume $|E_R \cap E_1| < |E_B \cap E_1|$. Then, we construct F_i' from $E(P)$ by adding a second copy of $E_R \cap E_1$, adding a second copy of the edges in the smaller of the two sets $|E_R \cap E_2|$ and $|E_B \cap E_2|$, and removing one arbitrary duplicated edge, if it exists. Then $|F_i'| \leq |E(P)| \leq \frac{3}{2}|\text{in}(P_i)| - \frac{3}{2}$ if there was no duplicated edge; otherwise $|F_i'| \leq |E(P_i)| + \lfloor \frac{1}{2}(|E_1| - 1) \rfloor + \lfloor \frac{1}{2}|E_2| \rfloor - 2 \leq \frac{3}{2}|E(P_i)| - 3 \leq \frac{3}{2}|\text{in}(P_i)| - \frac{3}{2}$. Thus, in both cases, $\text{gain}_i'(F_i') \geq \frac{3}{2}$ and $\Delta_i \leq 2$.

Case 2: $|E_R \cap E_1| = |E_B \cap E_1|$ and $|E_R \cap E_2| = |E_B \cap E_2|$

By the definition of a bad pair, we have $\text{in}(P_i) \cap T_i = \{w\}$, and w is the middle internal vertex of P_i. If in addition $|\text{in}(P_i) \cap (T_i' \triangle T_i)| \leq 1$, $E_R \cap E_1$, $E_B \cap E_1$, $E_R \cap E_2$, or $E_B \cap E_2$ must be empty, contradicting $|E_R \cap E_1| = |E_B \cap E_1|$ and $|E_R \cap E_2| = |E_B \cap E_2|$. Hence, we may assume that $|\text{in}(P_i) \cap (T_i' \triangle T_i)|$ is at least two.

Similar to Case 1, we construct F_i' from $E(P_i)$ by adding a second copy of either $E_R \cap E_1$ or $E_B \cap E_1$, adding a second copy of the edges of one the two sets $E_R \cap E_2$ and $E_B \cap E_2$, and removing both copies of an arbitrary duplicated edge. We choose between $E_R \cap E_1$ and $E_B \cap E_1$ and between $E_R \cap E_2$ and $E_B \cap E_2$ such that the endpoints of P_i have the same parity of degree in F_i' and in F_i. Then $|F_i'| = \frac{3}{2}|E(P)| - 2 = \frac{3}{2}|\text{in}(P_i)| - \frac{1}{2}$, so $\text{gain}_i'(F_i') = \frac{1}{2}$.

Moreover, we get from Lemma 8.17 that $|T_{i-1} \triangle T_{i-1}'| \leq |T_i \triangle T_i'| + 1 - |(T_i \triangle T_i') \cap \text{in}(P_i)|$. Since $|\text{in}(P_i) \cap (T_{i+1}' \triangle T_{i+1})|$ is at least two, we have $|T_{i-1} \triangle T_{i-1}'| \leq |T_i \triangle T_i'| - 1$. As by Lemma 8.18 the number Δ_i must be even, $\Delta_i \leq -2$.

(b) First suppose that $|E(P_i)| \geq 5$ or $T_i' \neq \emptyset$. Then we apply Lemma 8.19 to $(G_i, T_i')/V_{i-1}$. We get a set F_i' with $\text{gain}_i'(F_i') \geq \frac{1}{2}$. If P_i is a circuit, we get from Lemma 8.17 that $\Delta_i \leq 1$. If P_i is a path, we get from Lemma 8.19 that one of the following holds:

- We have $\text{gain}_i'(F_i') \geq 1$.
- For one endpoint of P_i we can choose the parity of its degree in F_i', implying $\Delta_i \leq 1$ by Lemma 8.17.

Using Lemma 8.18 this implies that ($\text{gain}_i'(F_i') \geq 1$ and $\Delta_i \leq 2$) or ($\text{gain}_i'(F_i') \geq \frac{1}{2}$ and $\Delta_i \leq 0$). In both cases we get $\text{gain}_i'(F_i') - \frac{1}{4}\Delta_i \geq \frac{1}{2}$.

Now suppose that $|E(P_i)| = 4$ and $T_i' = \emptyset$ and $(P_i, T_i \cap \text{in}(P_i))$ is bad; so $T_i = \emptyset$. Then we set $F_i' := F_i$. Then we have $T_{i-1} \triangle T_{i-1}' = T_i \triangle T_i'$ and thus $\Delta_i = 0$. By Lemma 8.8, we have $|F_i'| = |F_i| \leq \frac{3}{2}|\text{in}(P_i)| - \frac{1}{2}$. Hence, $\text{gain}_i'(F_i') \geq \frac{1}{2}$.

(c) We apply Lemma 8.1 to $(G_i, T_i')/V_{i-1}$ to obtain a multi-set F_i' with $\text{gain}_i'(F_i') \geq \frac{1}{2}$. By Lemma 8.18, we always have $\Delta_i \leq 2$. Hence, $\text{gain}_i'(F_i') - \frac{1}{4}\Delta_i \geq 0$.

□

Lemma 8.21. *Given a well-oriented ear-decomposition with long ears $P_1, \ldots, P_l$ where all short ears are clean and the oriented ears are precisely the clean ears, we can compute a T-tour with at most*

$$\tfrac{3}{2}(n-1) + \tfrac{1}{2}(k_2 + k_3 - k_{\geq 5}) + \tfrac{1}{2}k_3 - \tfrac{1}{2}k_{\text{bad}}$$

edges, where k_{bad} is the number of bad pairs $(P_i, \text{in}(P_i) \cap T_i)$.

Proof. Since $T_l = T_l'$ and $T_0 = T_0' = \emptyset$, we have

$$\sum_{i=1}^{l} \Delta_i \;=\; \sum_{i=1}^{l} \Big(|T_{i-1} \triangle T_{i-1}'| - |T_i \triangle T_i'| \Big) \;=\; |T_0 \triangle T_0'| - |T_l \triangle T_l'| \;=\; 0. \tag{8.5}$$

By construction of the sets F_i' we have

$$\sum_{i=1}^{l} \Big(\text{gain}_i'(F_i') - \tfrac{1}{4}\Delta_i \Big) \;\geq\; \tfrac{1}{2}k_{\geq 5} + \tfrac{1}{2}k_{\text{bad}}.$$

Using (8.5), this implies

$$\sum_{i=1}^{l} \text{gain}_i'(F_i') \;\geq\; \tfrac{1}{2}k_{\geq 5} + \tfrac{1}{2}k_{\text{bad}}.$$

By Lemma 8.16 we can construct a T-tour in G with at most

$$\begin{aligned} &\tfrac{3}{2}(n-1) + \tfrac{1}{2}k_2 + k_3 - \sum_{i=1}^{l} \text{gain}_i'(F_i') \\ \leq\; &\tfrac{3}{2}(n-1) + \tfrac{1}{2}k_2 + k_3 - \tfrac{1}{2}k_{\geq 5} - \tfrac{1}{2}k_{\text{bad}} \\ =\; &\tfrac{3}{2}(n-1) + \tfrac{1}{2}(k_2 + k_3 - k_{\geq 5}) + \tfrac{1}{2}k_3 - \tfrac{1}{2}k_{\text{bad}}. \end{aligned}$$

edges. □

We now combine Lemma 8.12 and Lemma 8.21 to prove a bound on the number of edges of the better of the two T-tours resulting from the two different kinds of ear induction.

Theorem 8.22. *Let G be a graph and $T \subseteq V(G)$ with $|T|$ even. Given a well-oriented ear-decomposition of G where all short ears are clean and the oriented ears are precisely the clean ears, we can compute a T-tour in G with at most*

$$\tfrac{3}{2}(n-1) - \tfrac{1}{26}\pi + \tfrac{1}{26}(k_4 - 2k_{\geq 5})$$

edges, where π is the number of non-entered ears.

Proof. We apply Lemma 8.12 and Lemma 8.21 and take the shorter of the two T-tours. This yields a T-tour with at most the following number of edges:

$$\min\Big\{ \tfrac{3}{2}(n-1) - \tfrac{7}{20}k_3 - \sum_{i\in I} \max\left\{ \tfrac{7}{20}(h_i - 1),\ \tfrac{3}{20} \right\},$$
$$\tfrac{3}{2}(n-1) + \tfrac{1}{2}(k_2 + k_3 - k_{\geq 5}) + \tfrac{1}{2}k_3 - \tfrac{1}{2}k_{\text{bad}} \Big\},$$

where $I = \{i \in \{1, \ldots, l\} : (P_i, T_i \cap \text{in}(P_i)) \text{ is good or special}\}$. Taking $\frac{10}{13}$ of the first term and $\frac{3}{13}$ of the second term, we can bound it by

$$\begin{aligned} &\tfrac{3}{2}(n-1) - \tfrac{7}{26}k_3 - \sum_{i\in I} \max\left\{ \tfrac{7}{26}(h_i - 1),\ \tfrac{3}{26} \right\} + \tfrac{3}{26}(k_2 + k_3 - k_{\geq 5}) + \tfrac{3}{26}k_3 - \tfrac{3}{26}k_{\text{bad}} \\ = &\tfrac{3}{2}(n-1) - \tfrac{4}{26}k_3 - \tfrac{1}{26}\sum_{i=1}^{l} \max\{7(h_i - 1), 3\} + \tfrac{3}{26}(k_2 + k_3) - \tfrac{3}{26}k_{\geq 5}, \end{aligned}$$

where we used $i \notin I$ if and only if $(P_i, T_i \cap \text{in}(P_i))$ is bad, and this implies $h_i = 1$. We get the following upper bound on the number of edges of our tour:

$$\begin{aligned} &\tfrac{3}{2}(n-1) - \tfrac{1}{26}\sum_{i=1}^{l} \max\{7(h_i - 1), 3\} + \tfrac{3}{26}(k_2 + k_3) - \tfrac{3}{26}k_{\geq 5} \\ \leq &\tfrac{3}{2}(n-1) - \tfrac{1}{26}\sum_{i=1}^{l} \max\{4h_i - 1, 0\} + \tfrac{3}{26}(k_2 + k_3) - \tfrac{3}{26}k_{\geq 5} \\ = &\tfrac{3}{2}(n-1) - \tfrac{1}{26}\sum_{i=1}^{l} \max\{4h_i, 1\} + \tfrac{3}{26}(k_2 + k_3) + \tfrac{1}{26}(k_4 - 2k_{\geq 5}) \\ = &\tfrac{3}{2}(n-1) - \tfrac{1}{26}\sum_{i=1}^{l} \max\{h_i, 1\} + \tfrac{1}{26}(k_4 - 2k_{\geq 5}) \\ = &\tfrac{3}{2}(n-1) - \tfrac{1}{26}\pi + \tfrac{1}{26}(k_4 - 2k_{\geq 5}). \end{aligned}$$

In the last inequality we used that every non-entered ear is a short ear or a long ear with $h_i = 0$. □

8.3 Computing the initial ear-decomposition

Previous papers that used ear-decompositions for approximation algorithms include [CSS01], [SV14], and [HV17]. They all exploit a theorem of Frank [Fra93]: one can compute an ear-decomposition with minimum number of even ears in polynomial time. This minimum is denoted by $\varphi(G)$. Our ear-decompositions will also have only $\varphi(G)$ even ears, although (in contrast to the above-mentioned papers) we exploit this property only during the construction of the ear-decomposition. The ear-decompositions in the above papers also have certain properties of 2-ears and 3-ears; we will additionally deal with 4-ears. As in [SV14] we will compute a nice ear-decomposition. In particular, we make all short ears pendant.

We have seen in Theorem 8.22 that non-entered ears are cheap but 4-ears are expensive. For pendant 4-ears we can apply Lemma 8.1 beforehand and apply Theorem 8.22 to the rest. Ideally, we would like to make all 4-ears pendant, but this is not always possible.

We distinguish four kinds of 4-ears: pendant, blocked, vertical, and horizontal (we will compute an ear-decomposition in which every ear is of exactly one of these kinds); see Figure 8.7:

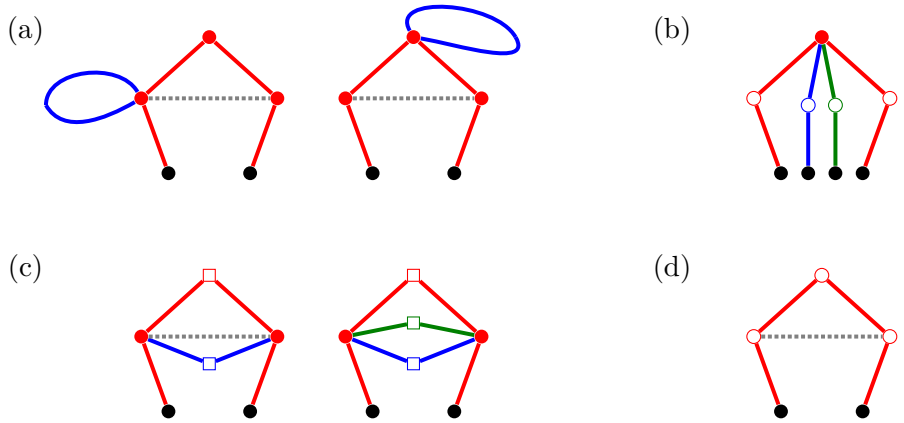

Figure 8.7: (a) blocked 4-ears, (b) vertical 4-ear, (c) horizontal 4-ears, (d) pendant 4-ear. Filled circles denote arbitrary vertices, unfilled circles denote pendant vertices, unfilled squares denote degree-2 vertices. The endpoints of the 4-ears and endpoints of 2-ears that are not internal vertices of these 4-ears are shown at the bottom (black filled circles); some of these can be identical. Curves denote closed ears. Dotted edges are possible trivial ears connecting colored vertices.

Definition 8.23. *A 4-ear is called*

- blocked *if a closed ear is attached to it.*
- vertical *if it is nonpendant, its internal vertices are v_1, v_2, v_3 in this order, v_1 and v_3 are pendant and not adjacent, and the only nontrivial ears attached to v_2 are 2-ears whose middle vertex is not adjacent to v_1 or v_3.*
- horizontal *if it is nonpendant, its internal vertices are v_1, v_2, v_3 in this order, v_2 has degree 2, and all nontrivial ears attached to it are 2-ears with vertices v_1, x, v_3, where x is a degree-2 vertex.*

Call an ear outer *if it is a 2-ear, 3-ear, pendant 4-ear, vertical 4-ear, or horizontal 4-ear. Call an ear* inner *if it is a blocked 4-ear or has length at least 5.*

Later we will show that there are only few blocked 4-ears (Lemma 8.33). Moreover, in Section 8.4 we make as many vertical 4-ears pendant as possible and raise the lower bound for every 2-ear that is still attached to a vertical or horizontal 4-ear.

In this section, we prove the following:

Theorem 8.24. *For any 2-vertex-connected graph G we can in polynomial time construct an ear-decomposition that satisfies the following conditions:*

(a) *All short ears are open and pendant.*

(b) *Each 4-ear is blocked or pendant or vertical or horizontal; no closed 4-ear is attached to any closed 4-ear.*

(c) *If there is an edge $\{v, w\}$ such that v is an internal vertex of an outer ear P and w is an internal vertex of another outer ear Q, then*

- *P is attached to Q at w, or*
- *Q is attached to P at v, or*
- *P and Q are 4-ears, and v and w are their middle vertices.*

No 2-ear is attached to two outer 4-ears.

For the proof, we will start with the following, which is a strengthening of Frank's theorem [Fra93] because it also yields that all ears are open.

Lemma 8.25 (Cheriyan, Sebő, and Szigeti [CSS01])**.** *For any given 2-vertex-connected graph G, one can compute an open ear-decomposition with $\varphi(G)$ even ears in polynomial time.*

We will maintain the following invariants at all times.

(d) All short ears are open.

(e) No closed ears are attached to short ears.

(f) No closed 4-ears or nonpendant 3-ears are attached to any closed 4-ear.

(g) The number of even ears is $\varphi(G)$.

Note that the result of Lemma 8.25 satisfies these invariants because all its ears are open, including the first ear (which is always open by definition). Now we apply a certain set of operations as long as possible. Each of them maintains these invariants and decreases the following potential function: we lexicographically

1. maximize the number of trivial ears,
2. minimize the number of 4-ears, and
3. minimize the number of trivial ears that are incident to middle vertices of outer 4-ears.

Thus after fewer than n^4 steps none of the operations can be applied (where we use that the number of trivial ears is always at least $|E(G)| - 2n$ and at most $|E(G)| - n$). Then the properties (a), (b), and (c) will hold. At any stage we put pendant ears at the end of the ear-decomposition in an order of nonincreasing length (in particular pendant 3-ears before pendant 2-ears), followed only by trivial ears.

Lemma 8.26. *Given an ear-decomposition with* (d)–(g) *but not* (a)*, we can compute an ear-decomposition with more trivial ears and* (d)–(g) *in polynomial time.*

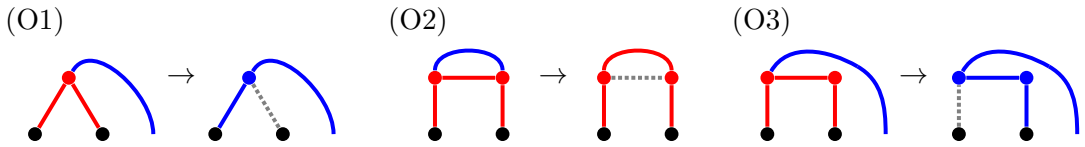

Figure 8.8: Removing nonpendant short ears. (O1): removing a nonpendant 2-ear; (O2),(O3): removing a nonpendant 3-ear. The blue curve in the left picture of (O1), (O2), (O3) denotes a non-trivial ear attached to a red short ear. New trivial ears are dotted.

Proof. The following set of operations removes a nonpendant 2-ear or 3-ear and is illustrated in Figure 8.8. Each operation increases the number of trivial ears and does not increase the number of even ears.

(O1) If there is a nonpendant 2-ear P, let Q be the first nontrivial ear attached to P. Note that Q is open by (e). We extend Q by one of the edges of P so that the resulting ear is open; P vanishes; the other edge of P becomes a trivial ear.

Note that Q must be odd because otherwise (O1) would reduce the number of even ears, contradicting (g). Since Q is not a 2-ear, the new ear is not short, and (f) is maintained.

If all 2-ears but not all 3-ears are pendant, let P be the first nonpendant 3-ear, and let v_0, v_1, v_2, v_3 be the vertices of P in this order. Note that P is open by (d). Let Q be the first nontrivial ear attached to P, and without loss of generality v_1 is an endpoint of Q. Note that Q is open by (e). There are two cases.

(O2) If Q has endpoints v_1 and v_2, we replace the middle edge of P by the edges of Q, creating a new open ear with at least four edges and a trivial ear.

(O3) If v_2 is not an endpoint of Q, we extend Q by the v_1-v_3-path in P; the remaining edge of P becomes a trivial ear.

The operation (O3) might create a closed ear of length at least 4, but then it is not attached to a short ear because P was the first nonpendant short ear, and it is not attached to a closed 4-ear because P was not attached to a closed 4-ear by (f). Moreover, if the new ear is a closed 4-ear, then Q was a pendant 2-ear, and (by the choice of Q and the order of the ear-decomposition) only pendant 2-ears are attached to the new closed 4-ear. Therefore (f) is maintained. □

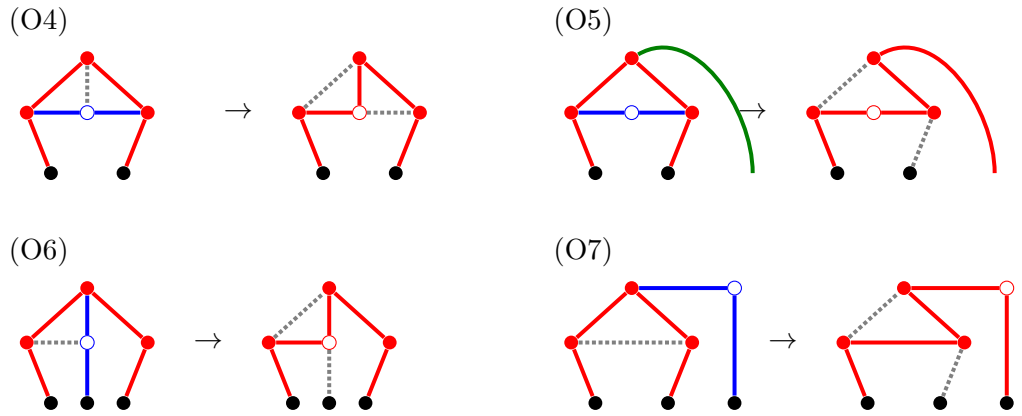

Figure 8.9: (O4)–(O7): excluding 4-ears that are neither pendant nor vertical nor horizontal.

Before we get to condition (b), let us show a sufficient condition for 4-ears to be pendant or vertical or horizontal:

Lemma 8.27. *Let P be a 4-ear in an ear-decomposition satisfying (a) and (g). Let v_0, v_1, v_2, v_3, v_4 be the vertices of P in this order (where v_0 and v_4 can be identical). Then P is pendant or vertical or horizontal if and only if every nontrivial ear attached to P is a 2-ear that is attached to P at v_1 and v_3 or only at v_2.*

Proof. Necessity follows directly from Definition 8.23, so we prove sufficiency. So let P be a 4-ear in an ear-decomposition satisfying (a) and (g) such that every nontrivial ear attached to P is a (pendant) 2-ear that is attached to P at v_1 and v_3 or only at v_2.

First suppose that a 2-ear Q with endpoints v_1 and v_3 exists. Let w be the middle vertex of Q. If P is not horizontal, at least one of the following two operations must apply:

(O4) If there is an edge connecting w and v_2, we replace P by a 5-ear, formed by the edges $\{v_0, v_1\}$, $\{v_2, v_3\}$ and $\{v_3, v_4\}$ of P, the edge $\{v_1, w\}$ of Q, and the edge $\{w, v_2\}$.

(O5) If any (possibly trivial) ear R has endpoints x and y, where $x \in \{v_2, w\}$ and $y \notin \{w, v_1, v_2, v_3\}$, we replace P by an ear of length at least 5, formed by R and all but two of the edges of P and Q; the remaining two edges become trivial ears.

Each of (O4) and (O5) decreases the number of even ears, and therefore this cannot happen. It remains to consider the case when all ears attached to P are 2-ears attached to P only at v_2. If P is not pendant, there is at least one such a 2-ear attached to P.

If P also not vertical, this means that v_1 and v_3 are adjacent or the middle vertex w of a 2-ear attached to v_2 is adjacent to v_1 or v_3. Then we can apply one of the following operations:

(O6) If w is adjacent to v_1, we can replace P by a 5-ear with edges $\{v_0, v_1\}$, $\{v_1, w\}$, $\{w, v_2\}$, $\{v_2, v_3\}$ and $\{v_3, v_4\}$. (By renumbering the vertices of P, the same operation applies if w is adjacent to v_3.)

(O7) If v_1 and v_3 are adjacent, we can replace P by a 5-ear formed by a 2-ear attached to P at v_2 and the edges $\{v_0, v_1\}$, $\{v_1, v_3\}$, $\{v_3, v_2\}$.

Both (O6) and (O7) replace P and an attached 2-ear by a 5-ear, again contradicting (g). □

Lemma 8.28. *Given an ear-decomposition with* (d)–(g) *but not* (b)*, we can compute an ear-decomposition with* (d)–(g) *in polynomial time that either has more trivial ears or has the same number of trivial ears but fewer 4-ears.*

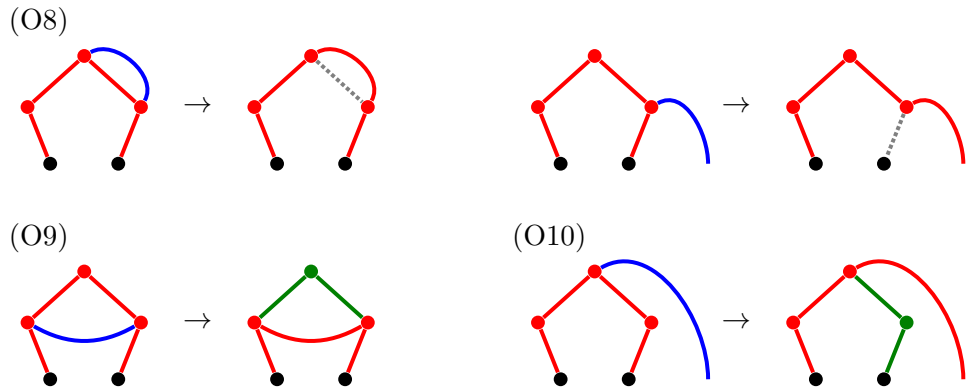

Figure 8.10: Making sure that every 4-ear is blocked or pendant or vertical or horizontal. (O8): removing nontrivial ears attached to a 4-ear either at v_1 or at v_3; (O9): removing ears of length at least 3 attached to a 4-ear at v_1 and v_3; (O10): removing ears of length at least 3 attached to a 4-ear only at v_2;

Proof. We may assume that (a) holds; otherwise we are done by Lemma 8.26. We have (f) but not (b), so let P be a 4-ear that is neither blocked nor pendant nor vertical nor horizontal. Let v_0, v_1, v_2, v_3, v_4 be the vertices of P in this order (where v_0 and v_4 can be identical), and let Q be the first ear attached to P. As P is neither pendant nor blocked, Q is nontrivial and open.

(O8) If either v_1 or v_3 is an endpoint of Q, we can replace one edge of P by Q; this edge becomes a trivial ear. The new nontrivial ear has length at least 5.

Note that if Q has endpoints v_0 and v_3, or v_1 and v_4, this creates a closed ear. Since all short ears were pendant, neither v_0 nor v_4 can be an internal vertex of a short ear, so (e) is maintained.

(O9) If Q has endpoints v_1 and v_3 and is not a 2-ear, we replace P by an ear of length at least 5, formed by the edges $\{v_0, v_1\}$ and $\{v_3, v_4\}$ of P and the edges of Q; in addition we replace Q by a 2-ear that consists of the remaining edges of P.

(O10) If Q is attached to P only at v_2 and is not a 2-ear, we replace P by an ear of length at least 5, formed by the edges $\{v_0, v_1\}$ and $\{v_1, v_2\}$ of P and the edges of Q; in addition we replace Q by a 2-ear with the remaining edges of P.

Operations (O9) and (O10) do not change the number of trivial ears, but each of them decreases the number of 4-ears.

If none of the above operations apply, every nontrivial ear attached to P is a pendant 2-ear, attached to P at v_1 and v_3 or only at v_2. Then we are done by Lemma 8.27. □

Lemma 8.29. *Given an ear-decomposition with* (d)–(g) *but not* (c)*, we can compute an ear-decomposition with* (d)–(g) *in polynomial time that either has more trivial ears, or has the same number of trivial ears but fewer 4-ears, or has the same number of trivial ears and the same number of 4-ears but fewer trivial ears that are incident to middle vertices of outer 4-ears.*

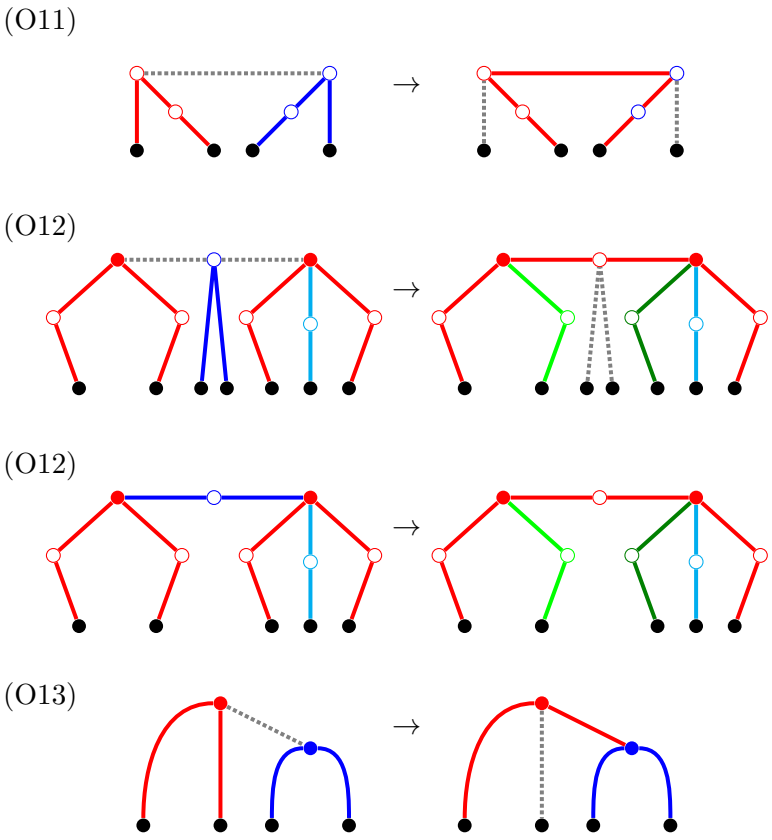

Figure 8.11: Ensuring condition (c). (O11): removing 3-ears whose middle vertices are adjacent; (O12): removing 4-ears whose middle vertices are both adjacent to the middle vertex of a 2-ear; (O13): an internal vertex of an outer ear (red) that is adjacent to an endpoint of this ear and to an internal vertex of a different outer ear (blue, 2-ear or 4-ear).

Proof. We may assume that (a) and (b) hold; otherwise Lemma 8.26 or Lemma 8.28 does the job. First we consider adjacent 3-ears:

(O11) If internal vertices of two (pendant) 3-ears are adjacent, we replace these ears by a pendant 5-ear. One trivial ear vanishes, but there are two new trivial ears.

Next we consider 2-ears adjacent to 4-ears:

(O12) If the middle vertices of two outer 4-ears are both adjacent to the middle vertex of a 2-ear, we replace these three ears by a 6-ear and two pendant 2-ears.

(O12) does not change the number of trivial ears but removes two 4-ears. Note that the new 5-ear or 6-ear in (O11) and (O12) can be closed, but then its endpoint was an endpoint of nontrivial ears before. Due to (a) this means that (e) is preserved.

Now we may assume that (a) and (b) hold and neither (O11) nor (O12) applies. Since (c) is violated, there are internal vertices v of an outer ears P and w of an outer ear Q that are adjacent, but P is not attached to Q at w, Q is not attached to P at v, and the vertices v and w are not both the middle vertex of a 4-ear. Thus, after possibly exchanging the roles of P and Q and of v and w, the vertex v is adjacent to an endpoint x of P. Since (O11) does not apply, P and Q cannot both be 3-ears. As all internal vertices of 3-ears are adjacent to an endpoint of the ear, we can assume that Q is not a 3-ear. (Otherwise, exchange the roles of P and Q and of v and w.) Hence, the final case is covered by the following operation:

(O13) If there are two outer ears P and Q none of which is attached to the other, and two internal vertices v of P and w of Q that are adjacent, and v is adjacent to an endpoint x of P, and Q is not a 3-ear, then we replace the edge $\{v, x\}$ by the edge $\{v, w\}$ in P so that then P is attached to Q. If this violates (a) or (b), we then apply Lemma 8.26 or Lemma 8.28.

Note that we avoid generating nonpendant 3-ears in order to maintain (f). If (a) and (b) are not violated by (O13), this does not change the number of trivial ears or 4-ears. In this case, P is a 2-ear, Q is an outer 4-ear, and w is the middle vertex of Q. Moreover, v is not adjacent to the middle vertex of any other outer 4-ear since (O12) does not apply. Therefore the new trivial ear is not incident to a middle internal vertex of an outer 4-ear and thus the operation (O13) reduces the number of trivial ears that are incident to middle vertices of outer 4-ears. □

Lemma 8.25, 8.26, 8.28, and 8.29 imply Theorem 8.24.

8.4 Optimizing outer ears and improving the lower bound

The ear-decomposition from Theorem 8.24 is the starting point for optimizing the outer ears. While we do not touch pendant or horizontal 4-ears, we will change short ears and vertical 4-ears. Like in [SV14], our goal is that as many short ears as possible form a forest. Because 2-ears entering 4-ears are not always useful, we will in addition try to make the vertical 4-ears pendant, by re-designing the 2-ears attached to them. The two subpaths of a 4-ear from the middle vertex to an endpoint will be part of this optimization, and might be replaced by attached 2-ears. See Figure 8.12.

For every 2-ear that will not be part of the forest or remains attached to an outer 4-ear, we will raise the lower bound. This includes in particular 2-ears attached to horizontal ears, which cannot be optimized.

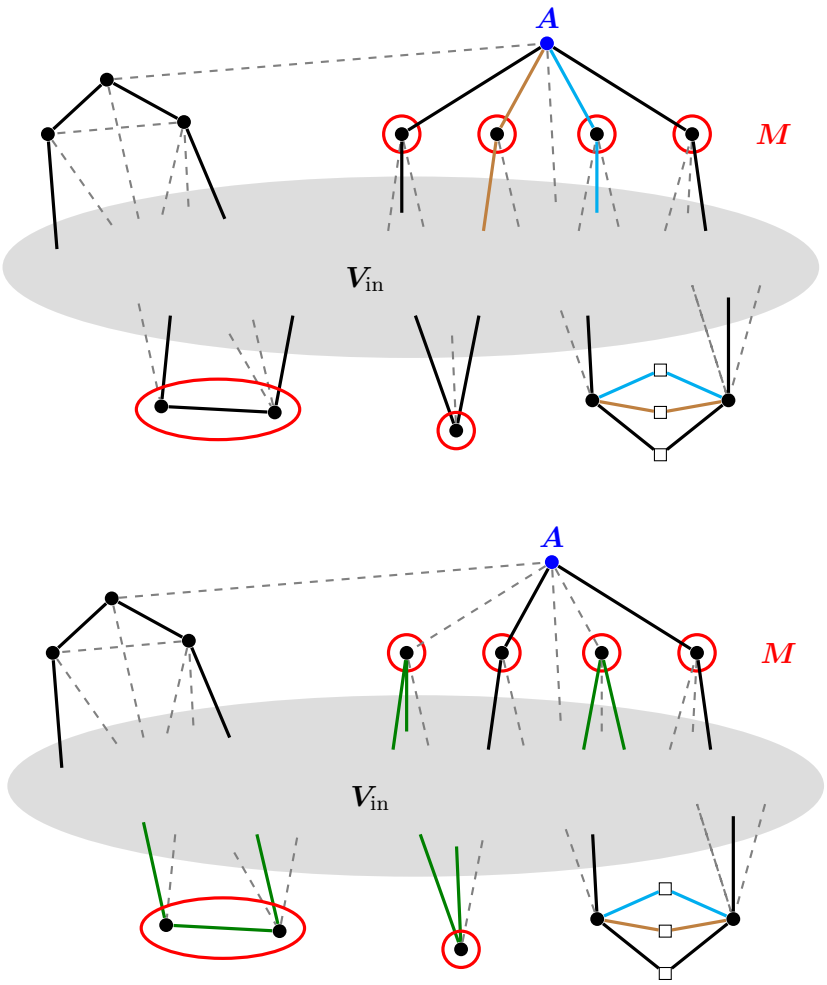

Figure 8.12: The upper picture shows the outer ears in an ear-decomposition as in Theorem 8.24. We see (clockwise) a pendant 4-ear, a vertical 4-ear with two attached 2-ears, a horizontal 4-ear with two attached 2-ears, a 2-ear, and a 3-ear. Dashed edges show trivial ears. The picture also shows the sets in M (red) and the set A (blue), in this case a singleton. The lower picture shows the optimized ear-decomposition, resulting from the green set of paths that is independent in both matroids. Note that not only short ears, but also the vertical 4-ear changes.

8.4.1 Matroid intersection

Given an ear-decomposition as in Theorem 8.24, let

$$\begin{aligned} M \;:=\; &\{\mathrm{in}(P) : P \text{ short ear not attached to a horizontal 4-ear}\} \;\dot\cup \\ &\{\{v\} : v \text{ a non-middle internal vertex of a vertical 4-ear}\}. \end{aligned}$$

Moreover, let A denote the set of middle vertices of vertical 4-ears and for $a \in A$ let

$$M(a) \;:=\; \{\{v\} \in M : \{v, a\} \in E(G)\}.$$

Note that $|M(a)| - 2$ is the number of 2-ears attached to a. Let

$$V_{\mathrm{in}} \;:=\; \{v \in V(P) : P \text{ an inner ear}\}.$$

The sets A, V_{in} and the elements of M are pairwise disjoint. Moreover, the vertices of G not contained in any of these sets are precisely the internal vertices of pendant or horizontal 4-ears and the internal vertices of 2-ears attached to horizontal 4-ears. See Figure 8.12.

Now let $\mathcal{P}_f$ for $f \in M$ denote the set of paths in G with the following two properties:

- The set of internal vertices of the path is f.
- The endpoints of the path are both contained in V_{in}.

We now define two matroids on the ground set $\bigcup_{f\in M} \mathcal{P}_f$.

The independent sets of the first matroid $\mathcal{M}_1$ are given by all sets $\mathcal{I} \subseteq \bigcup_{f\in M} \mathcal{P}_f$ such that $(V(G), \bigcup_{I\in\mathcal{I}} E(I))$ is a forest. It is clear that $\mathcal{M}_1$ is a graphic matroid (every path can be represented by an edge connecting its endpoints).

In the second matroid $\mathcal{M}_2$ a set $\mathcal{I} \subseteq \bigcup_{f\in M} \mathcal{P}_f$ is independent if and only if it fulfills the following two conditions:

(i) $|\mathcal{I} \cap \mathcal{P}_f| \leq 1$ for all $f \in M$

(ii) For every $a \in A$ we have

$$\left| \bigcup_{f\in M(a)} \mathcal{P}_f \cap \mathcal{I} \right| \;\leq\; |M(a)| - 2.$$

Note that (i) and (ii) characterize the independent sets of a laminar matroid (see Section 2.6).

We compute a maximum cardinality set $\mathcal{I} \subseteq \bigcup_{f\in M} \mathcal{P}_f$ such that $\mathcal{I}$ is an independent set in both matroids defined above.

The idea is that we want to choose as many short ears as possible to form a forest but leave two elements of each $M(a)$ unused; they will form the 4-ear with middle vertex a. Thus the vertical 4-ears can also change. We will formally describe how the new ear-decomposition is formed in the proof of Theorem 8.34. The paths in $\mathcal{I}$ will end up exactly as the forest of clean ears. In this forest we will not include clean ears that are attached to outer 4-ears.

8.4.2 Improving the lower bound

In this section we use the matroid intersection theorem (Theorem 2.13) to derive a lower bound on LP and hence on OPT. If $\mathcal{I}$ is smaller than the number of short ears, we will not be able to include all short ears of our optimized ear-decomposition into the forest of clean ears. We will make up for this by a larger lower bound.

For $f \in M$ let $U_f \subseteq V_{\text{in}}$ be the set containing all neighbors of elements of f in V_{in}. If $\mathcal{P}_f \neq \emptyset$, this is the set of endpoints of paths in $\mathcal{P}_f$. If $\mathcal{P}_f = \emptyset$, then U_f has a single element. In particular, we have $U_f \neq \emptyset$ for all $f \in M$.

For a set $W \subseteq V_{\text{in}}$ let

$$\text{sur}(W) \;:=\; |\{f \in M : U_f \subseteq W\}| - (|W| - 1)$$

be the surplus of W. For a partition $\mathcal{W}$ of V_{in} and a vertex $a \in A$, let

$$\text{sur}(a, \mathcal{W}) \;:=\; 2 - \sum_{W \in \mathcal{W}} |\{f \in M(a) : U_f \subseteq W\}|\,.$$

Finally, for $A' \subseteq A$ let

$$\mu(\mathcal{W}, A') \;:=\; \sum_{W \in \mathcal{W}} \text{sur}(W) + \sum_{a \in A'} \text{sur}(a, \mathcal{W}).$$

Lemma 8.30. *For a maximum cardinality set $\mathcal{I}$ that is independent in both matroids,*

$$|\mathcal{I}| \;=\; |M| - \max\left\{\mu(\mathcal{W}, A') : A' \subseteq A,\ \mathcal{W} \text{ partition of } V_{\text{in}}\right\}.$$

Proof. For any partition $\mathcal{W}$ of V_{in} and any $A' \subseteq A$ we set

$$\mathcal{Q}_{\mathcal{W}} \;:=\; \Big\{P \in \bigcup_{f \in M} \mathcal{P}_f \;:\; \exists W \in \mathcal{W} \text{ s.t. both endpoints of } P \text{ belong to } W\Big\}. \tag{8.6}$$

and

$$M'_{\mathcal{W},A'} \;:=\; \bigcup_{a \in A'} M(a) \cup \{f \in M : \mathcal{P}_f \subseteq \mathcal{Q}_{\mathcal{W}}\}.$$

Then, using the definitions of μ and sur:

$$\begin{aligned} \mu(\mathcal{W}, A') \;&=\; |\{f \in M : \mathcal{P}_f \subseteq \mathcal{Q}_{\mathcal{W}}\}| - \sum_{W \in \mathcal{W}} (|W| - 1) \\ &\quad + \sum_{a \in A'} \left(2 - |\{f \in M(a) : \mathcal{P}_f \subseteq \mathcal{Q}_{\mathcal{W}}\}|\right) \\ &=\; |M'_{\mathcal{W},A'}| - \sum_{a \in A'} (|M(a)| - 2) - \sum_{W \in \mathcal{W}} (|W| - 1). \end{aligned} \tag{8.7}$$

Moreover, every set $\mathcal{I}$ that is independent in both matroids has at most $\sum_{W \in \mathcal{W}}(|W|-1)$ elements of $\mathcal{Q}_{\mathcal{W}}$ (due to $\mathcal{M}_1$), at most $|M(a)| - 2$ elements of any $\bigcup_{f \in M(a)} \mathcal{P}_f$ (due to $\mathcal{M}_2$), and thus

$$|\mathcal{I}| \;\leq\; \sum_{W \in \mathcal{W}} (|W| - 1) + \sum_{a \in A'} (|M(a)| - 2) + |M \setminus M'_{\mathcal{W},A'}|.$$

This shows "$\leq$".

Now let r_1 denote the rank function of the matroid $\mathcal{M}_1$ and r_2 the rank function of the matroid $\mathcal{M}_2$. By the matroid intersection theorem ([Edm70], Theorem 2.13),

$$|\mathcal{I}| \;=\; \min\Big\{r_1(\mathcal{Q}) + r_2\big(\textstyle\bigcup_{f\in M}\mathcal{P}_f \setminus \mathcal{Q}\big) : \mathcal{Q} \subseteq \bigcup_{f\in M}\mathcal{P}_f\Big\}. \tag{8.8}$$

Let $\mathcal{Q}$ be a set attaining the minimum, and among these a maximal one. Let $\mathcal{W}$ contain the intersections of V_{in} with the vertex sets of the connected components of the graph $G_{\mathcal{Q}} := \big(V_{\text{in}} \cup \bigcup_{P\in\mathcal{Q}} V(P),\ \bigcup_{P\in\mathcal{Q}} E(P)\big)$. Then $\mathcal{Q} = \mathcal{Q}_{\mathcal{W}}$ by the maximality assumption.

The rank of $\mathcal{Q}$ in the graphic matroid $\mathcal{M}_1$ is

$$r_1(\mathcal{Q}) \;=\; \sum_{W\in\mathcal{W}} (|W| - 1). \tag{8.9}$$

Now let

$$A' \;:=\; \{a \in A : |\{f \in M(a) : \mathcal{P}_f \not\subseteq \mathcal{Q}\}| > |M(a)| - 2\}.$$

Then the rank of $\bigcup_{f\in M}\mathcal{P}_f \setminus \mathcal{Q}$ in the laminar matroid $\mathcal{M}_2$ is given by

$$r_2\big(\textstyle\bigcup_{f\in M}\mathcal{P}_f \setminus \mathcal{Q}\big) \;=\; |M \setminus M'_{\mathcal{W},A'}| + \sum_{a\in A'} (|M(a)| - 2). \tag{8.10}$$

(8.8), (8.9), (8.10), and (8.7) yield

$$|\mathcal{I}| \;=\; \sum_{W\in\mathcal{W}} (|W| - 1) + |M \setminus M'_{\mathcal{W},A'}| + \sum_{a\in A'} (|M(a)| - 2) \;=\; |M| - \mu(\mathcal{W}, A').$$

□

We denote by LP the value of the standard linear programming relaxation

$$\begin{aligned} \min\ & x(E(G)) \\ \text{s.t.}\quad & x(\delta(U)) \geq 2 \quad \text{for } \emptyset \subsetneq U \subsetneq V(G) \text{ with } |U \cap (\{s\} \triangle \{t\})| \text{ even}, \\ & x(\delta(U)) \geq 1 \quad \text{for } U \subsetneq V(G) \text{ with } |U \cap (\{s\} \triangle \{t\})| \text{ odd}, \\ & x_e \geq 0 \quad \text{for } e \in E(G). \end{aligned} \tag{8.11}$$

We now construct a dual solution to the LP (8.11). The dual is given by

$$\begin{aligned} \max\ & \sum_{\emptyset\subsetneq U\subsetneq V(G)} 2y(U) - \sum_{U:|U\cap(\{s\}\triangle\{t\})| \text{ odd}} y(U) \\ \text{s.t.}\ & \sum_{U:e\in\delta(U)} y(U) \leq 1 \quad \text{for } e \in E(G) \\ & y(U) \geq 0 \quad \text{for } \emptyset \subsetneq U \subsetneq V(G). \end{aligned} \tag{8.12}$$

Let $\mathcal{W}$ be a partition of V_{in} and $A' \subseteq A$ such that

$$|\mathcal{I}| \;=\; |M| - \mu(\mathcal{W}, A'). \tag{8.13}$$

(Such sets $\mathcal{W}$ and A' exist by Lemma 8.30.) Let $\bar{M} \subseteq M$ be the set of all $f \in M$ that have the following two properties:

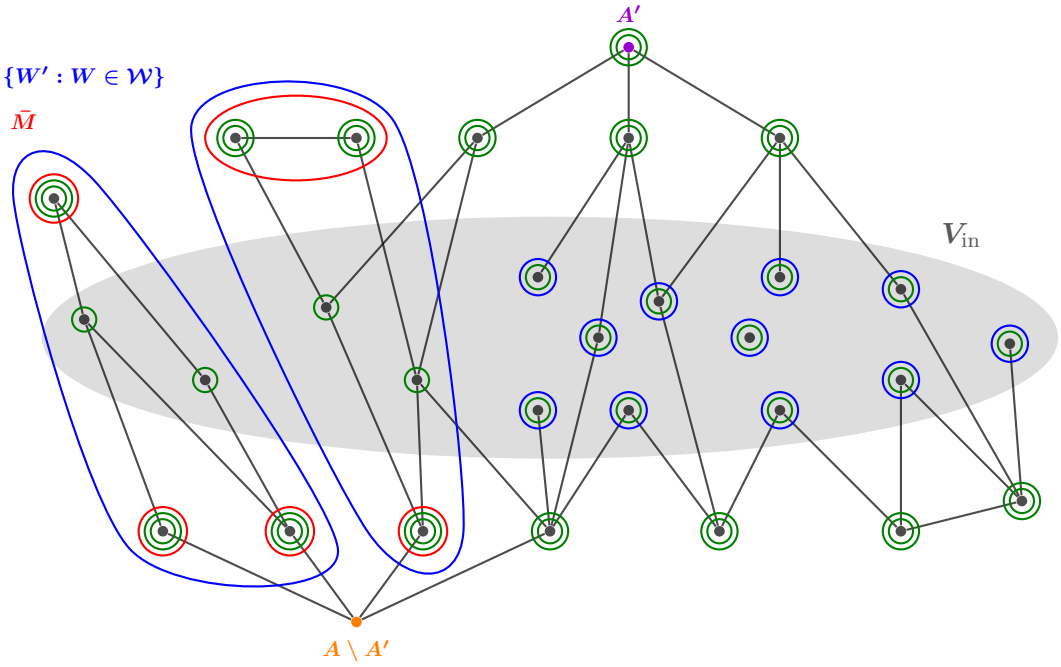

Figure 8.13: The dual solution y' is shown in green, blue and red. Every red, blue or green line around a set U indicates an (additional) value of $\frac{1}{4}$ of the corresponding dual variable $y'(U)$. Edges of the graph with both endpoints in V_{in} are not shown.

- $U_f \subseteq W$ for some $W \in \mathcal{W}$
- For every $a \in A'$ we have $f \notin M(a)$.

Let V_{hor} be the union of the set of internal vertices of horizontal 4-ears and of 2-ears attached to horizontal 4-ears. We first define a vector y' that is a feasible solution to the dual LP (8.12) if $V_{hor} = \emptyset$. (This is a fact that we will prove later, in Lemma 8.31).

- For every vertex $v \in V(G) \setminus V_{hor}$ we set

$$y'(\{v\}) \;:=\; \begin{cases} 0, & \text{if } v \in A \setminus A' \\ \frac{1}{4}, & \text{if } v \in V_{\text{in}} \\ \frac{1}{2}, & \text{else.} \end{cases} \tag{8.14}$$

  (green in Figure 8.13)

- For every set $W \in \mathcal{W}$ we define

$$W' \;:=\; W \cup \{v \in f : f \in \bar{M}, U_f \subseteq W\}$$

  and set $y'(W') := \frac{1}{4}$. If $|W'| = 1$, we instead increase $y'(W')$ by $\frac{1}{4}$ (from $\frac{1}{4}$ to $\frac{1}{2}$). (blue in Figure 8.13)

- We set $y'(f) := \frac{1}{4}$ for $f \in \bar{M}$. If $|f| = 1$, we instead increase $y'(f)$ by $\frac{1}{4}$ (from $\frac{1}{2}$ to $\frac{3}{4}$). (red in Figure 8.13)

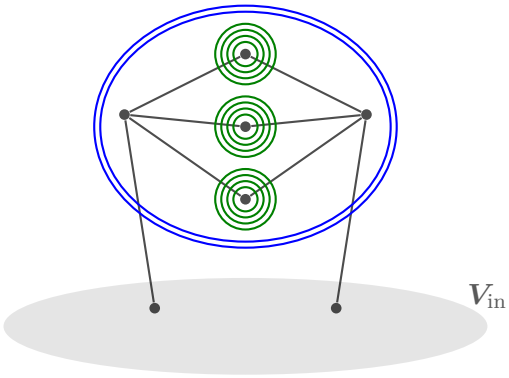

Figure 8.14: The dual solution y_{hor} for a horizontal 4-ear with one 2-ear attached to it. Every blue or green line around a set U indicates an (additional) value of $\frac{1}{4}$ of the corresponding dual variable $y_{hor}(U)$.

All other dual variables $y'(U)$ (for $\emptyset \subsetneq U \subsetneq V(G)$) are set to zero.

To construct a feasible dual solution also if $V_{hor} \neq \emptyset$, we define a vector y_{hor} and set $y := y' + y_{hor}$. To define y_{hor} we define for every horizontal 4-ear dual variables as follows: Let w be the middle vertex of P and let $Q_1, \ldots, Q_h$ be the 2-ears attached to P. Then we set

$$\begin{aligned} y_{hor}(\{w\}) &:= 1 \\ y_{hor}(\text{in}(Q_i)) &:= 1 \quad (\text{for } i = 1, \ldots, h) \\ y_{hor}(V(P) \cup V(Q_1) \cup \cdots \cup V(Q_h)) &:= \tfrac{1}{2}. \end{aligned}$$

See Figure 8.14. All other dual variables $y_{hor}(U)$ (for $\emptyset \subsetneq U \subsetneq V(G)$) are set to zero.

Lemma 8.31. *The vector $y = y' + y_{hor}$ is a feasible solution to the dual LP* (8.12) *of the LP* (8.11)*.*

Proof. We clearly have $y(U) \geq 0$ for $\emptyset \subsetneq U \subsetneq V(G)$. Moreover, we have for every vertex v that is contained neither in V_{hor} nor in $f \in \bar{M}$, that

$$\sum_{U : v \in U} y(U) \leq \tfrac{1}{2}. \tag{8.15}$$

Now let e be an edge of G.

Case 1: At least one of the endpoints of e is contained in V_{hor}.

If both endpoints of e are contained in V_{hor}, one endpoint v of e is a non-middle internal vertex of a horizontal 4-ear P, and the other endpoint w of e is the middle vertex of either P or a 2-ear attached to P at v. (This follows from Theorem 8.24 and the definitions of a horizontal 4-ear and V_{hor}.) Then, $\sum_{U : e \in \delta(U)} y(U) = y(\{w\}) = 1$. It remains to consider the case $e \in \delta(V_{hor})$ and without loss of generality $w \in V_{hor}$. By Theorem 8.24, this implies that $v \in V_{\text{in}}$. Moreover, by definition of horizontal 4-ears, w

is a non-middle internal vertex of a horizontal 4-ear. (Otherwise, the edge $\{v, w\}$ could not exist.) Using (8.15), this implies

$$\sum_{U:e\in\delta(U)} y(U) \;\leq\; \sum_{U:w\in U} y_{hor}(U) + \sum_{U:v\in U} y(U) \;\leq\; \tfrac{1}{2}+\tfrac{1}{2} \;=\; 1.$$

Case 2: None of the endpoints of e is contained in V_{hor}.

If none of the two endpoints of e is contained in $\bigcup_{f\in\bar{M}} f$, we have by (8.15), that $\sum_{U:e\in\delta(U)} y(U) \leq 1$. So we may assume that $e = \{v, w\}$ with $w \in f \in \bar{M}$. Then $\sum_{U:v\in U} y(U) = 1$. By Theorem 8.24, either

- $v \in A$ and $w = f \in M(v)$, or
- $v \in V_{\text{in}}$, or
- $v \in f$ and $f = \{v, w\}$ is the set of internal vertices of a 3-ear.

If $v \in A$, by definition of $\bar{M}$, we have $v \in A \setminus A'$, since $f \in \bar{M}$. Then $\sum_{U:v\in U} y(U) = 0$ and thus

$$\sum_{U:e\in\delta(U)} y(U) \;\leq\; \sum_{U:w\in U} y(U) \;=\; 1.$$

As $w \in f \in \bar{M}$, there exists a (unique) set $W \in \mathcal{W}$ with $U_f \subseteq W$ and $w \in W'$. If $v \in V_{\text{in}}$, we have $v \in U_f \subseteq W \subseteq W'$. Hence,

$$\sum_{U:e\in\delta(U)} y(U) \;\leq\; \sum_{U:w\in U} y(U) + \sum_{U:v\in U} y(U) - 2y(W') \;\leq\; 1+\tfrac{1}{2}-\tfrac{1}{2} \;=\; 1.$$

Finally, if $v \in f$ and $f = \{v, w\}$ is the set of internal vertices of a 3-ear, we have

$$\sum_{U:e\in\delta(U)} y(U) \;=\; y(\{v\}) + y(\{w\}) \;=\; \tfrac{1}{2}+\tfrac{1}{2} \;=\; 1. \qquad \square$$

Theorem 8.32. *If $\mathcal{I}$ is a maximum cardinality set that is independent in both matroids, then*

$$\text{LP} \;\geq\; n - 3 + \tfrac{1}{2}(k_2 + k_3 - |\mathcal{I}|).$$

Proof. By Lemma 8.31, the vector $y = y' + y_{hor}$ is a feasible solution to the dual of (8.11). Hence,

$$\text{LP} \;\geq \sum_{\emptyset\subsetneq U\subsetneq V(G)} 2y(U) - \sum_{U:|U\cap(\{s\}\triangle\{t\})| \text{ odd}} y(U).$$

Using the definitions of sur and $\bar{M}$, we get

$$\begin{aligned}\sum_{a\in A'} \text{sur}(a, \mathcal{W}) \;&=\; \sum_{a\in A'} \left(2 - |\{f \in M(a) : U_f \subseteq W, W \in \mathcal{W}\}|\right) \\ &=\; 2|A'| - |\{f \in M \setminus \bar{M} : U_f \subseteq W, W \in \mathcal{W}\}|\end{aligned}$$

and

$$\begin{aligned}\sum_{W\in\mathcal{W}} \operatorname{sur}(W) &= \sum_{W\in\mathcal{W}} \big(|\{f\in M : U_f\subseteq W\}| - (|W|-1)\big)\\ &= |\bar{M}| + |\{f\in M\setminus\bar{M} : U_f\subseteq W, W\in\mathcal{W}\}| - |V_{\text{in}}| + |\mathcal{W}|,\end{aligned}$$

which together with (8.13) implies

$$\tfrac{1}{2}(|M|-\mathcal{I}|) \;=\; \tfrac{1}{2}\mu(\mathcal{W},A') \;=\; |A'| + \tfrac{1}{2}|\bar{M}| - \tfrac{1}{2}|V_{\text{in}}| + \tfrac{1}{2}|\mathcal{W}|. \tag{8.16}$$

By construction of y', we have

$$\begin{aligned}\sum_{\emptyset\subsetneq U\subsetneq V(G)} 2y'(U) &= |V(G)\setminus V_{hor}| - |A\setminus A'| - \tfrac{1}{2}|V_{\text{in}}| + \tfrac{1}{2}|\mathcal{W}| + \tfrac{1}{2}|\bar{M}|\\ &= |V(G)\setminus V_{hor}| - |A| + \tfrac{1}{2}(|M| - |\mathcal{I}|),\end{aligned} \tag{8.17}$$

where we used (8.16) in the second equality.

Furthermore, for a horizontal 4-ear P with middle vertex w and 2-ears $Q_1, \dots, Q_h$ attached to P, we have $|V(P)\cup V(Q_1)\cup\cdots\cup V(Q_h)| = 3+h$ and thus

$$\begin{aligned}&y_{hor}(\{w\}) + \sum_{i=1}^{h} y_{hor}(\text{in}(Q_i)) + y_{hor}(V(P)\cup V(Q_1)\cup\cdots\cup V(Q_h))\\ &= 1 + h + \tfrac{1}{2}\\ &= \tfrac{1}{2}|V(P)\cup V(Q_1)\cup\cdots\cup V(Q_h)| + \tfrac{1}{2}h.\end{aligned}$$

This implies

$$\sum_{\emptyset\subsetneq U\subsetneq V(G)} 2y_{hor}(U) \;=\; |V_{hor}| + |\{Q : Q \text{ 2-ear attached to a horizontal 4-ear}\}|. \tag{8.18}$$

Combining (8.17) and (8.18), we get

$$\begin{aligned}\sum_{\emptyset\subsetneq U\subsetneq V(G)} 2y(U) &= n - |A| + \tfrac{1}{2}(|M| - |\mathcal{I}|)\\ &\quad + |\{Q : Q \text{ 2-ear attached to a horizontal 4-ear}\}|.\end{aligned}$$

Finally, we observe that we defined the dual solution y such that for every vertex v we have $\sum_{U:v\in U} y(U) \le \frac{3}{2}$. In particular,

$$\sum_{U:|U\cap(\{s\}\triangle\{t\})| \text{ odd}} y(U) \;\le\; \sum_{U:s\in U} y(U) + \sum_{U:t\in U} y(U) \;\le\; 3.$$

This shows

$$\begin{aligned}\text{LP} &\ge \sum_{\emptyset\subsetneq U\subsetneq V(G)} 2y(U) - \sum_{U:|U\cap(\{s\}\triangle\{t\})| \text{ odd}} y(U)\\ &\ge n - |A| + \tfrac{1}{2}(|M| - |\mathcal{I}|) + |\{Q : Q \text{ 2-ear attached to a horizontal 4-ear}\}| - 3\end{aligned}$$

$$\begin{aligned} &= n - 3 - |A| + \tfrac{1}{2}(2|A| + k_2 + k_3 - |\mathcal{I}|) \\ &\quad + \tfrac{1}{2}|\{Q : Q \text{ 2-ear attached to a horizontal 4-ear}\}| \\ &\geq n - 3 + \tfrac{1}{2}(k_2 + k_3 - |\mathcal{I}|) \end{aligned}$$

□

8.4.3 Optimizing outer ears

After optimizing the outer ears via the matroid intersection approach described above, we will distinguish between primary ears (those that were inner ears, plus clean ears that form a forest) and secondary ears. For secondary ears we will apply Lemma 8.1, and for secondary short ears we raise the lower bound using Theorem 8.32. For primary ears we will apply Theorem 8.22. Hence we need to bound the number of primary 4-ears, which correspond exactly to the blocked 4-ears before optimization. Their number can be bounded easily as follows.

Lemma 8.33. *Given an ear-decomposition as in Theorem 8.24, denote by $k_{4,\,blocked}$ and $k_{4,\,non\text{-}blocked}$ the number of blocked and non-blocked 4-ears, respectively. Then*

$$k_{4,\,blocked} \;\leq\; 2k_{\geq 5} + k_{4,\,non\text{-}blocked}.$$

Proof. Recall that a 4-ear is blocked if a closed ear is attached to it. Since all short ears are open and no closed 4-ear is attached to any closed 4-ear, the only closed ears attached to a 4-ear can be ears of length at least five, non-blocked 4-ears, and 4-ears to which a closed ear of length at least five is attached. Using that every closed ear is attached to exactly one ear, we get the result. □

We now describe in detail how we optimize the outer ears and summarize the results of this section in the following theorem.

Theorem 8.34. *Given a 2-vertex-connected graph G and $s, t \in V(G)$, we can compute a well-oriented ear-decomposition $P_1, \ldots, P_l$ of G and an index $p \leq l$ with the following properties in polynomial time. Call $P_1, \ldots, P_p$ the primary ears and $P_{p+1}, \ldots, P_l$ the secondary ears. Then:*

- *The primary short ears are clean; an ear is oriented if and only if it is short and primary.*
- $k_{4,\,\text{primary}} - 2k_{\geq 5,\,\text{primary}} \leq k_{4,\,\text{secondary}}$
- $\text{LP} \geq n - 3 + \frac{1}{2}k_{\text{clean, secondary}}$

where $k_{4,\,\text{primary}}$ is the number of primary 4-ears, $k_{4,\,\text{secondary}}$ is the number of secondary 4-ears, $k_{\geq 5,\,\text{primary}}$ is the number of primary ears with at least five edges, and $k_{\text{clean, secondary}}$ is the number of secondary clean ears.

Proof. We first compute an ear-decomposition as in Theorem 8.24. Then we compute a maximum independent set $\mathcal{I}$ in both matroids $\mathcal{M}_1$ and $\mathcal{M}_2$ by a matroid intersection algorithm. Then we modify the outer ears of our ear-decomposition as follows:

- For every short ear Q such that no internal vertex of Q is adjacent to an internal vertex of an outer 4-ear and $\mathcal{I} \cap \mathcal{P}_{\mathrm{in}(Q)} \neq \emptyset$, we replace Q by the unique element of $\mathcal{I} \cap \mathcal{P}_{\mathrm{in}(Q)}$.
- Let $a \in A$ and let $\{u\}, \{u'\}$ be two distinct elements of $M(a)$ such that both $\mathcal{I} \cap \mathcal{P}_{\{u\}}$ and $\mathcal{I} \cap \mathcal{P}_{\{u'\}}$ are empty. Then we replace the vertical 4-ear with middle vertex a and the 2-ears attached to it as follows. We choose a 4-ear with internal vertices u, a, u' and with endpoints in V_{in}. For every $\{v\} \in M(a) \setminus \{\{u\}, \{u'\}\}$ we choose a 2-ear with internal vertex v: if $\mathcal{I} \cap \mathcal{P}_{\{v\}} \neq \emptyset$, we choose the 2-ear to be the unique element of $\mathcal{I} \cap \mathcal{P}_{\{v\}}$; otherwise, we choose the 2-ear to consist of the edge $\{a, v\}$ and an arbitrary edge from v to a vertex in V_{in}.

This modification of the ear-decomposition does not change any inner ear. It does not change the total number of 4-ears. Moreover, all short ears are still pendant, and the (short) ears in $\mathcal{I}$ form a forest. We orient the clean ears in $\mathcal{I}$ so that we have a well-oriented ear-decomposition.

We declare an ear as primary if all its vertices belong to V_{in} or if it is a clean ear in $\mathcal{I}$. Other ears are secondary; their internal vertices do not belong to V_{in}. Since $\mathcal{I}$ contains only paths with both endpoints in V_{in}, we can reorder the ears so that the first ears $P_1, \dots, P_p$ are the primary ears (for some p).

Note that all ears of length at least five are primary, and the primary 4-ears are exactly those that were blocked before the optimization. Hence $k_{4,\mathrm{primary}} - 2k_{\geq 5,\mathrm{primary}} \leq k_{4,\mathrm{secondary}}$ follows from Lemma 8.33.

Finally, we have by Theorem 8.32 that

$$\begin{aligned} \mathrm{LP} &\geq n - 3 + \tfrac{1}{2}(k_2 + k_3 - |\mathcal{I}|) \\ &\geq n - 3 + \tfrac{1}{2} k_{\mathrm{clean,\,secondary}}, \end{aligned}$$

where the last inequality follows since clean secondary ears do not belong to $\mathcal{I}$. □

8.5 Ear-decompositions with many non-entered ears

Now we apply our ear induction to the optimized ear-decomposition, combining Theorem 8.34 and Theorem 8.22.

Theorem 8.35. *Given a graph G and $s, t \in V(G)$, we can compute a well-oriented ear-decomposition of G with π non-entered ears, and an s-t-tour with at most*

$$\left(\tfrac{3}{2} - \tfrac{1}{26} \cdot \tfrac{\pi}{n-1}\right) \mathrm{LP} + 3$$

edges in polynomial time.

Proof. We compute a well-oriented ear-decomposition of a graph G as in Theorem 8.34. Let π be the number of non-entered ears and V_{primary} the union of the vertex sets of all primary ears. We apply Lemma 8.1 to all nontrivial secondary ears (in reverse order). This yields a set $F' \subseteq \bigcup_{P\ \mathrm{secondary}} 2E(P)$ which is a T'-join for some T' with $T' \triangle \{s\} \triangle \{t\} \subseteq V_{\mathrm{primary}}$ such that $(V(G), F')/V_{\mathrm{primary}}$ is connected and

$$|F'| \leq \tfrac{3}{2}(n - |V_{\mathrm{primary}}|) - \tfrac{1}{2} k_{\mathrm{nontrivial,\,secondary}} + k_{\mathrm{clean,\,secondary}}.$$

Now let $T := T' \triangle \{s\} \triangle \{t\}$, and we can apply Theorem 8.22 to the primary ears. We get a T-join F such that (V_{primary}, F) is connected and

$$\begin{aligned} |F| &\leq \tfrac{3}{2}(|V_{\text{primary}}| - 1) - \tfrac{1}{26}k_{\text{non-entered, primary}} + \tfrac{1}{26}(k_{4,\text{ primary}} - 2k_{\geq 5,\text{ primary}}) \\ &\leq \tfrac{3}{2}(|V_{\text{primary}}| - 1) - \tfrac{1}{26}k_{\text{non-entered, primary}} + \tfrac{1}{26}k_{4,\text{ secondary}}. \end{aligned}$$

Then $F' \dot\cup F$ is an s-t-tour in G with at most

$$\begin{aligned} &\tfrac{3}{2}(n-1) - \tfrac{1}{2}k_{\text{nontrivial, secondary}} + k_{\text{clean, secondary}} \\ &\qquad - \tfrac{1}{26}k_{\text{non-entered, primary}} + \tfrac{1}{26}k_{4,\text{ secondary}} \\ \leq\ &\tfrac{3}{2}(n-1) - \tfrac{1}{26}\left(k_{\text{nontrivial, secondary}} + k_{\text{non-entered, primary}}\right) + \left(\tfrac{1}{2} + \tfrac{1}{26}\right)k_{\text{clean, secondary}} \\ =\ &\tfrac{3}{2}(n-1) - \tfrac{1}{26}\pi + \left(\tfrac{1}{2} + \tfrac{1}{26}\right)k_{\text{clean, secondary}} \end{aligned}$$

edges.

By Theorem 8.34 we have $\text{LP} \geq (n-1) + \frac{1}{2}k_{\text{clean, secondary}} - 2$. Thus, we have (using $\pi \leq n-1$)

$$\begin{aligned} \left(\tfrac{3}{2} - \tfrac{1}{26}\cdot\tfrac{\pi}{n-1}\right)\text{LP} &\geq \tfrac{3}{2}(n-1) - \tfrac{1}{26}\pi + \left(\tfrac{3}{2} - \tfrac{1}{26}\tfrac{\pi}{n-1}\right)\cdot\left(\tfrac{1}{2}k_{\text{clean, secondary}} - 2\right) \\ &\geq \tfrac{3}{2}(n-1) - \tfrac{1}{26}\pi + (\tfrac{3}{4} - \tfrac{1}{52})k_{\text{clean, secondary}} - 3. \end{aligned}$$

Hence we can bound the number of edges of our s-t-tour $F' \dot\cup F$ by

$$\left(\tfrac{3}{2} - \tfrac{1}{26}\cdot\tfrac{\pi}{n-1}\right)\text{LP} + 3.$$

□

8.6 Ear-decompositions with few non-entered ears

In this section we show how to compute a cheap s-t-tour if our well-oriented ear-decomposition has only few non-entered ears and s and t have small distance. As mentioned in Section 8.1.6, this is easy if we just want to achieve an approximation ratio smaller than $\frac{3}{2}$:

Lemma 8.36. *Let G be a graph with a well-oriented ear-decomposition, and $s, t \in V(G)$. Let π be the number of non-entered ears. Then we can compute an s-t-tour with at most*

$$\tfrac{4}{3}(n-1) + \tfrac{8}{3}\pi + \tfrac{1}{3}\text{dist}(s,t)$$

edges in polynomial time, where $\text{dist}(s,t)$ denotes the distance of s and t in G.

Proof. Let G' result from G by deleting the trivial ears. Then G' is still 2-edge-connected. The number of non-trivial ears is at most 2π because every entered ear is entered by a clean (and hence pendant and in particular non-entered) ear. Thus $|E(G')| \leq n - 1 + 2\pi$. We obtain an s-t-tour by adding a shortest T-join J to $E(G')$, where $T = \{s\} \triangle \{t\} \triangle \{v : v \text{ has odd degree in } G'\}$. Let P be a shortest s-t-path, and

let the vector $x \in \mathbb{R}^{E(G)}$ be the sum of the incidence vectors of $E(P)$ and $E(G')$, both multiplied by $\frac{1}{3}$. We claim that x belongs to the T-join polyhedron

$$\Big\{x \in [0,1]^{E(G)} : x(\delta_G(U)) \geq 1 \text{ for all } U \subseteq V(G) \text{ with } |U \cap T| \text{ odd}\Big\}, \tag{8.19}$$

which implies $|J| \leq x(E(G)) = \frac{1}{3}|E(G')| + \frac{1}{3}\mathrm{dist}(s,t)$. To see that x is in the T-join polyhedron (8.19), let $\emptyset \neq U \subsetneq V(G)$. Notice that $x(\delta_G(U)) < 1$ only if $|\delta_{G'}(U)| = 2$ and $\delta(U)$ does not separate s and t; but then $|U \cap T|$ is even. □

By applying the removable-pairing technique of Mömke and Svensson [MS16], we will get a better bound. The following lemma shows a new way to apply the removable-pairing technique to ear-decompositions. In contrast to Sebő and Vygen [SV14], we do not require the graph after deleting the trivial ears to be 2-vertex-connected. This requires a slight modification of their proof.

Lemma 8.37. *Let G be a graph with a well-oriented ear-decomposition, and $s, t \in V(G)$. Let π be the number of non-entered ears. Then we can compute an s-t-tour with at most*

$$\tfrac{4}{3}(n-1) + \tfrac{2}{3}\pi + \tfrac{1}{3}\mathrm{dist}(s,t)$$

edges in polynomial time, where $\mathrm{dist}(s,t)$ denotes the distance of s and t in G.

Proof. Let E' be the set of edges of nontrivial ears. Note that $(V(G), E')$ is 2-edge-connected, but in general not 2-vertex-connected.

For every non-entered ear choose one arbitrary edge of the ear and declare it removable. For every entered ear Q let $v \in \mathrm{in}(Q)$ such that there is an oriented ear entering Q at v; declare the two edges of Q that are incident to v a removable pair.

Let R be the set of all removable edges (including those in removable pairs), and let $\mathcal{P}$ be the set of removable pairs. We have $|E'| - (n-1)$ nontrivial ears and

$$|R| = 2(|E'| - (n-1)) - \pi. \tag{8.20}$$

Note that $(V(G), E' \setminus R')$ is connected for every subset $R' \subseteq R$ that contains at most one element of each removable pair. (Mömke and Svensson [MS16] called $(R, \mathcal{P})$ a removable pairing.)

We construct a 2-edge-connected auxiliary graph G' with edge weights c as follows. We begin with $(V(G), E')$, and initially all removable edges have weight -1 and all other edges have weight 1. Now consider the nontrivial ears in reverse order. Let P_i be the current ear. If P_i is non-entered, we do nothing. Otherwise there is a removable pair with edges $\{u,v\}$ and $\{v,w\}$ of P_i (then $v \in \mathrm{in}(P_i)$). We insert a new vertex v' and a new edge $\{v,v'\}$ with weight 0, and replace the edges $\{u,v\}$ and $\{v,w\}$ by $\{u,v'\}$ and $\{v',w\}$, both with weight -1. Note that the subgraph of oriented ears never changes.

To show that the graph remains 2-edge-connected we have to prove the new edge $\{v,v'\}$ is not a bridge (because contracting this edge results in the previous graph, which was 2-edge-connected by induction).

To prove that the new edge $\{v,v'\}$ is not a bridge, we construct paths from v to r and from v' to r, both not using this edge; here r denotes the initial vertex of the ear-decomposition (the vertex of P_0). From v' we follow the edge $\{v',w\}$ and then edges of P_i to an endpoint x of P_i. Since $x \in V_{i-1}$, there is a path from x to r in G_{i-1} (which

is a subgraph that we haven't changed yet). From v we follow the entering oriented ear (backwards) until we reach the root $r(v)$ of the connected component of oriented ears. This root belongs to $V_i \setminus \{v\}$. If $r(v) \in \text{in}(P_i) \setminus \{v\}$, we follow edges of P_i to an endpoint of P_i without visiting v. Then we reach a vertex in V_{i-1} and again have path to r within G_{i-1} from there.

The result is a 2-edge-connected weighted graph G', for which the contraction of the zero-weight edges would result in $(V(G), E')$. The removable edges have weight -1, other edges have weight 1. Finally, if $s \neq t$, we add a new edge d with endpoints s and t and weight $\text{dist}(s,t)$ to G' (possibly adding a parallel edge). We call the result G''. Let $T'' := \{v \in V(G'') : |\delta_{G''}(v)| \text{ odd}\}$.

Consider the vector $x \in \mathbb{R}^{E(G'')}$ whose entries are all $\frac{1}{3}$. We claim that x is in the T''-join polytope

$$\Big\{x \in [0,1]^{E(G'')} : |F| - x(F) + x(\delta_{G''}(U) \setminus F) \geq 1 \text{ for all } U \subseteq V(G'') \text{ and } F \subseteq \delta_{G''}(U) \text{with } |U \cap T''| + |F| \text{ odd}\Big\}, \tag{8.21}$$

which is the convex hull of incidence vectors of simple T''-joins (by simple we mean that no edge is used twice); see Theorem 2.17. To see that x is in (8.21), let $U \subsetneq V(G'')$ and $F \subseteq \delta_{G''}(U)$ with $|U \cap T''| + |F|$ odd. If $|\delta_{G''}(U)| \geq 3$, the inequality holds because every edge of $\delta_{G''}(U)$ contributes at least $\frac{1}{3}$ to the left-hand side. Otherwise $|\delta_{G''}(U)| = 2$ because G'' is 2-edge-connected. Hence $|U \cap T''|$ is even by definition of T''. Then $|F|$ is odd, so $|F| = 1$. Then $|F| - x(F) + x(\delta_{G''}(U) \setminus F) = 1 - \frac{1}{3} + \frac{1}{3} = 1$.

We conclude that x is in (8.21), and in fact in the face of this integral polytope defined by $x(\delta(v')) = 1$ for every new vertex v' (they have degree three). Faces of integral polytopes are integral, so x is in the convex hull of simple T''-joins that contain exactly one edge incident to each new vertex.

Hence there exists a simple T''-join J'' in G'' that contains exactly one edge incident to each new vertex, and with

$$c(J'') \;\leq\; c(x) \;=\; \tfrac{1}{3}|E'| - \tfrac{2}{3}|R| + \tfrac{1}{3}\text{dist}(s,t).$$

Such a J'' can be computed in polynomial time (using a standard reduction to weighted matching; see [SV14]).

Let D be the edge set of a shortest s-t-path if $d \in J''$, and $D = \emptyset$ otherwise. Let $T := \text{odd}(E') \triangle \{s\} \triangle \{t\}$. After contracting the zero-weight edges of G'', J'' corresponds to a simple T-join J in $(V(G), E' \,\dot\cup\, \{d\})$ with

$$|J \cap (E' \setminus R)| - |J \cap R| + |D| \;=\; c(J'') \;\leq\; \tfrac{1}{3}|E'| - \tfrac{2}{3}|R| + \tfrac{1}{3}\text{dist}(s,t),$$

not containing both edges of any removable pair. Then $(E' \setminus (J \cap R)) \,\dot\cup\, (J \cap (E' \setminus R)) \,\dot\cup\, D$ is an s-t-tour with at most $|E'| + c(J'')$ edges. The result now follows from (8.20). □

Theorem 8.38. *Given an instance of the s-t-path graph TSP with a 2-vertex-connected graph in which s and t have distance at most* $0.3334\,\text{OPT}$*, where* OPT *denotes the number of edges in the optimum solution, we can compute an s-t-tour with at most* $1.497\,\text{OPT}$ *edges in polynomial time.*

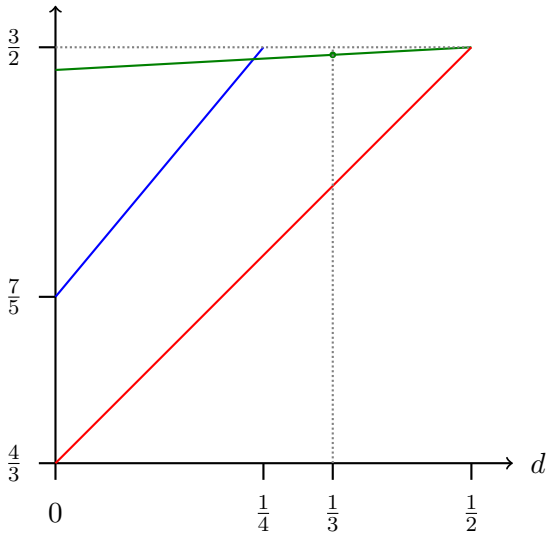

Figure 8.15: Lower bound (red) and upper bounds (green and blue) on the integrality ratio depending on $d = \frac{\text{dist}(s,t)}{n}$ (for n large enough). Section 8.8 shows that it suffices to consider instances with $d \leq \frac{1}{3} + \delta$ for arbitrary small $\delta > 0$.

Proof. We apply Theorem 8.35 and obtain a well-oriented ear-decomposition and an s-t-tour. If the number of non-entered ears is at least $\frac{13}{165}(n-1)$, then this s-t-tour has at most $(\frac{3}{2} - \frac{1}{330})\text{LP} + 3$ edges. If $n > 99000$, this is at most $1.497\,\text{LP} \leq 1.497\,\text{OPT}$ since $\text{LP} \geq n - 1$. For $n \leq 99000$ we can solve the instance by complete enumeration.

If the number of non-entered ears is at most $\frac{13}{165}(n-1)$, then Lemma 8.37 yields an s-t-tour with at most $(\frac{4}{3} + \frac{26}{495})(n-1) + \frac{0.3334}{3}\,\text{OPT} < 1.497\,\text{OPT}$ edges. □

8.7 Instances with large integrality ratio

The analysis of our algorithm allows us to give necessary and sufficient conditions for $\frac{\text{OPT}}{\text{LP}}$ being close to the integrality ratio $\frac{3}{2}$: we will show that the classical family of examples proving the lower bound of $\frac{3}{2}$ (see Figure 6.2) is "essentially the only such family of examples" (with large n).

First we analyze how the integrality ratio depends on the distance of s and t. For every $d \in [0, \frac{1}{2}]$, let $\rho(d, n)$ denote the integrality ratio for 2-vertex-connected instances with at least n vertices and $\text{dist}(s,t) \leq d \cdot n$. Let $\rho(d) = \lim_{n\to\infty} \rho(d, n)$. The proof of Theorem 8.38 shows that $\rho(d) < \frac{3}{2}$ for all $d < \frac{1}{2}$, and the bound improves as d decreases. More precisely, we get:

Lemma 8.39. *For all $d \in [0, \frac{1}{2}]$ and every integer $n \geq 2$ we have $\rho(d, n) \leq \frac{82+d}{55} + \frac{3}{n-1}$.*

Proof. We compute a well-oriented ear-decomposition as in Theorem 8.34 and consider two different bounds. Lemma 8.37 yields an s-t-tour with at most $\frac{4}{3}(n-1) + \frac{2}{3}\pi + \frac{1}{3}dn$ edges. Theorem 8.35 yields an s-t-tour with at most $(\frac{3}{2} - \frac{1}{26}\frac{\pi}{n-1})\text{LP} + 3$ edges. Taking $\frac{3}{55}$ times the first bound and $\frac{52}{55}$ times the second bound yields the upper bound $\frac{82+d}{55}\text{LP} + 3$. This shows $\rho(d, n) \leq \frac{82+d}{55} + \frac{3}{n-1}$. □

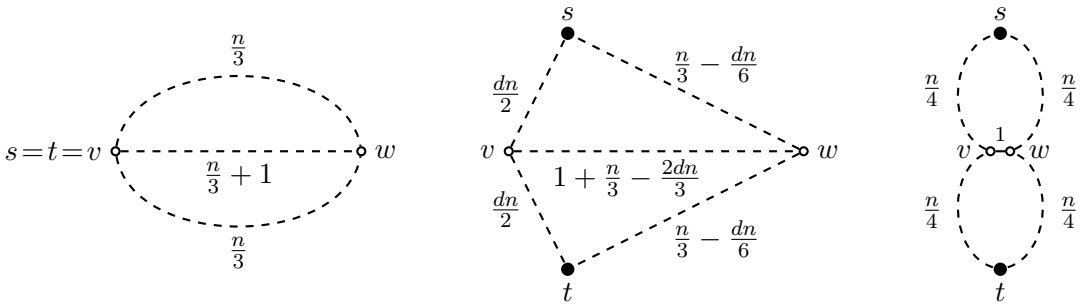

Figure 8.16: Middle picture: Instances with $\operatorname{dist}(s,t) = dn$ whose integrality ratio tends to $\frac{4+d}{3}$ as $n \to \infty$. Every dashed line represents a path with the indicated number of edges. The left picture shows instances with $d = 0$ whose integrality ratio tends to $\frac{4}{3}$ as $n \to \infty$; the right picture instances with $d = \frac{n}{2}$ whose integrality ratio tends to $\frac{3}{2}$ as $n \to \infty$. These two extreme cases are almost identical to the well-known examples that are the worst known for $s = t$ (left) and $s \neq t$ (right).

We can also give a lower bound and for very small d we can derive a better upper bound from [SV14] (cf. Figure 8.15):

Theorem 8.40. *For all $d \in [0, \frac{1}{2}]$ we have $\frac{4+d}{3} \leq \rho(d) \leq \min\{\frac{82+d}{55}, \frac{7+2d}{5}\}$.*

Proof. The first upper bound (green in Figure 8.15) follows from Lemma 8.39. If the distance from s to t is very small ($d < \frac{5}{21}$), a better upper bound $\rho(d) \leq \frac{7+2d}{5}$ (blue in Figure 8.15) can be derived from [SV14]; see Theorem 45 in the appendix of [TV18].

For the lower bound (red in Figure 8.15), we first observe that it suffices to prove the bound for rational numbers $d \in [0, \frac{1}{2}]$. Then there are infinitely many integers n for which $\frac{dn}{6}$ and $\frac{n}{3}$ are integers. We construct a graph G with n vertices (four of which are called s, t, v, w) and $n+1$ edges as follows: join s and v by a path of length $\frac{dn}{2}$, join t and v by a path of length $\frac{dn}{2}$, join s and w by a path of length $\frac{n}{3} - \frac{dn}{6}$, join t and w by a path of length $\frac{n}{3} - \frac{dn}{6}$, and join v and w by a path of length $1 + \frac{n}{3} - \frac{2dn}{3}$. See Figure 8.16, and observe that this graph indeed has n vertices.

For this instance of the s-t-path TSP we have $\mathrm{LP} \leq n+1$ because setting $x_e = 1$ for all $e \in E(G)$ is a feasible solution. However, any s-t-tour contains all but two edges, and some with two copies. Since the minimum $(\{s\} \triangle \{t\} \triangle \{v\} \triangle \{w\})$-join has $\min\{\frac{dn}{2} + \frac{dn}{2} + 1 + \frac{n}{3} - \frac{2dn}{3}, \frac{dn}{2} + \frac{n}{3} - \frac{dn}{6}\} = \frac{dn}{3} + \frac{n}{3}$ edges, any s-t-tour has at least $\frac{(4+d)n}{3} - 3$ edges. For $n \to \infty$, the ratio converges to $\frac{4+d}{3}$. □

Next, we show that for 2-vertex-connected instances with many vertices, we have that $\frac{\mathrm{OPT}}{\mathrm{LP}}$ is close to $\frac{3}{2}$ if and only if the length of every s-t-path in G is close to $\frac{n}{2}$. The proof of Theorem 8.41 shows that such instances are very similar to the classical examples; see Figure 8.17.

Theorem 8.41. *Let (G, s, t) be an instance of the s-t-path graph TSP where G is 2-vertex-connected. Then for $0 < \varepsilon < 0.001$ we have:*

(a) If $\frac{\mathrm{OPT}}{\mathrm{LP}} \geq \frac{3}{2} - \varepsilon$, then the length of every s-t-path in G is between $(\frac{1}{2} - 55 \cdot \varepsilon) \cdot (n-1) - 330$ and $(\frac{1}{2} + \varepsilon) \cdot (n-1)$.

(b) If the length of every s-t-path in G is between $(\frac{1}{2}-\varepsilon)\cdot n$ and $(\frac{1}{2}+\varepsilon)\cdot n$, then we have $\frac{\mathrm{OPT}}{\mathrm{LP}} \geq \frac{3}{2}\cdot\frac{1-9\varepsilon}{1+2\varepsilon}-\frac{3}{n}$.

Proof. Statement (a) is trivial if $n=1$; so we may assume $n\geq 2$. Let (G,s,t) be an instance of the s-t-path graph TSP and let P be an s-t-path in G. Then we can construct an s-t-tour from P that has $2(n-1)-|E(P)|$ edges: we complete P to a spanning tree S by adding $n-1-|E(P)|$ edges; then $E(P)\mathbin{\dot\cup} 2(S\setminus E(P))$ is an s-t-tour. Thus, if G contains an s-t-path P with $|E(P)|>(\frac{1}{2}+\varepsilon)\cdot(n-1)$, we have

$$\frac{\mathrm{OPT}}{\mathrm{LP}} \;\leq\; \frac{2(n-1)-|E(P)|}{n-1} \;<\; 2-\left(\frac{1}{2}+\varepsilon\right) \;=\; \frac{3}{2}-\varepsilon.$$

If G contains an s-t-path with at most $(\frac{1}{2}-55\varepsilon)\cdot(n-1)-330$ edges, i.e. $\mathrm{dist}(s,t)\leq(\frac{1}{2}-55\varepsilon)\cdot(n-1)-330$, then we can apply Lemma 8.39 to obtain

$$\frac{\mathrm{OPT}}{\mathrm{LP}} \;\leq\; \frac{82}{55}+\frac{(\frac{1}{2}-55\varepsilon)\cdot(n-1)-330}{55n}+\frac{3}{n-1} \;<\; \frac{3}{2}-\varepsilon,$$

where we used $\frac{n}{n-1}\leq 2$ and $55\varepsilon\leq\frac{1}{2}$ in the last inequality. This completes the proof of (a).

To prove (b), suppose that every s-t-path in G has between $(\frac{1}{2}-\varepsilon)\cdot n$ and $(\frac{1}{2}+\varepsilon)\cdot n$ many edges. First, we give an upper bound on the optimum LP value. Because G is 2-vertex-connected, it contains two vertex disjoint s-t-paths P_1 and P_2. Let $S\subseteq E(G)$ be a minimal subset of edges such that $(V, E(P_1)\cup E(P_2)\cup S)$ is connected. Since $E(P_1)\mathbin{\dot\cup} E(P_2)$ is the edge set of a cycle of length at least $(1-2\varepsilon)\cdot n$, we have $|S|=n-|E(P_1)\cup E(P_2)|\leq 2\varepsilon\cdot n$. Moreover,

$$x_e \;:=\; \begin{cases} 1 & ,\text{ if } e\in E(P_1)\cup E(P_2)\\ 2 & ,\text{ if } e\in S\\ 0 & ,\text{ else}\end{cases}$$

is a feasible solution to (Graph TSPP LP), implying $\mathrm{LP}\leq x(E)\leq(1+2\varepsilon)\cdot n$. Before giving a lower bound on OPT, we first prove that G has a lot of structure and, informally speaking, looks essentially like the classical examples with integrality ratio close to $\frac{3}{2}$.

For $i\in\{0,\ldots,\mathrm{dist}(s,t)-1\}$, let $U_i:=\{v\in V:\mathrm{dist}(s,v)=i\}$ be the set of vertices with distance i from s. Moreover, for $k:=\mathrm{dist}(s,t)\leq(\frac{1}{2}+\varepsilon)\cdot n$, let $U_k:=\{v\in V:\mathrm{dist}(s,v)\geq k\}$; then $V=U_0\mathbin{\dot\cup}U_1\mathbin{\dot\cup}\ldots,\mathbin{\dot\cup}U_k$. Notice that then every edge of G has either both endpoints in the same set U_i or has endpoints in two consecutive sets U_{i-1} and U_i. Moreover, the paths P_1 and P_2 both contain at least one vertex in each of the sets $U_0,U_1,\ldots,U_k$. Let $v_0=w_0=s$ and $v_k=w_k=t$. For $i\in\{1,\ldots,k-1\}$ let v_i be the first vertex that P_1 visits in U_i and w_i the first vertex that P_2 visits in U_i; then the vertices $v_0,v_1,\ldots,v_k$ are visited by P_1 in this order and similarly the vertices $w_0,w_1,\ldots,w_k$ are visited by P_2 in this order. For $i\in\{1,\ldots,k-1\}$ we have $v_i\neq w_i$ because P_1 and P_2 are vertex disjoint. We call a set U_i with $U_i\supsetneq\{v_i,w_i\}$ *large* and a set U_i with $U_i=\{v_i,w_i\}$ *small*. We have

$$|\{s,t,v_1,\ldots,v_{k-1},w_1,\ldots,w_{k-1}\}|=2k\geq(1-2\varepsilon)\cdot n.$$

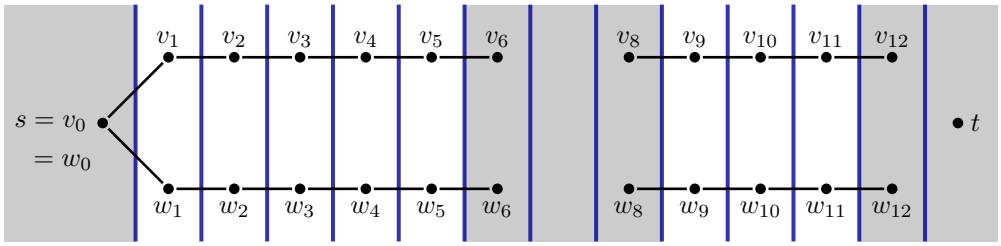

Figure 8.17: Illustration of the proof of Theorem 8.41 (b): an instance of the s-t-path graph TSP in which every s-t-path has length close to $\frac{n}{2}$. The vertical blue lines show the cuts $\delta(U_0 \cup U_1 \cup \cdots \cup U_i)$ with $0 \leq i < k$. The white area shows sets U_i with $i \in \{1, \ldots, k-1\} \setminus I$, whereas the gray areas show sets U_i with $i \in I \cup \{0, k\}$; in this example $k = \text{dist}(s, t) = 13$. The black lines show edges with at least one endpoint in a white set, i.e. a set U_i with $i \in \{1, \ldots, k-1\} \setminus I$. The figure shows only vertices that are endpoints of these edges and it shows s and t; the gray sets U_i with $i \in I \cup \{0, k\}$ might contain further vertices and edges. However, the gray sets constitute only a small fraction of the overall graph: they contain only $O(\varepsilon n)$ many vertices.

Therefore, at most $2\varepsilon \cdot n$ sets U_i are large. Note that the small sets U_i are visited exactly once by each of the paths P_1 and P_2 and they are visited in an order of increasing index i.

If there exists an edge $\{v, w\} \in E(G)$ with $v \in V(P_1) \setminus \{s, t\}$ and $w \in V(P_2) \setminus \{s, t\}$, we call the vertices v and w *bad*. Let I be the set of all $i \in \{0, \ldots, k\}$ such that U_{i-1}, U_i or U_{i+1} is large, or U_i contains a bad vertex. Now we observe that by definition of I, every set U_i with $i \in \{1, \ldots, k-1\} \setminus I$ has exactly two elements and these are the vertices v_i and w_i. Moreover, both of these vertices have degree 2 and we have $\delta(v_i) = \{\{v_{i-1}, v_i\}, \{v_i, v_{i+1}\}\} \subseteq E(P_1)$ and $\delta(w_i) = \{\{w_{i-1}, w_i\}, \{w_i, w_{i+1}\}\} \subseteq E(P_2)$; see Figure 8.17.

Claim 1. *We have* $|I| \leq 12\varepsilon \cdot n$.

Proof. Recall that there are at most $2\varepsilon \cdot n$ large sets U_i. Therefore there are at most $6\varepsilon \cdot n$ sets U_i such that U_{i-1}, U_i or U_{i+1} is large. Now suppose there are more than $6\varepsilon \cdot n$ other sets U_i that contain a bad vertex. Then there exist edges $e_1 = \{v_{i_1}, w_{j_1}\}, \ldots e_l = \{v_{i_l}, w_{j_l}\}$ for some $l > 2\varepsilon \cdot n$ such that $1 < i_1 < i_2 < \cdots < i_l < k$ and $1 < j_1 < j_2 < \cdots < j_l < k$; see Figure 8.18. Here we used that every edge of G has either both endpoints in the same set U_i or has endpoints in two consecutive sets U_{i-1} and U_i.

We will show that then G contains an s-t-path of length at least $l + (\frac{1}{2} - \varepsilon) \cdot n$, contradicting the fact that every s-t-path has length at most $(\frac{1}{2} + \varepsilon) \cdot n$. For $0 \leq i \leq j \leq k$, let P_1^{ij} denote the subpath of P_1 from v_i to v_j and let P_2^{ij} denote the subpath of P_2 from w_i to w_j. Then we can construct two s-t-paths Q_1 and Q_2 as follows. We obtain Q_1 by appending

$$P_1^{0i_1}, e_1, P_2^{j_1 j_2}, e_2, P_1^{i_2 i_3}, e_3, P_2^{j_3 j_4}, e_4, \ldots$$

and we obtain Q_2 by appending

$$P_2^{0j_1}, e_1, P_1^{i_1 i_2}, e_2, P_2^{j_2 j_3}, e_3, P_1^{i_3 i_4}, e_4, \ldots \quad ;$$

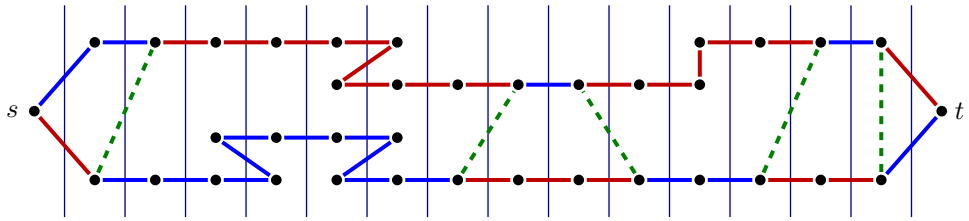

Figure 8.18: The vertical thin lines indicate the boundaries between the sets U_i for different i. The thick solid edges show the two vertex disjoint s-t-paths P_1 (upper half of the picture) and P_2 (lower half of the picture). The green dashed edges are $e_1, \ldots, e_l$. (The graph G might also contain further vertices and edges not shown in the picture.) The edge set of the path Q_1 is the union of the blue and the green edges; the edge set of the path Q_2 is the union of the red and green edges.

see Figure 8.18. Then every edge of $E(P_1) \dot\cup E(P_2)$ is contained in exactly one of the two paths Q_1 and Q_2 and the edges $e_1, \ldots, e_l$ are contained in both paths. The longer of these paths has length at least

$$\frac{1}{2}(|E(P_1)| + |E(P_2)| + 2l) \geq l + (\tfrac{1}{2} - \varepsilon) \cdot n > (\tfrac{1}{2} + \varepsilon) \cdot n,$$

a contradiction. □

(proof of Claim 1)

Finally, we give a lower bound on the length OPT of an optimum s-t-tour. To this end we will show that such a tour needs to contain at least $k - |I| - 2$ parallel edges. Because the s-t-tour needs to be connected, this implies

$$\text{OPT} \;\geq\; n - 1 + k - |I| - 2 \;\geq\; \left(\tfrac{3}{2} - 13\varepsilon\right) n - 3.$$

Therefore,

$$\frac{\text{OPT}}{\text{LP}} \;\geq\; \frac{\left(\frac{3}{2} - 13\varepsilon\right) \cdot n - 3}{(1 + 2\varepsilon) \cdot n} \;\geq\; \frac{3}{2} \cdot \frac{1 - 9\varepsilon}{1 + 2\varepsilon} - \frac{3}{n}.$$

It remains to show that every s-t-tour in G contains at least $k - |I| - 2$ parallel edges. To prove this, consider any arbitrary, but fixed s-t-tour F. Now consider indices $0 < i_{\min} \leq i_{\max} < k$ such that for $i_{\min} \leq i \leq i_{\max}$ we have $i \notin I$. Then the edges incident to vertices in $U_{i_{\min}}, U_{i_{\min}+1} \ldots, U_{i_{\max}}$ are precisely the edges of the two vertex disjoint paths with vertices $v_{i_{\min}-1}, v_{i_{\min}}, \ldots, v_{i_{\max}}, v_{i_{\max}+1}$ and $w_{i_{\min}-1}, w_{i_{\min}}, \ldots, w_{i_{\max}}, w_{i_{\max}+1}$ (visited in this order); see Figure 8.17. Because the s-t-tour F is connected and because G contains no edge from a set U_i with $i < i_{\min}$ to a set U_j with $j > i_{\max}$, all of the edges of these two paths but one must be part of F. Moreover, we have $s \in U_0$ and $t \in U_k$ and therefore every cut of the form $\delta(U_0 \cup U_1 \cup \cdots \cup U_i)$ with $0 \leq i < k$ must contain an odd number of edges of F. Using this for $i = i_{\min} - 1, i_{\min}, \ldots, i_{\max}$, we obtain that $i_{\max} - i_{\min} + 1$ of the edges incident to a vertex in $U_{i_{\min}} \cup \cdots \cup U_{i_{\max}}$ must be contained twice in the s-t-tour F.

Applying the above argument to every maximal set $\{i_{\min}, i_{\min} + 1, \ldots, i_{\max}\}$ of consecutive indices that are not contained in $I \cup \{0, k\}$, yields that F contains at least $k - |I| - 2$ parallel edges. □

We remark that Theorem 8.41 can also be generalized to instances where the graph G is not 2-vertex-connected by applying Theorem 8.41 to every 2-vertex-connected component of G (as explained in Section 8.1.1). Our goal was to show that the instances with integrality ratio close to $\frac{3}{2}$ look "essentially like the classical examples" and we did not optimize the constants in the bounds in Theorem 8.41.

8.8 A 1.497-approximation algorithm

We have shown that there is a 1.497-approximation algorithm for instances of the s-t-path graph TSP with $\text{dist}(s,t) \leq \left(\frac{1}{3}+\varepsilon\right) \cdot \text{OPT}$ for some $\varepsilon > 0$ (Theorem 8.38). In this section we prove a result from [Tra17] that implies that this is sufficient in order to obtain a 1.497-approximation algorithm for the s-t-path graph TSP without any restriction on the distance of s and t.

Theorem 8.42. *Let $\varepsilon > 0$ and $\alpha > 1$ be constants. Assume there exists a polynomial-time algorithm that computes a solution with at most $\alpha \cdot \text{OPT}$ edges for instances of the s-t-path graph TSP with a 2-vertex-connected graph G and $\text{dist}(s,t) \leq (\frac{1}{3}+\varepsilon) \cdot \text{OPT}$. Then there exists an α-approximation algorithm for the s-t-path graph TSP.*

Theorem 8.42 and Theorem 8.38 directly imply the following result:

Theorem 8.43. *There is a 1.497-approximation algorithm for the s-t-path graph TSP.*

This shows that one can achieve an approximation ratio below the integrality ratio, even while using the LP in the analysis. We did not attempt to optimize the running time or the approximation ratio of this algorithm and in Chapter 9 we will give a different algorithm achieving a better approximation ratio.

We now give a proof of Theorem 8.42 for completeness, but also because the algorithm that we will present in Chapter 9 uses ideas from this proof.

8.8.1 Outline

Let us first sketch the main idea of the proof, which is inspired by an algorithm for the orienteering problem by Blum, Chawla, Karger, Lane, Meyerson, and Minkoff [BCK+07]. We consider the sets $L_i := \{v \in V : \text{dist}(s,v) \leq i\}$ for $i = 0, \dots, \text{dist}(s,t) - 1$. Then the cuts $\delta(L_i)$ induced by these sets are disjoint. If now $\text{dist}(s,t) > \left(\frac{1}{3}+\varepsilon\right) \cdot \text{OPT}$, then the average number of edges in a cut $\delta(L_i)$ is significantly less than 3. Because $s \in L_i \subseteq V \setminus \{t\}$ for all $i \in \{0, \dots, \text{dist}(s,t)-1\}$, all these cuts $\delta(L_i)$ contain an odd number of edges of every s-t-tour. Therefore, a fixed optimum s-t-tour F^* must intersect a constant fraction of the cuts $\delta(L_i)$ exactly in one edge, i.e. $|F^* \cap \delta(L_i)| = 1$. Let us call such a cut a 1-cut. The 1-cuts separate the instance into several instances of the s-t-path graph TSP; see Figure 8.19.

We will show that we can guess the 1-cuts and the edges of F^* that are contained in them (blue in Figure 8.19) by dynamic programming. If we know these cuts and edges, we can proceed as follows. We apply some constant-factor approximation algorithm for the s-t-path graph TSP (e.g. Christofides' algorithm with approximation ratio $\beta = \frac{5}{3}$) to each of the sub-instances between two consecutive 1-cuts. If $\text{dist}(s,t) > (\frac{1}{3}+\varepsilon) \cdot \text{OPT}$, then this yields an approximation ratio $\beta' < \beta$. Otherwise, we can apply the given α-approximation algorithm for the case where the distance of s and t is relatively small.

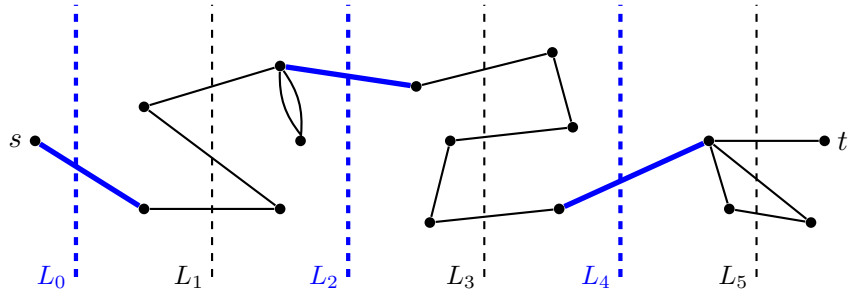

Figure 8.19: An optimum s-t-tour F^*. The dashed vertical lines show the chain $\mathcal{L}$. The family $\mathcal{L}^{F^*} \subseteq \mathcal{L}$ of 1-cuts is shown in blue. The thick blue edges are those edges of F^* that are contained in a 1-cut. The 1-cuts separate the instance into several smaller instances of the s-t-path TSP. In this example there are 4 such subinstances, where the first instance is trivial and consists only of the vertex s.

For $\beta > \alpha$, we therefore obtained an approximation ratio of $\min\{\beta', \alpha\} < \beta$. Then we iterate this procedure of strengthening our β-approximation algorithm until we obtain an α-approximation algorithm.

Our proof of Theorem 8.42 also applies to the more general weighted version of the s-t-path TSP. More precisely, we will also show the following.

Theorem 8.44. *Let $\varepsilon > 0$ and $\alpha > 1$ be constants. Assume there exists a polynomial-time algorithm that computes a solution of cost at most $\alpha \cdot \mathrm{OPT}$ for instances of the s-t-path TSP with a 2-vertex-connected graph G and $dist(s,t) = c(s,t) \leq (\frac{1}{3} + \varepsilon) \cdot \mathrm{OPT}$. Then there exists an α-approximation algorithm for the s-t-path TSP.*

8.8.2 Strengthening a constant-factor approximation

We now consider the s-t-path TSP with general nonnegative edge weights c. In this section we will prove the following lemma.

Lemma 8.45. *Let $\varepsilon > 0$ and $\alpha > 1$. Let $\mathcal{A}$ be a polynomial-time algorithm for the s-t-path TSP that computes for every instance with $dist(s,t) = c(s,t) \leq (\frac{1}{3} + \varepsilon) \cdot \mathrm{OPT}$ a solution of cost at most $\alpha \cdot \mathrm{OPT}$. Moreover, let $\beta > 1$ and let $\mathcal{B}$ be a polynomial-time β-approximation algorithm for the s-t-path TSP.*

Then there is a polynomial-time algorithm for the s-t-path TSP with approximation ratio

$$\max\{\alpha, \beta - \tfrac{3}{2}\varepsilon \cdot (\alpha - 1)\}.$$

This algorithm when called on an instance (G, c, s, t), uses $\mathcal{A}$ and $\mathcal{B}$ only for instances (G', c', s', t) where (G', c') is an induced subgraph of the original graph (G, c).

Because we need to call $\mathcal{A}$ and $\mathcal{B}$ only on sub-instances of the original instance, we can apply Lemma 8.45 not only to the weighted case, but also to the graph version.

We now describe our algorithm to prove Lemma 8.45. If $\beta \leq \alpha$, the statement is trivial and we can simply apply the given β-approximation. So we may assume $\beta > \alpha > 1$. First, we sort the vertices $V = \{v_1, \ldots, v_n\}$ such that $\mathrm{dist}(s, v_1) \leq \mathrm{dist}(s, v_2) \leq \cdots \leq \mathrm{dist}(s, v_n)$. We now consider the chain

$$\mathcal{L} := \{\{v_1, \ldots, v_i\} \ : \ \mathrm{dist}(s, v_i) < \mathrm{dist}(s, t),\ 1 \leq i < n\}.$$

Let $k := |\mathcal{L}|$ and for $i \in \{1, \dots, k\}$ let $L_i := \{v_1, \dots, v_i\}$. Note that then $\{s\} \subseteq L_1 \subseteq \cdots \subseteq L_k \subseteq V \setminus \{t\}$. We also define values

$$y_{L_i} := \operatorname{dist}(s, v_{i+1}) - \operatorname{dist}(s, v_i) \;\geq\; 0$$

for $i \in \{1, \dots, k\}$ and we set $y_i := 0$ for $i = k+1, \dots n-1$. The following lemma describes the key property of the chain $\mathcal{L}$ that we will need for the analysis of our dynamic programming algorithm.

Lemma 8.46. *Let F be an s-t-tour and let $\mathcal{L}^F := \{L \in \mathcal{L} : |F \cap \delta(L)| = 1\}$. Then we have*

$$c\left(F \cap \bigcup_{L \in \mathcal{L}^F} \delta(L)\right) \geq \frac{1}{2} \cdot (3 \cdot \operatorname{dist}(s,t) - c(F)) \,.$$

Proof. We have $\sum_{i=1}^{k} y_{L_i} = \operatorname{dist}(s,t)$. Moreover, for every edge $e = \{v_i, v_j\} \in F$ with $i < j$ we have

$$c(e) \;\geq\; \operatorname{dist}(s, v_j) - \operatorname{dist}(s, v_i) \;=\; \sum_{l=1}^{j-1} y_{L_l} - \sum_{l=1}^{i-1} y_{L_l} \;=\; \sum_{l=i}^{j-1} y_{L_l} \;=\; \sum_{L \in \mathcal{L}: e \in \delta(L)} y_L. \tag{8.22}$$

By the definition of $\mathcal{L}^F$ and by (8.22), we have

$$c\left(F \cap \bigcup_{L \in \mathcal{L}^F} \delta(L)\right) \;\geq\; \sum_{e \in F} \sum_{L \in \mathcal{L}^F : e \in \delta(L)} y_L \;=\; \sum_{L \in \mathcal{L}^F} y_L. \tag{8.23}$$

Because F is an s-t-tour, $|F \cap \delta(L)|$ is odd for every $L \in \mathcal{L}$; here we used $s \in L \subseteq V \setminus \{t\}$. Therefore, using (8.22), we get

$$\begin{aligned} c(F) \;&=\; \sum_{e \in F} c(e) \;\geq\; \sum_{e \in F} \sum_{L \in \mathcal{L}: e \in \delta(L)} y_L \;=\; \sum_{L \in \mathcal{L}} |F \cap \delta(L)| \cdot y_L \\ &\geq\; \sum_{L \in \mathcal{L}^F} y_L + \sum_{L \in \mathcal{L} \setminus \mathcal{L}^F} 3 \cdot y_L \;=\; 3 \cdot \sum_{L \in \mathcal{L}} y_L - 2 \cdot \sum_{L \in \mathcal{L}^F} y_L \\ &= 3 \cdot \operatorname{dist}(s,t) - 2 \cdot \sum_{L \in \mathcal{L}^F} y_L. \end{aligned}$$

Combining this with (8.23) completes the proof. □

Our dynamic programming algorithm works as follows. We compute for every set $L \in \mathcal{L} \cup \{V\}$ and every vertex $v \in L$ an s-v-tour $F_{L,v}$ in $G[L]$. For this we consider the sets $L \in \mathcal{L} \cup \{V\}$ in an order of increasing cardinality. Then for every vertex v we enumerate all pairs $(L', \{v', w'\})$ with $L' \in \mathcal{L}$, $L' \subsetneq L$, $\{v', w'\} \in \delta(L') \setminus \delta(L)$, and $v' \in L'$; see Figure 8.20. Since we consider the elements of $\mathcal{L} \cup \{V\}$ in an order of increasing cardinality, we have already computed an s-v'-tour $F_{L',v'}$ in $G[L']$. We apply the given β-approximation algorithm $\mathcal{B}$ to to obtain a w'-v-tour F' in $G[L \setminus L']$. Then $F_{L',v'} \,\dot\cup\, \{\{v', w'\}\} \,\dot\cup\, F'$ is an s-v-tour in $G[L]$. Finally, we choose $F_{L,v}$ to be the cheapest such tour that results from any choice of $(L', \{v', w'\})$ or from applying the algorithm $\mathcal{B}$ to obtain an s-v-tour in $G[L]$.

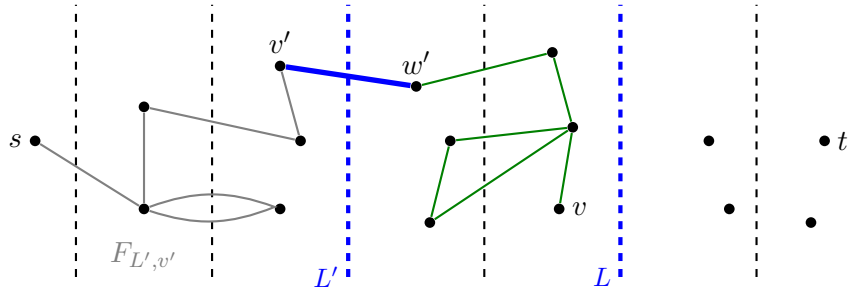

Figure 8.20: Illustration of the dynamic programming algorithm. The dashed vertical lines show the chain $\mathcal{L}$. Possible sets L and L' and an edge $\{v', w'\} \in \delta(L') \setminus \delta(L)$ are shown in blue. The gray edges show the s-v'-tour $F_{L',v'}$ in $G[L']$ and the green edges show the w'-v-tour in $G[L \setminus L']$ computed by the algorithm $\mathcal{B}$. The union of these two tours and the edge $\{v', w'\}$ is an s-v-tour in $G[L]$ and is a possible choice of $F_{L,v}$.

Lemma 8.47. *Let F be an s-t-tour and let $\mathcal{L}^F := \{L \in \mathcal{L} : |F \cap \delta(L)| = 1\}$. Moreover, let $L \in \mathcal{L}^F \cup \{V\}$. Let $v \in L$ be the unique vertex such that $F[L]$ is an s-v-tour in $G[L]$. Then*

$$c\left(F_{L,v}\right) \;\leq\; \beta \cdot c\left(F[L]\right) - (\beta - 1) \cdot c\left(F[L] \cap \bigcup_{U \in \mathcal{L}^F} \delta(U)\right).$$

Proof. We prove the statement for sets $L \in \mathcal{L}^F \cup \{V\}$ in an order of increasing cardinality. So let $L \in \mathcal{L}^F \cup \{V\}$ and suppose Lemma 8.47 holds for every set $L' \in \mathcal{L}^F$ with $L' \subsetneq L$. If L is the minimal set in $\mathcal{L}^F \cup \{V\}$, then $F[L] \cap \bigcup_{U \in \mathcal{L}^F} \delta(U) = \emptyset$. Because $F[L]$ is an s-v-tour in $G[L]$, applying the β-approximation algorithm $\mathcal{B}$ to compute such a tour yields an s-v-tour of cost at most $\beta \cdot c(F[L])$. Hence, $c(F_{L,v}) \leq \beta \cdot c(F[L])$ as claimed.

Otherwise, if L is not minimal in $\mathcal{L}^F$, consider a maximal set $L' \subsetneq L$ with $L' \in \mathcal{L}^F$. Let $e = \{v', w'\}$ be the unique edge in $F \cap \delta(L')$ with $v' \in L'$. Then $e \notin \delta(L)$ because otherwise we have $e \notin F \cap \delta(L \setminus L') \subseteq \delta(L) \cup \delta(L') = \{e\}$ by definition of $\mathcal{L}^F$ and therefore $L \setminus L'$ is not visited by the s-t-tour F. Hence our dynamic programming algorithm considers the pair $(L', \{v', w'\})$.

Because L' is the maximal subset of L with $L' \in \mathcal{L}^F$, we have

$$F[L] \cap \textstyle\bigcup_{U \in \mathcal{L}^F} \delta(U) \;=\; \{\{v', w'\}\} \,\dot\cup\, \left(F[L'] \cap \textstyle\bigcup_{U \in \mathcal{L}^F} \delta(U)\right)$$

and

$$F[L] \;=\; F[L'] \,\dot\cup\, \{\{v', w'\}\} \,\dot\cup\, F[L \setminus L'].$$

Moreover, $F[L \setminus L']$ is a w'-v-tour in $G[L \setminus L']$. Thus, the s-v-tour resulting from the pair $(L', \{v', w'\})$ has cost at most

$$\begin{aligned}
&c(F_{L',v'}) + c(\{v', w'\}) + \beta \cdot c(F[L \setminus L']) \\
&\leq \beta \cdot c\left(F[L']\right) - (\beta - 1) \cdot c\Big(F[L'] \cap \textstyle\bigcup_{U \in \mathcal{L}^F} \delta(U)\Big) + c(\{v', w'\}) + \beta \cdot c(F[L \setminus L']) \\
&= \beta \cdot c\left(F[L]\right) - (\beta - 1) \cdot c\Big(F[L] \cap \textstyle\bigcup_{U \in \mathcal{L}^F} \delta(U)\Big).
\end{aligned}$$

□

The running time of the dynamic programming algorithm we described in this section is dominated by $O(n^4)$ calls to algorithm $\mathcal{B}$. Combining Lemma 8.47 for $L = V$ and Lemma 8.46 for an optimum s-t-tour F shows that $F_{V,t}$ has cost at most $(\beta - \frac{3}{2}\varepsilon \cdot (\beta - 1)) \cdot \mathrm{OPT} \leq (\beta - \frac{3}{2}\varepsilon \cdot (\alpha - 1)) \cdot \mathrm{OPT}$ if $\mathrm{dist}(s,t) \geq \left(\frac{1}{3} + \varepsilon\right) \cdot \mathrm{OPT}$. Otherwise, the output of algorithm $\mathcal{A}$ has cost at most $\alpha \cdot \mathrm{OPT}$. Taking the better of these two s-t-tours yields a $\max\{\alpha, \beta - \frac{3}{2}\varepsilon \cdot (\alpha - 1)\}$-approximation. This completes the proof of Lemma 8.45.

8.8.3 Proof of Theorem 8.42 and Theorem 8.44

We now show that an iterative application of Lemma 8.45 implies Theorem 8.42 and Theorem 8.44. To this end, we define a sequence $\mathcal{B}_i$ of approximation algorithms with approximation ratio β_i. We define $\mathcal{B}_0$ to be Christofides' algorithm and set $\beta_0 = \frac{5}{3}$. For $i \in \mathbb{N}$, the algorithm $\mathcal{B}_i$ results from $\mathcal{B}_{i-1}$ by applying Lemma 8.45 to the given polynomial-time algorithm $\mathcal{A}$ that computes an α-approximation if the distance of s and t is at most $(\frac{1}{3} + \varepsilon) \cdot \mathrm{OPT}$ and the algorithm $\mathcal{B} := \mathcal{B}_{i-1}$ with approximation ratio β_{i-1}. For $\beta_i := \max\{\alpha, \beta_{i-1} - \frac{3}{2}\varepsilon \cdot (\alpha - 1)\}$ we get by induction on i that for any fixed $i \in \mathbb{N}$ the algorithm $\mathcal{B}_i$ is a β_i-approximation algorithm. The running time of $\mathcal{B}_i$ is $O(n^{4i})$ times the running time of $\mathcal{B}_0$.

We claim that for

$$i_{\max} := \left\lceil \frac{2(\frac{5}{3} - \alpha)}{3\varepsilon \cdot (\alpha - 1)} \right\rceil$$

we have $\beta_{i_{\max}} = \alpha$. Proving this completes the proof of Theorem 8.42 and Theorem 8.44 because $i_{\max}$ is a constant.

By induction on i we have $\beta_i = \max\{\alpha, \beta_0 - i \cdot \frac{3}{2}\varepsilon \cdot (\alpha - 1)\}$ for all $i \in \mathbb{N}$. Therefore,

$$\begin{aligned}
\beta_{i_{\max}} &= \max\left\{\alpha,\ \beta_0 - \left\lceil \frac{2(\frac{5}{3} - \alpha)}{3\varepsilon \cdot (\alpha - 1)} \right\rceil \cdot \tfrac{3}{2}\varepsilon \cdot (\alpha - 1)\right\} \\
&\leq \max\left\{\alpha,\ \tfrac{5}{3} - \left(\tfrac{5}{3} - \alpha\right)\right\} \\
&= \alpha.
\end{aligned}$$

This shows $\beta_{i_{\max}} = \alpha$ and completes the proof.

Chapter 9

Reducing s-t-path TSP to TSP

In this chapter we give a black-box reduction from the s-t-path (graph) TSP to (graph) TSP. As a corollary, we obtain a $(1.4+\varepsilon)$-approximation algorithm for the s-t-path graph TSP for any fixed $\varepsilon > 0$. This chapter is based on [TVZ19], which is joint work with Jens Vygen and Rico Zenklusen.

9.1 Introduction

We have seen in Chapter 6 that Christofides' algorithm for the s-t-path TSP has an approximation ratio of only $\frac{5}{3}$, while for TSP it has approximation ratio $\frac{3}{2}$. Moreover, the integrality ratios of the classical LP relaxations are widely believed to differ between TSP and its path version. For the unit-weight special cases we know that this is the case: Sebő and Vygen [SV14] showed that the integrality ratio for graph TSP is at most 1.4, while for the s-t-path TSP it is at least $\frac{3}{2}$ (see Figure 6.2). The fact that the integrality ratios for graph TSP and s-t-path graph TSP are different can also be seen from the upper bound on the integrality ratio for the s-t-path graph TSP that we gave in the previous chapter (Lemma 8.39): our bound is about $\frac{3}{2}$ for instances where the distance of s and t is close to $\frac{1}{2} \cdot \mathrm{OPT}$, but better if the distance is smaller. In particular, for graph TSP the distance between s and t is zero and hence our upper bound is less than $\frac{3}{2}$.

In the previous chapter we also gave a 1.497-approximation algorithm for the s-t-path graph TSP, achieving an approximation ratio below the integrality ratio of the classical LP relaxations. Therefore, one might hope that s-t-path graph TSP is actually no harder than graph TSP although the integrality ratios differ.

This leads to the following general question regarding the relation between the approximability of s-t-path TSP and TSP, which we address in this chapter:

Is s-t-path (graph) TSP substantially harder to approximate than
its well-known special case (graph) TSP?

The answer is no. The main result of this chapter is to show in a constructive way that the s-t-path TSP can be approximated equally well as TSP (up to an arbitrarily small error), by presenting a black-box reduction that transforms approximation algorithms for TSP into ones for s-t-path TSP.

9.1.1 Our results

The main consequence of our reduction can be summarized as follows.

Theorem 9.1. *Let $\mathcal{A}$ be an α-approximation algorithm for TSP. Then, for any $\varepsilon > 0$, there is an $(\alpha + \varepsilon)$-approximation algorithm for s-t-path TSP that, for any instance (G, c, s, t), calls $\mathcal{A}$ a polynomial number of times on TSP instances defined on subgraphs of (G, c), and performs further operations taking polynomial time.*

The following two statements are immediate consequences of the above theorem.

Corollary 9.2. *Let $\varepsilon > 0$ and $\alpha > 1$. If there is an α-approximation algorithm for TSP, then there is an $(\alpha + \varepsilon)$-approximation algorithm for s-t-path TSP.*

Corollary 9.3. *Let $\varepsilon > 0$ and $\alpha > 1$. If there is an α-approximation algorithm for graph TSP, then there is an $(\alpha + \varepsilon)$-approximation algorithm for s-t-path graph TSP.*

The above statements create a strong link between the approximability of s-t-path TSP and TSP, as well as its graph versions. More precisely, Theorem 9.1 implies that such a link exists for any class of TSP instances that is closed under taking instances on subgraphs of the original instance (without changing the edge lengths). In particular, any potential future progress on the approximability of (graph) TSP will immediately carry over to s-t-path (graph) TSP.

Moreover, Corollary 9.3 allows us to make significant progress on the previously best approximation ratio of 1.497 for s-t-path graph TSP (Chapter 8, [TV18]), through a black-box reduction to the 1.4-approximation algorithm for graph TSP by Sebő and Vygen [SV14].

Corollary 9.4. *For any $\varepsilon > 0$, there is a $(1.4+\varepsilon)$-approximation algorithm for s-t-path graph TSP.*

Our reduction technique is quite versatile. In particular, it applies to a pretty general problem class (the Φ-tour problem with interfaces of bounded size; see Definition 9.6 and Theorem 9.23). This includes the T-tour problem for bounded $|T|$ (see Section 8.1.1 for a definition) and certain uncapacitated vehicle routing problems such as the one with a fixed number of depots studied in [XR15].

9.1.2 Organization of this chapter

After some brief preliminaries in Section 9.1.3, we provide an overview of our approach in Section 9.2. Here, we first focus on some key aspects of our approach, which is based on a new way to employ dynamic programming by using a well-chosen auxiliary problem, which we call Φ-TSP. Moreover, we break down the problem of finding a short solution to Φ-TSP into two cases. Combining the two cases, applying the same algorithm recursively, and using a constant-factor approximation algorithm for Φ-TSP on the final recursion level will imply our main reduction result, Theorem 9.1.

For one case, we show in Section 9.3 how to reduce the problem to TSP. For the other case, we show in Section 9.4 how to guess a constant fraction of an optimum solution via dynamic programming. The detailed proof of Theorem 9.1 is in Section 9.5. Finally, Section 9.6 contains a 4-approximation algorithm for Φ-TSP.

9.1.3 Preliminaries

In this chapter we only consider undirected graphs with nonnegative edge lengths and do not always state this explicitly. Also we assume that a weighted graph (V, E, c) has no loops or parallel edges. We often deal with multi-sets of edges. Although E does not contain parallel edges, when we write $F \subseteq E$, we mean a multi-set F that can contain several copies of the same edge.

We will work with the formulation of (s-t-path) TSP where we do not assume that the cost function c fulfills the triangle inequality, but allow the tour to visit vertices more than once. So a solution to (graph) TSP is a multi edge set F such that (V, F) is connected and $\text{odd}(F) = \emptyset$ and a solution to s-t-path (graph) TSP (with $s \neq t$) is a multi edge set F such that (V, F) is connected and $\text{odd}(F) = \{s, t\}$.

We say that an algorithm computes an α-approximate solution for a minimization problem if it computes a solution of cost at most $\alpha \cdot \text{OPT}(\mathcal{I})$ for every instance $\mathcal{I}$. In contrast to an α-approximation algorithm, we do not require the algorithm to have polynomial runtime.

In the interest of clarity and simplicity of the presentation, we did not try to optimize the running times of our procedures. Consequently, we often opt for weaker constants that are easier to obtain.

9.2 Overview of our approach

A key novelty of our approach is a new way to set up a dynamic program to successively strengthen a basic algorithm by combining it with a stronger algorithm for TSP. Every time we apply our dynamic program to obtain a stronger algorithm, we end up with a more difficult problem, slowly approaching problem settings for which it is very challenging to find strong approximation algorithms. However, as we show, by guessing a well-chosen set of edges through the dynamic program, we can limit the recursion depth by a constant, which allows us to stay in a regime where our approach runs efficiently.

9.2.1 Key challenges and high-level approach

Assume we are given an α-approximation algorithm $\mathcal{A}$ for TSP. Then finding a short s-t-tour using $\mathcal{A}$ as an oracle would be easy if the distance $\text{dist}(s, t)$ between the start s and the end t was short compared to OPT, i.e. the length of a shortest s-t-tour. Indeed, in this case the length of a shortest TSP tour and a shortest s-t-tour do not differ by much because any solution of one problem can be converted to a solution of the other one by adding a shortest s-t-path. Hence, one can simply compute an α-approximate TSP tour F and a shortest s-t-path P and return $F \dot{\cup} P$.

Consequently, a canonical plan would be to try to transform the s-t-path TSP instance to another one with small s-t-distance. It turns out that if the distance between s and t is very large, then such a reduction is indeed possible by using a technique based on dynamic programming. We have seen this in the previous chapter (in Section 8.8). There we showed that we can reduce to s-t-path TSP instances where the distance between s and t is at most $(\frac{1}{3} + \varepsilon) \cdot \text{OPT}$, for some arbitrarily small constant $\varepsilon > 0$.

Notice that if $\text{dist}(s, t) \leq \frac{1}{3} \cdot \text{OPT}$, it might happen that every cut in the chain $\mathcal{L}$ of s-t-cuts considered by the dynamic program contains more than one edge of an optimum

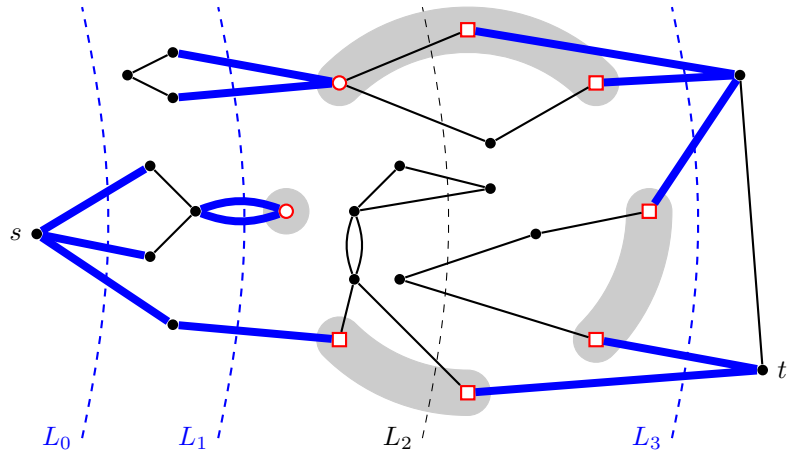

Figure 9.1: Illustration for guessing edges in cuts $\delta(L_i)$ that contain at most five F^*-edges. In this example, $\delta(L_0), \delta(L_1)$, and $\delta(L_3)$ are these cuts and the edges crossing them are highlighted as thick blue edges. We now define a sub-problem in $G[L_3 \setminus L_1]$. We call the endpoints of the thick blue edges *interface vertices* (red). The gray sets show the connectivity requirements on the interfaces vertices in this sub-problem: in a solution of the sub-problem, vertices in the same gray set must be in the same connected component. Interface vertices shown as squares must have odd degree; all other vertices must have even degree.

s-t-tour. Then no decomposition into smaller s-t-path TSP instances is possible. This is the reason why we reduced the start-to-end distance only to about a third of OPT.

If we could guess not only 1-cuts (containing only one edge of an optimum s-t-tour), but also cuts with a larger constant number of F^*-edges, say up to 5, then we could handle instances with an s-t-distance below $\frac{1}{3} \cdot \text{OPT}$. (This idea is inspired by a recent dynamic programming approach in [NZ19] in the context of chain-constrained spanning trees.) Our approach aims at realizing this high-level plan. However, this ostensibly simple algorithmic idea comes with several significant technical hurdles. Most importantly, if we guess more than one edge, the resulting sub-problems are not s-t-path TSP problems anymore. More precisely, if we guess 5 edges in each of two consecutive 5-cuts, then we have up to 10 *interface vertices*, i.e., endpoints of guessed edges. See Figure 9.1 for an example.

The optimum s-t-tour F^* is not necessarily connected inside the vertex set of a sub-problem but every connected component must contain at least one interface vertex. Moreover, F^* needs to connect some of the interface vertices to each other. This induces connectivity constraints for the sub-problem, shown as gray sets in Figure 9.1. They can also be guessed since the number of interface vertices is constant. Note, however, that we cannot guess the entire connected components, as there are exponentially many options.

Clearly, these sub-problems become significantly more difficult than the original s-t-path TSP problem. Moreover, if we try to apply such a procedure recursively, then the sub-problems can become more complex with each recursion step, because of an increase in the number of interface vertices per sub-problem. Another important issue in a recursive application to our more complex sub-problem is to identify good cuts

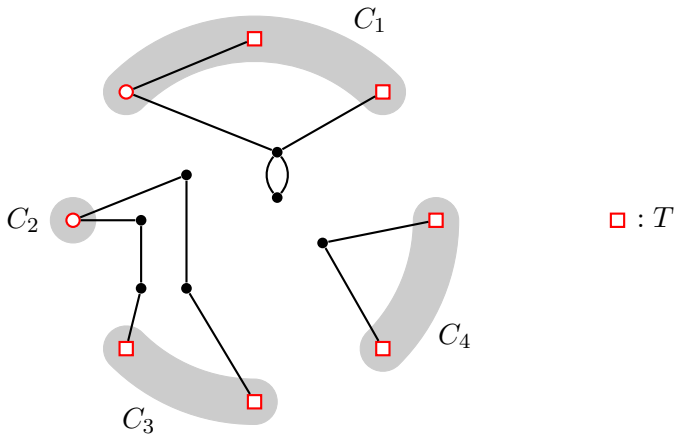

Figure 9.2: Example of an interface $\Phi = (I, T, \mathcal{C})$ and a Φ-tour. The partition $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$ of I is highlighted as gray sets. Hence, $I = \cup_{i=1}^{4} C_i$. Moreover, the vertices in T are drawn as red rectangles. This defines a Φ-TSP instance that results as a sub-problem in Figure 9.1. The shown edges F are a Φ-tour.

in which we should guess edges of F^*. Our cuts will result from the dual of a T-join problem. They will no longer form a chain. However, as we will see later, they will have a laminar structure that can be used to design a dynamic programming approach.

Moreover, it is not obvious how to reduce the problem to TSP in the case when we cannot guess edges by dynamic programming, and this will involve a careful guessing of further edges of F^*.

We will now describe our approach in detail. We start by defining the Φ-TSP problem, a new problem class around which our method is centered and which captures the type of sub-problems we face when guessing many edges.

9.2.2 Φ-TSP

As described above, when guessing edges, the endpoints of those edges play a special role in terms of how we have to connect things. We capture this through the notion of an interface. We define this notion for a general graph G below and will typically use it for subgraphs of the instance we are interested in.

Definition 9.5 (interface)**.** *An interface Φ of a graph $G = (V, E)$ is a triple $\Phi = (I, T, \mathcal{C})$ with*

(i) $T \subseteq I \subseteq V$, where $|T|$ is even, and

(ii) $\mathcal{C} \subseteq 2^I$ is a partition of I.

For an interface Φ of G, we denote by $(I_\Phi, T_\Phi, \mathcal{C}_\Phi)$ its corresponding triple and call $|I_\Phi|$ its size.

For a given interface, we are interested in finding what we call Φ-*tours*, which are defined as follows.

Definition 9.6 (Φ-tour)**.** *Let $G = (V, E)$ be a graph. Let $\Phi = (I, T, \mathcal{C})$ be an interface of G. A Φ-tour in G is a multi-set $F \subseteq E$ with*

(i) $T = \mathrm{odd}(F)$*, i.e.* F *is a* T*-join,*

(ii) $(V, F)/I$ *is connected, and*

(iii) for any $C \in \mathcal{C}$*, the vertices in* C *lie in the same connected component of* (V, F)*.*

Figure 9.2 exemplifies the notion of an interface Φ and a Φ-tour.

The problem we focus on in the following, which we call Φ-TSP, seeks to find a shortest Φ-tour.

Definition 9.7 (Φ-TSP)**.** *Given a weighted graph* $G = (V, E, c)$ *and an interface* Φ *of* G*, compute a shortest* Φ*-tour in* G *or decide that none exists. In short,*

$$\min \{c(F) : F \text{ is a } \Phi\text{-tour in } G\}. \tag{Φ-TSP}$$

Note that for any distinct $s, t \in V$, by choosing the interface $\Phi = (I, T, \mathcal{C})$ with $I = T = \{s, t\}$ and $\{C\} = \{\{s, t\}\}$, we have that Φ-tours correspond to solutions to s-t-path TSP. Analogously, for larger sets T, one captures the T-tour problem (see [SV14, CFG15, Seb13]). Another special case is the uncapacitated vehicle routing problem with a fixed number of depots, for which Xu and Rodrigues [XR15] gave a $\frac{3}{2}$-approximation algorithm. Here, I is the set of depots, $T = \emptyset$, and $\mathcal{C}$ is the partition into singletons.

Depending on the structure of the graph G and the interface Φ, it may be that no Φ-tour exists. We call an interface Φ of G *feasible* if G admits a Φ-tour. The existence of a Φ-tour admits the following easy characterization, which can be checked in linear time.

Lemma 9.8. *Let* $G = (V, E, c)$ *be a weighted graph. Let* $\Phi = (I, T, \mathcal{C})$ *be an interface of* G*. Then* G *admits a* Φ*-tour if and only if all of the following conditions hold.*

(i) Each connected component of G *contains an even number of vertices in* T*,*

(ii) G/I *is connected, and*

(iii) for every $C \in \mathcal{C}$*, the vertices in* C *lie in the same connected component of* G*.*

Proof. The three mentioned conditions are clearly necessary for G to admit a Φ-tour. Moreover, if they are satisfied then, due to (i), there exists a T-join $J \subseteq E$, and points (ii) and (iii) guarantee that $E \mathbin{\dot\cup} (E \setminus J)$ is a Φ-tour in G. □

It is crucial for our approach to start with a constant-factor approximation algorithm, which we will successively strengthen as discussed in the following.

A 7-approximation algorithm for Φ-TSP can be obtained easily as follows. Compute a minimum cost edge set F_1 satisfying (i) (T-join), a minimum cost edge set F_2 satisfying (ii) (spanning tree in G/I), and a 2-approximation F_3 of a minimum cost edge set satisfying (iii) (Steiner forest; see Section 2.9). Then the disjoint union $F_1 \mathbin{\dot\cup} F_2 \mathbin{\dot\cup} F_2 \mathbin{\dot\cup} F_3 \mathbin{\dot\cup} F_3$ is a 7-approximation.

With a little more care we can obtain a 4-approximation algorithm, using Jain's iterative rounding framework [Jai01]:

Theorem 9.9. *There is a* 4*-approximation algorithm for* Φ*-TSP.*

We defer the proof to Section 9.6. In the rest of this chapter, we will derive an $(\alpha+\varepsilon)$-approximation algorithm for Φ-TSP instances with bounded interface size, where α is the approximation ratio for TSP; see Theorem 9.23.

9.2.3 Iterative improvement of basic algorithm

For a TSP algorithm $\mathcal{A}$, we denote for every weighted graph G by $f_{\mathcal{A}}(G)$ the maximum runtime of algorithm $\mathcal{A}$ on any subgraph of G. Similarly, for a Φ-TSP algorithm $\mathcal{B}$, we denote for every weighted graph G and any $k \in \mathbb{R}_{\geq 0}$ by $f_{\mathcal{B}}(G,k)$ the maximum runtime of algorithm $\mathcal{B}$ on any instance (G',Φ), where G' is a subgraph of G and $|I_\Phi| \leq k$.

Our plan is to start with the 4-approximation algorithm for Φ-TSP guaranteed by Theorem 9.9, and successively improve it through a TSP algorithm with an approximation ratio α. The following Boosting Theorem is the main technical result towards this goal and quantifies the improvement in terms of approximation ratio that we are able to obtain in one improvement step.

Theorem 9.10 (Boosting Theorem)**.** *Let $\alpha, \beta > 1$. Suppose we are given:*

(a) an algorithm $\mathcal{A}$ that computes α-approximate solutions for TSP, and

(b) an algorithm $\mathcal{B}$ that computes β-approximate solutions for Φ-TSP.

Then there is an algorithm that, for any $\varepsilon \in (0,1]$, any weighted graph $G=(V,E,c)$, and any feasible interface $\Phi=(I,T,\mathcal{C})$ of G, returns a Φ-tour F in G of length

$$c(F) \leq \max\left\{(1+\varepsilon)\alpha,\ \beta - \frac{\varepsilon}{8}(\beta-1)\right\} \cdot \mathrm{OPT} \tag{9.1}$$

in time $|V|^{O\left(\frac{|I|}{\varepsilon}\right)} \cdot \left(f_{\mathcal{A}}(G) + f_{\mathcal{B}}\left(G, \frac{9|I|}{\varepsilon}\right)\right)$, where OPT *is the length of a shortest Φ-tour in G. In particular, the algorithm makes calls to $\mathcal{B}$ only on instances with interfaces of size bounded by $\frac{9|I|}{\varepsilon}$.*

To prove Theorem 9.1, we start with $\beta = 4$ (Theorem 9.9) and apply Theorem 9.10 repeatedly, but only a constant number of times. The approximation ratio β decreases until it reaches $(1+\varepsilon)\alpha$. All interfaces will have constant size. We defer the details to Section 9.5.

9.2.4 Proof outline of the Boosting Theorem (Theorem 9.10)

Theorem 9.10 is obtained by designing two algorithms to obtain a Φ-tour and then returning the better of the Φ-tours computed by these algorithms. Each of the following two theorems summarizes the guarantee we obtain with one of the two algorithms. After that, Algorithm 4, described below, combines these two sub-procedures to obtain an algorithm that implies Theorem 9.10.

The following theorem yields a short Φ-tour if the length of a minimum T_Φ-join is small.

Theorem 9.11. *Let $\alpha > 1$. Assume we are given an algorithm $\mathcal{A}$ that computes α-approximate solutions for TSP. Then, for any $\delta > 0$, any weighted graph $G=(V,E,c)$, and any feasible interface $\Phi=(I,T,\mathcal{C})$ of G, one can determine a Φ-tour F in G with*

$$c(F) \leq (1+\delta)\cdot\alpha\cdot\mathrm{OPT} + (\alpha+1)\cdot c(J)$$

in time $|V|^{O\left(\frac{|I|}{\delta}\right)} \cdot f_{\mathcal{A}}(G)$, where J is a shortest T-join in G and OPT *is the length of a shortest Φ-tour in G.*

We will give the proof in Section 9.3. The next theorem, proven in Section 9.4, states that we also obtain a short Φ-tour if the length of a minimum T-join is large.

Theorem 9.12. *Let $\beta > 1$. Assume we are given an algorithm $\mathcal{A}$ that computes α-approximate solutions for TSP. Then, for any $\delta > 0$, any weighted graph $G = (V, E, c)$, and any feasible interface $\Phi = (I, T, \mathcal{C})$ of G, one can determine a Φ-tour F in G with*

$$c(F) \leq \big(\beta + \delta \cdot (\beta - 1)\big) \cdot \mathrm{OPT} - (\beta - 1) \cdot c(J)$$

in time $|V|^{O\left(|I| + \frac{|T|}{\delta}\right)} \cdot f_{\mathcal{B}}\left(G, |I| + \frac{|T|}{\delta}\right)$, where J is a shortest T-join in G and OPT is the length of a shortest Φ-tour in G.

Algorithm 4: Approximation algorithm for Φ-TSP to prove Theorem 9.10

1. Run algorithm guaranteed by Theorem 9.11 with $\delta = \frac{\varepsilon}{2}$ to obtain Φ-tour F_1.
2. Run algorithm guaranteed by Theorem 9.12 with $\delta = \frac{\varepsilon}{8}$ to obtain Φ-tour F_2.
3. Return the shorter Φ-tour among F_1 and F_2.

Lemma 9.13. *Given a weighted graph G and a feasible interface $\Phi = (I, T, \mathcal{C})$ of G, Algorithm 4 returns a Φ-tour F in G with the guarantees stated in Theorem 9.10.*

Proof. The running time guarantee stated in Theorem 9.10 immediately follows from Theorem 9.11 and Theorem 9.12, using $|I| + \frac{8|T|}{\varepsilon} \leq \frac{9|I|}{\varepsilon}$. Let $F \in \{F_1, F_2\}$ be the Φ-tour returned by Algorithm 4. To show that F fulfills the approximation guarantee stated in (9.1), we distinguish two cases.

If $c(J) \leq \frac{\varepsilon}{4} \cdot \mathrm{OPT}$, then the solution F_1 will be short enough:

$$c(F) \leq c(F_1) \leq \left(1 + \frac{\varepsilon}{2}\right) \alpha \cdot \mathrm{OPT} + (\alpha + 1)\frac{\varepsilon}{4} \cdot \mathrm{OPT} \leq (1 + \varepsilon) \cdot \alpha \cdot \mathrm{OPT},$$

where we used $\frac{(\alpha+1)}{2} \leq \alpha$ for the last inequality, which holds because $\alpha \geq 1$.

If $c(J) \geq \frac{\varepsilon}{4} \cdot \mathrm{OPT}$, then the Φ-tour F_2 will be short enough:

$$c(F) \leq c(F_2) \leq \left(\beta + \frac{\varepsilon}{8}(\beta - 1)\right) \cdot \mathrm{OPT} - (\beta - 1) \cdot \frac{\varepsilon}{4} \cdot \mathrm{OPT} = \left(\beta - \frac{\varepsilon}{8}(\beta - 1)\right) \cdot \mathrm{OPT},$$

thus completing the proof of Lemma 9.13. □

For the proof of Theorem 9.10, it remains to show Theorem 9.11 and Theorem 9.12.

9.3 Finding a short Φ-tour if there is a short T-join

In this section we prove Theorem 9.11, i.e., how to get a short Φ-tour if the shortest T-join has small length compared to OPT.

We start by analyzing a simple algorithm for computing a Φ-tour. However, this simple algorithm will not be sufficient to prove Theorem 9.11. Thus in a second step, we will refine the algorithm to obtain the desired bound.

Algorithm 5: A simple Φ-TSP algorithm

Input: a weighted graph G, an interface $\Phi = (I, T, \mathcal{C})$ of G, and a T-join J in G.
Output: an edge set F.

1. In each connected component of G apply $\mathcal{A}$ to get an α-approximate TSP-tour. Let Q be the union of these tours.
2. Return $F = Q \dot\cup J$.

Notice that Algorithm 5 always returns an edge set F, even if the input is infeasible. We therefore show first that Algorithm 5 does return a Φ-tour whenever it is run with a feasible input.

Lemma 9.14. *The set F returned by Algorithm 5 is a Φ-tour if and only if the input is feasible, i.e. G admits a Φ-tour.*

Proof. Assume that G admits a Φ-tour, which implies by Lemma 9.8 that the three properties (i), (ii), and (iii) listed in Lemma 9.8 are fulfilled. Because the set Q computed in Algorithm 5 consists of TSP tours in each connected component of G, the vertex sets of the connected components of (V, Q) and G are the same. Because G fulfills (ii) and (iii), this implies that also (V, Q) and $(V, Q \dot\cup J)$ fulfill these two conditions. Finally, $\operatorname{odd}(Q \dot\cup J) = \operatorname{odd}(J) = T$, because $\operatorname{odd}(Q) = \emptyset$ and J is a T-join, which shows that $Q \dot\cup J$ is indeed a Φ-tour. □

We now analyze the length of the Φ-tour returned by Algorithm 5.

Lemma 9.15. *Assume we are given an algorithm $\mathcal{A}$ that computes α-approximate solutions for TSP. Let $G = (V, E, c)$ be a weighted graph, $\Phi = (I, T, \mathcal{C})$ a feasible interface of G, and J a T-join in G. Then, Algorithm 5 computes a Φ-tour F in G with*

$$c(F) \leq \alpha \cdot \mathrm{OPT} + (\alpha + 1) \cdot c(J) + 2\alpha \cdot |I| \cdot \max\left\{c(e) : e \in E \setminus (F^* \cup J)\right\},$$

in time $O(|V| \cdot f_{\mathcal{A}}(G))$, where F^* is a shortest Φ-tour. Here $\max \emptyset := 0$.

Proof. First, observe that the running time is indeed as claimed, because the bottleneck of the algorithm is calling $\mathcal{A}$ for each connected component of G; moreover, the connected components can be found in linear time and there are at most $|V|$ many of them.

To bound $c(Q)$, we transform a shortest Φ-tour F^* into a union of TSP solutions, one for each connected component of G. Let $S \subseteq E$ be a minimal edge set such that the vertex sets of the connected components of $(V, F^* \cup J \cup S)$ and G are the same. Observe that the multi-set $F^* \dot\cup J \dot\cup S \dot\cup S$ is a union of TSP solutions, one for each connected component of G. Because the set Q determined in Algorithm 5 was obtained through $\mathcal{A}$, which computes α-approximate solutions, we have

$$c(Q) \leq \alpha \cdot \big(\mathrm{OPT} + c(J) + 2c(S)\big),$$

and, hence, the solution $F = Q \dot\cup J$ returned by the algorithm satisfies

$$c(F) = c(Q) + c(J) \leq \alpha \cdot \mathrm{OPT} + (\alpha + 1) \cdot c(J) + 2\alpha \cdot c(S). \tag{9.2}$$

Moreover, because F^* is a Φ-tour, we have that $(V, F^*)/I$ must be connected, which implies that (V, F^*) has at most $|I|$ connected components, and thus

$$|S| \leq |I| - 1.$$

Together with (9.2), this leads to the desired guarantee:

$$c(F) \leq \alpha \cdot \mathrm{OPT} + (\alpha + 1) \cdot c(J) + 2\alpha \cdot |I| \cdot \max\left\{c(e) : e \in E \setminus (F^* \cup J)\right\},$$

where the inequality follows from $S \subseteq E \setminus (F^* \cup J)$, which holds because the edges in S connect different connected components of $(V, F^* \cup J)$. □

We now explain how to refine Algorithm 5 by a guessing step to obtain the guarantees claimed in Theorem 9.11. If all edges that are not contained in $F^* \cup J$ have length at most $\frac{\delta \cdot \mathrm{OPT}}{2 \cdot |I|}$, Lemma 9.15 already implies the desired bound. To obtain this property, we delete all edges from G that are *heavy*, i.e. have length at least $\frac{\delta \cdot \mathrm{OPT}}{2 \cdot |I|}$, and are not contained in $F^* \cup J$. We guess this set of edges to delete as follows. First we guess the set H of heavy edges, which can be done in polynomial time by guessing a minimum length edge in H. Then we guess the set $H^* = F^* \cap H$ of heavy edges contained in the optimum Φ-tour F^*. Algorithm 6 formalizes this procedure and, as we show next, indeed implies Theorem 9.11.

Algorithm 6: Φ-TSP algorithm to prove Theorem 9.11

Compute a shortest T-join J in G.
For $f \in E$ define $H_f \coloneqq \{e \in E : c(e) \geq c(f)\}$.
for *every edge set* $H \in \{H_f : f \in E\} \cup \{\emptyset\}$ **do**
  for *every set* $H^* \subseteq H$ *with* $|H^*| \leq \frac{2|I|}{\delta}$ **do**
    Set $D \coloneqq H \setminus (H^* \cup J)$.
    Apply Algorithm 5 to the graph $(V, E \setminus D)$ to obtain a multi-set F_D of edges, which is a Φ-tour in G if the input is feasible.
Among all computed Φ-tours F_D, return a shortest one.

Proof of Theorem 9.11. We start by observing that the running time of Algorithm 6 is indeed bounded by $|V|^{O\left(\frac{|I|}{\delta}\right)} \cdot f_{\mathcal{A}}(G)$. There are at most $|V|^2$ possible edges f that are being considered in the outer for-loop. For each of them, there are $|V|^{O\left(\frac{|I|}{\delta}\right)}$ possible sets H^* considered in the inner for-loop. Thus, there are at most $|V|^{O\left(\frac{|I|}{\delta}\right)}$ calls to Algorithm 5. Finally, all other operations can be done in time $|V|^{O(1)}$.

We now show that Algorithm 6 returns a Φ-tour with the guarantees claimed by Theorem 9.11. Let F^* be a shortest Φ-tour and let $H \coloneqq \{e \in E : c(e) \geq \frac{\delta \cdot \mathrm{OPT}}{2|I|}\}$ be the set of heavy edges. Then in some iteration of the outer for-loop we consider the set H. Because

$$\mathrm{OPT} \geq c(H \cap F^*) \geq |H \cap F^*| \cdot \frac{\delta \cdot \mathrm{OPT}}{2 \cdot |I|},$$

we have $|H \cap F^*| \leq \frac{2|I|}{\delta}$, and thus, we consider the set $H^* \coloneqq H \cap F^*$ in some iteration of the inner for-loop. As $D = H \setminus (H^* \cup J)$ does not contain any edge of F^*, the Φ-tour F^* is a feasible solution of the instance to which we apply Algorithm 5. Moreover, the

set D contains all heavy edges not contained in $F^* \cup J$ and hence by Lemma 9.15, we obtain

$$\begin{aligned} c(F_D) &\leq \alpha \cdot \mathrm{OPT} + (\alpha+1) \cdot c(J) + 2\alpha \cdot |I| \cdot \max\left\{c(e) : e \in (E \setminus D) \setminus (F^* \cup J)\right\} \\ &\leq \alpha \cdot \mathrm{OPT} + (\alpha+1) \cdot c(J) + 2\alpha \cdot |I| \cdot \frac{\delta \cdot \mathrm{OPT}}{2 \cdot |I|} \\ &= (1+\delta) \cdot \alpha \cdot \mathrm{OPT} + (\alpha+1) \cdot c(J). \end{aligned}$$

□

9.4 Iterative improvement via dynamic programming

In this section, we show how to prove Theorem 9.12, i.e., how to obtain a short Φ-tour if the length of a shortest T-join is large. Here, our goal is to use dynamic programming to "guess" a significant portion, in terms of total length, of edges used in a fixed optimum Φ-tour F^*. Very recently, dynamic programming has become a strong tool in the context of s-t-path TSP, Chain-Constrained Spanning Trees, and related problems [TV19a, TV18, Zen19, NZ19], leading to the currently best known approximation ratios for these settings. The dynamic programming idea we employ combines and extends elements used in these prior dynamic programming techniques.

What we aim to achieve with dynamic programming in the context of Φ-TSP, for some interface $\Phi = (I, T, \mathcal{C})$ of G, is the following. We can fix an arbitrary laminar family $\mathcal{L}$ of subsets of V. Our goal is to guess what edges of optimum Φ-tour F^* are crossing the cuts in $\mathcal{L}$. Clearly, if $F^* \cap \delta(L)$ contains many edges for some $L \in \mathcal{L}$, it seems computationally elusive to guess them. This is the reason why we fix some constant k and only guess F^*-edges in cuts $\delta(L)$ for $L \in \mathcal{L}$ if $|F^* \cap \delta(L)| \leq k$. We denote the sets inducing these cuts by $\mathcal{L}(F^*, k) \subseteq \mathcal{L}$ and the F^*-edges in these cuts by $F^*(\mathcal{L}, k) \subseteq F^*$. Formally, for any edge set $R \subseteq E$, we define

$$\begin{aligned} \mathcal{L}(R,k) &\coloneqq \left\{L \in \mathcal{L} : |R \cap \delta(L)| \leq k\right\}, \text{ and} \\ R(\mathcal{L},k) &\coloneqq \bigcup_{L \in \mathcal{L}(R,k)} (\delta(L) \cap R). \end{aligned}$$

As we discuss in more detail later, guessing the edges $F^*(\mathcal{L}, k)$ can be achieved through a dynamic program that guesses the F^*-edges in the different cuts defined by $\mathcal{L}$ step by step, from smaller to larger sets in $\mathcal{L}$. However, the running time of the propagation step of the dynamic program depends on the number of disjoint sets in $\mathcal{L}$ that can be contained in some larger set $L \in \mathcal{L}$. We capture this dependency through the *width* $\mathrm{width}(\mathcal{L})$ of the laminar family $\mathcal{L}$.

Definition 9.16 (width of a laminar family)**.** *The* width $\mathrm{width}(\mathcal{L})$ *of a laminar family* $\mathcal{L}$ *is the number of minimal sets contained in the family.*

Observe that the number of minimal sets of a laminar family bounds the size of any subfamily of disjoint sets.

The following theorem formalizes what we can achieve through our dynamic program, which we present later in detail. Notice that for the algorithm to be efficient, we need $\mathcal{L}$ to have width bounded by a constant.

Theorem 9.17. *Let $\beta > 1$. Assume there is an algorithm $\mathcal{A}$ that computes α-approximate solutions for TSP. Then there is an algorithm that computes for any feasible interface $\Phi = (I, T, \mathcal{C})$ of a weighted graph $G = (V, E, c)$, any $k \in \mathbb{Z}_{\geq 0}$, and any laminar family $\mathcal{L}$ over V, a Φ-tour F with*

$$c(F) \leq \min\left\{\beta \cdot c(R) - (\beta - 1) \cdot c(R(\mathcal{L}, k)) : R \text{ is a } \Phi\text{-tour}\right\} \tag{9.3}$$

in time $|V|^{O(|I|+k\cdot\mathrm{width}(\mathcal{L}))} \cdot f_{\mathcal{B}}\left(G, |I| + k \cdot (\mathrm{width}(\mathcal{L}) + 1)\right)$. In particular, the algorithm calls $\mathcal{B}$ only on instances with interfaces of size bounded by $|I| + k \cdot (\mathrm{width}(\mathcal{L}) + 1)$.

Note that the guarantee stated in (9.3) for $R = F^*$ indeed reflects the guessing of the edges in $F^*(\mathcal{L}, k)$. More precisely, by replacing R by F^* in (9.3), we obtain a Φ-tour F with an upper bound on its length $c(F)$ that decomposes into two terms:

(i) a term $c(F^*(\mathcal{L}, k))$, i.e., each edge $e \in F^*(\mathcal{L}, k)$ contributes its length $c(e)$, and

(ii) a term $\beta \cdot c(F^* \setminus F^*(\mathcal{L}, k))$, where the length of each other edge in F^* gets inflated by the approximation factor β of the algorithm $\mathcal{B}$.

9.4.1 Finding a suitable laminar family

To make significant progress through Theorem 9.17, we need to find a laminar family $\mathcal{L}$ over V such that $c(F^*(\mathcal{L}, k))$ is large. Let J be a shortest T-join. If $c(J)$ is large, then we will construct a family $\mathcal{L}$ with the property that even for any T-join R, the length $c(R(\mathcal{L}, k))$ is large. Notice that this implies what we want because the Φ-tour F^* is a T-join.

This statement is formalized in Lemma 9.19, which can be derived from the dual of the natural linear program to find a shortest T-join. More precisely, this dual corresponds to a fractional T-cut packing problem, which is well-known to admit solutions with laminar support. This is the laminar family we will be using. The lemma below summarizes the statement we need. It is well-known and has been proven in different contexts (see, e.g. [Seb97] and references therein). For completeness, we provide a self-contained proof below.

Lemma 9.18. *Let $G = (V, E, c)$ be a weighted graph and let $T \subseteq V$ such that G contains a T-join. Moreover, let $t \in T$ and let J be a shortest T-join. Then there is a polynomial-time algorithm that computes a laminar family $\mathcal{L}$ over $V \setminus \{t\}$ and values $y \in \mathbb{R}^{\mathcal{L}}_{>0}$ such that*

$$\sum_{\substack{L\in\mathcal{L}:\\ e\in\delta(L)}} y_L \leq c(e) \qquad \forall e \in E, \tag{9.4}$$

$$\sum_{L\in\mathcal{L}} y_L = c(J), \text{ and} \tag{9.5}$$

$$|L \cap T| \text{ is odd} \qquad \forall L \in \mathcal{L}. \tag{9.6}$$

Proof. We start with a classical linear description to find a minimum length T-join, based on the dominant of the T-join polytope. To this end, let $\mathcal{F} = \{Q \subseteq V \setminus \{t\} :$

$|Q \cap T|$ is odd}; these vertex sets induce all T-cuts. Then, the following linear program computes the value $c(J)$ of a shortest T-join.

$$\begin{aligned} \min\ & c(x) \\ \text{s.t.}\quad & x(\delta(Q)) \geq 1 \quad \forall\, Q \in \mathcal{F} \\ & x_e \geq 0 \quad \forall\, e \in E \end{aligned} \tag{9.7}$$

Its dual problem, which is a fractional T-cut packing problem, is given below.

$$\begin{aligned} \max\ & \sum_{F \in \mathcal{F}} y_F \\ \text{s.t.}\quad & \sum_{\substack{Q \in \mathcal{F}: \\ e \in \delta(Q)}} y_Q \leq c(e) \quad \forall\, e \in E \\ & y_F \geq 0 \qquad \forall\, F \in \mathcal{F} \end{aligned} \tag{9.8}$$

If $y \in \mathbb{R}^{\mathcal{F}}_{\geq 0}$ is an optimum dual solution with laminar support $\mathcal{L}$, then y and $\mathcal{L}$ have the desired properties. Here (9.5) follows from strong duality (Theorem 2.7) and (9.6) follows from $L \in \mathcal{F}$.

We can solve (9.7) in polynomial time through the ellipsoid method. To find an optimal dual basis, one can delete all variables from (9.8) that do not correspond to constraints encountered by the ellipsoid algorithm when solving (9.7). The restricted dual LP has polynomial size and can therefore be solved in polynomial time.

Finally, this solution can be transformed into a laminar one by uncrossing: if $y_A > 0$ and $y_B > 0$ for $A, B \in \mathcal{F}$ with $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$ and $A \cap B \neq \emptyset$, then either $A \cap B$ and $A \cup B$ belong to $\mathcal{F}$ or $A \setminus B$ and $B \setminus A$ belong to $\mathcal{F}$; we can increase the dual variables on these two sets by $\min\{y_A, y_B\}$ and decrease the dual variables y_A and y_B by the same amount, maintaining a feasible dual solution. Karzanov [Kar96] showed how to obtain a laminar family by a sequence of such uncrossing steps in polynomial time. □

We now show that the family $\mathcal{L}$ from Lemma 9.18 has the desired properties.

Lemma 9.19. *Let $G = (V, E, c)$ be a weighted graph. Moreover, let $T \subseteq V$ such that G admits a T-join. Then, there is a polynomial-time algorithm that computes a laminar family $\mathcal{L}$ over V with* $\operatorname{width}(\mathcal{L}) \leq \max\{0, |T| - 1\}$ *such that for any T-join $R \subseteq E$, and any $k \in \mathbb{Z}_{\geq 0}$, we have*

$$c(R(\mathcal{L}, k)) \geq c(J) - \frac{1}{k+1} \cdot c(R),$$

where J is a shortest T-join in G.

Proof. If $T = \emptyset$, we can simply set $\mathcal{L} = \emptyset$ because $c(J) = 0$. Otherwise, we compute $\mathcal{L}$ and y as in Lemma 9.18 and show that $\mathcal{L}$ has the desired properties. Since every set in $\mathcal{L}$ must contain an element of $T \setminus \{t\}$, we have $\operatorname{width}(\mathcal{L}) \leq |T| - 1$.

Let now $R \subseteq E$ be a T-join, and let $k \in \mathbb{Z}_{\geq 0}$. Since R is a T-join, it has a non-empty intersection with every cut $\delta(L)$ with $L \in \mathcal{L}$ because of (9.6). Hence, by (9.4),

$$\begin{aligned} c(R(\mathcal{L},k)) &= \sum_{e \in R(\mathcal{L},k)} c(e) \\ &\geq \sum_{e \in R(\mathcal{L},k)} \sum_{\substack{L \in \mathcal{L}: \\ e \in \delta(L)}} y_L \\ &= \sum_{L \in \mathcal{L}} |R(\mathcal{L},k) \cap \delta(L)| \cdot y_L \\ &\geq \sum_{L \in \mathcal{L}(R,k)} y_L. \end{aligned} \tag{9.9}$$

Again using (9.4), we moreover obtain

$$c(R) \;\geq\; \sum_{e \in R} \sum_{\substack{L \in \mathcal{L}: \\ e \in \delta(L)}} y_L \;=\; \sum_{L \in \mathcal{L}} |R \cap \delta(L)| \cdot y_L \;\geq\; \sum_{L \in \mathcal{L} \setminus \mathcal{L}(R,k)} (k+1) \cdot y_L. \tag{9.10}$$

Combining (9.5), (9.9), and (9.10), we obtain

$$c(J) \;=\; \sum_{L \in \mathcal{L}} y_L \;=\; \sum_{L \in \mathcal{L}(R,k)} y_L + \sum_{L \in \mathcal{L} \setminus \mathcal{L}(R,k)} y_L \;\leq\; c(R(\mathcal{L},k)) + \frac{1}{k+1} \cdot c(R),$$

as desired. □

Finally, Theorem 9.12 is a direct consequence of Theorem 9.17 and Lemma 9.19.

Proof of Theorem 9.12. If $T = \emptyset$, we simply call the given algorithm $\mathcal{B}$ that computes a β-approximate solution. Otherwise, let $k = \lfloor \frac{1}{\delta} \rfloor$. We apply Lemma 9.19 to obtain in polynomial time a laminar family $\mathcal{L}$ over V such that

- $c(R(\mathcal{L},k)) \;\geq\; c(J) - \dfrac{1}{k+1} \cdot c(R) \;\geq\; c(J) - \delta \cdot c(R) \quad \forall\, T$-join $R \subseteq E$, and
- $\text{width}(\mathcal{L}) \;\leq\; \max\{0, |T| - 1\} \;=\; |T| - 1$, where the equality follows from the assumption $T \neq \emptyset$.

Because a shortest Φ-tour F^* is a T-join, we have

$$c(F^*(\mathcal{L},k)) \;\geq\; c(J) - \delta \cdot c(F^*) \;=\; c(J) - \delta \cdot \text{OPT},$$

which, together with Theorem 9.17 implies the desired results, i.e., that one can find a Φ-tour F in G with

$$\begin{aligned} c(F) &\leq \beta \cdot \text{OPT} - (\beta - 1) \cdot c(F^*(\mathcal{L},k)) \\ &\leq \beta \cdot \text{OPT} - (\beta - 1)\,(c(J) - \delta \cdot \text{OPT}) \\ &= (\beta + \delta \cdot (\beta - 1)) \cdot \text{OPT} - (\beta - 1) \cdot c(J), \end{aligned}$$

in time

$$|V|^{O\left(|I| + \frac{\text{width}(\mathcal{L})}{\delta}\right)} \cdot f_{\mathcal{B}}\left(G, |I| + \frac{\text{width}(\mathcal{L}) + 1}{\delta}\right) \leq |V|^{O\left(|I| + \frac{|T|}{\delta}\right)} \cdot f_{\mathcal{B}}\left(G, |I| + \frac{|T|}{\delta}\right). \quad \square$$

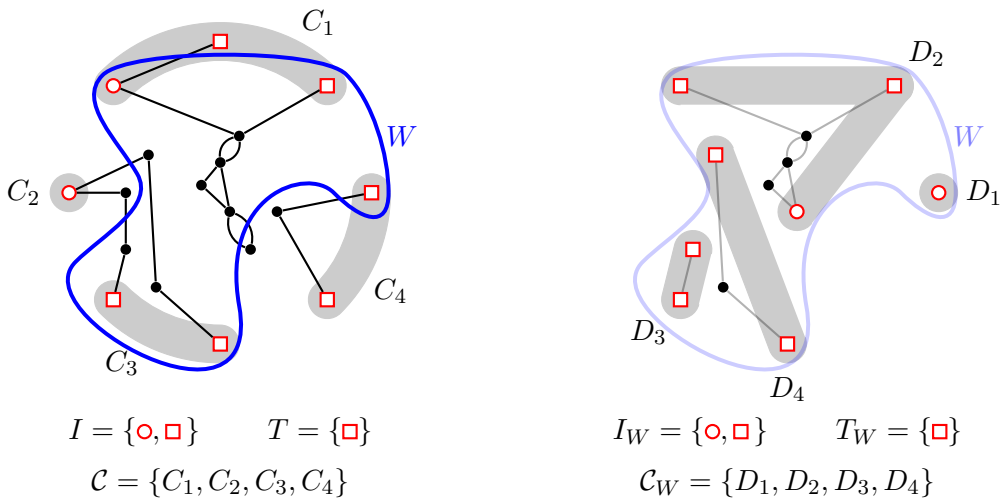

Figure 9.3: On the left-hand side, an interface $\Phi = (I, T, \mathcal{C})$ on a graph $G = (V, E)$ is shown together with a Φ-tour $F \subseteq E$ (the black edges) and a set $W \subseteq V$. The right-hand side figure depicts the interface $\Phi_W = (I_W, T_W, \mathcal{C}_W)$ induced by (F, Φ) on W.

It remains to derive Theorem 9.17, which, as mentioned, we show through a dynamic programming approach.

9.4.2 Combining partial solutions

In the analysis of our dynamic programming algorithm we use the following notion of an *induced interface*, which allows us to analyze the algorithm with respect to interfaces coming from a shortest Φ-tour.

Definition 9.20 (induced interface)**.** *Let $G = (V, E, c)$ be a weighted graph. Let $\Phi = (I, T, \mathcal{C})$ be an interface of G, and let F be a Φ-tour in G. For $W \subseteq V$, the interface $\Phi_W = (I_W, T_W, \mathcal{C}_W)$ induced by (F, Φ) on W is defined by*

(i) $I_W = (I \cap W) \cup U$, where U is the set of vertices in W that are connected by an edge of F to a vertex in $V \setminus W$,

(ii) $T_W = \operatorname{odd}(F[W])$, and

(iii) $\mathcal{C}_W \subseteq 2^{I_W}$ contains, for each connected component of $(W, F[W])$, a set including all vertices of I_W contained in that connected component.

See Figure 9.3 for an example of an induced interface. Moreover, also Figure 9.1, which we used as an illustrative example in the introduction to showcase the guessing of multiple edges per cut, highlights an induced interface with $W = L_3 \setminus L_1$, which is induced by an s-t-tour. We remark that the interface induced by (F, Φ) depends only on F and I, not on T or $\mathcal{C}$.

The following lemma shows some basic properties of induced interfaces.

Lemma 9.21. *Let $G = (V, E, c)$ be a weighted graph and $\Phi = (I, T, \mathcal{C})$ an interface of G. Let F be a Φ-tour in G and $W \subseteq V$. Let Φ_W be the interface induced by (F, Φ) on W. Then*

(i) Φ_W *is an interface of* $G[W]$*,*

(ii) $F[W]$ *is a* Φ_W*-tour in* $G[W]$*, and*

(iii) for every $W' \subseteq W$*, the interface induced by* $(F[W], \Phi_W)$ *on* W' *equals the interface induced by* (F, Φ) *on* W'*.*

Proof. Let $\Phi_W = (I_W, T_W, \mathcal{C}_W)$. As in Definition 9.20 (i), let U be the set of vertices in W that are connected by an edge of F to a vertex in $V \setminus W$.

To prove (i), we have to observe that $T_W \subseteq I_W$. (Notice that we clearly have that $|T_W|$ is even because $T_W = \operatorname{odd}(F[W])$.) Let $u \in T_W$. If F contains an edge connecting u with $V \setminus W$, then $u \in U$ and hence $u \in I_W$. Otherwise we have $(\delta(u) \cap F) \subseteq E[W]$ and hence $u \in T_W = \operatorname{odd}(F[W])$ implies $u \in \operatorname{odd}(F)$. Since F is a Φ-tour, we conclude $u \in T \subseteq I$. Moreover, $u \in T_W = \operatorname{odd}(F[W]) \subseteq W$, so $u \in I \cap W \subseteq I_W$.

To prove (ii), we have to show that $(W, F[W])/I_W$ is connected (the other two conditions of Definition 9.6 trivially hold). Suppose not. Then there is a set $W' \subseteq W \setminus I_W$ with $W' \neq W$ and $F[W] \cap \delta(W') = \emptyset$. This implies, together with $I_W = (I \cap W) \cup U$—which holds by definition of I_W—that $W' \subseteq V \setminus I$ with $W' \neq V$ and $F \cap \delta(W') = F[W] \cap \delta(W') = \emptyset$. This contradicts the fact that $(V, F)/I$ is connected, which has to hold because F is a Φ-tour.

To show (iii), let $(I_1, T_1, \mathcal{C}_1)$ be the interface induced by (F, Φ) on W' and let $(I_2, T_2, \mathcal{C}_2)$ be the interface induced by $(F[W], \Phi_W)$ on W'. Let U_1 be the set of vertices in W' that are connected by an edge of F to a vertex in $V \setminus W'$. Let U_2 be the set of vertices in W' that are connected by an edge of $F[W]$ to a vertex in $W \setminus W'$. Then $U_1 = (U \cap W') \cup U_2$. Therefore,

$$\begin{aligned} I_1 &= (I \cap W') \cup U_1 \\ &= (I \cap W') \cup (U \cap W') \cup U_2 \\ &= \big(((I \cap W) \cup U) \cap W'\big) \cup U_2 \\ &= (I_W \cap W') \cup U_2 \\ &= I_2. \end{aligned}$$

Finally, because $(F[W])[W'] = F[W']$, which follows from $W' \subseteq W$, we have

$$T_2 = \operatorname{odd}((F[W])[W']) = \operatorname{odd}(F[W']) = T_1,$$

and also $\mathcal{C}_1 = \mathcal{C}_2$, because these partitions of $I_1 = I_2$ are both defined with respect to the connected components of $(W', F[W'])$, because $(W', (F[W])[W']) = (W', F[W'])$. □

Notice that given an interface $\Phi = (I, T, \mathcal{C})$ on a graph $G = (V, E)$ and a Φ-tour $F \subseteq E$, then the interface Φ_V induced by (F, Φ) on V is not necessarily identical to Φ. More precisely, $\Phi_V = (I_V, T_V, \mathcal{C}_V)$ always fulfills $I_V = I$ and $T_V = T$. However, F may connect different parts of the partition $\mathcal{C}$, which, in the interface Φ_V, will then only appear as one set in $\mathcal{C}_V$. See the left-hand side illustration in Figure 9.3 for such an example where the highlighted Φ-tour would induce an interface $\Phi_V \neq \Phi$ on V because $C_2 \cup C_3$ is a single set in $\mathcal{C}_V$.

In our dynamic program we will combine solutions for different subgraphs with induced interfaces. The following lemma shows sufficient conditions under which this works out.

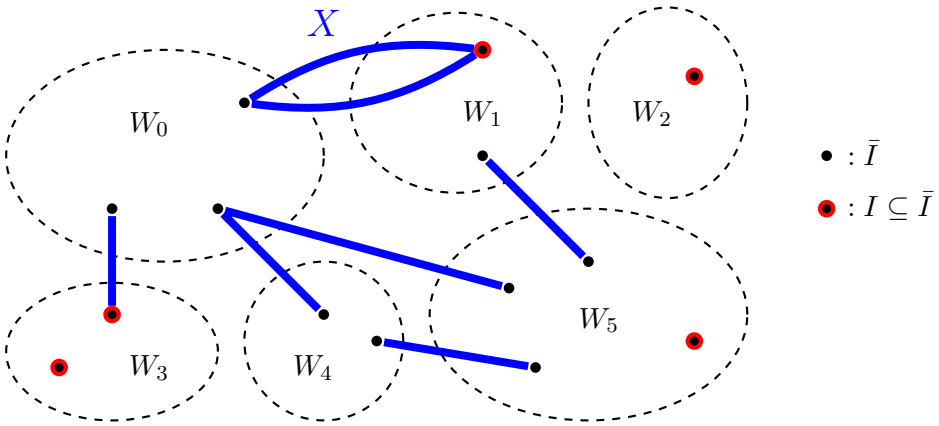

Figure 9.4: The dashed ellipsoids show the partition of V into $W_0, \dots, W_p$. The thick blue edges are the edges in X. Only the vertices in $\bar{I}$ are shown here, where the vertices with a red boundary are those contained in I.

Lemma 9.22. *Let $G = (V, E, c)$ be a weighted graph. Let $\Phi = (I, T, \mathcal{C})$ be an interface of G and let F be a Φ-tour in G. Let $W_0, \dots, W_p$ be a partition of V. For $i \in \{0, \dots, p\}$, let $\Phi_i = (I_i, T_i, \mathcal{C}_i)$ be the interface induced by (F, Φ) on W_i, and let F_i be a Φ_i-tour in $G[W_i]$. Then*

$$F' \coloneqq X \dot\cup \left(\bigcup_{i=0}^{p} F_i \right)$$

is a Φ-tour in G, where $X \coloneqq F \cap \bigcup_{i=0}^{p} \delta(W_i)$.

Proof. We first show point (i) of Definition 9.6, i.e. $\operatorname{odd}(F') = T$. For $i \in \{0, \dots, p\}$, we have $\operatorname{odd}(F_i) = T_i = \operatorname{odd}(F[W_i])$ since F_i is a Φ_i-tour. Thus

$$\begin{aligned} \operatorname{odd}(F') &= \operatorname{odd}(X) \mathbin{\triangle} \operatorname{odd}(F_0) \mathbin{\triangle} \cdots \mathbin{\triangle} \operatorname{odd}(F_p) \\ &= \operatorname{odd}(X) \mathbin{\triangle} \operatorname{odd}(F[W_0]) \mathbin{\triangle} \cdots \mathbin{\triangle} \operatorname{odd}(F[W_p]) \\ &= \operatorname{odd}(F) \\ &= T, \end{aligned}$$

where $\triangle$ denotes the symmetric difference; we used $F = X \dot\cup F[W_0] \dot\cup \ldots \dot\cup F[W_p]$. Before proving that F' also fulfills the remaining two properties of a Φ-tour, we show the following claim. See Figure 9.4 for an illustration.

Claim 2. *Let $\overline{I} \coloneqq I_0 \dot\cup \ldots \dot\cup I_p$ and $a, b \in \overline{I}$. Suppose (V, F) contains an a-b path. Then (V, F') contains an a-b path.*

Proof of Claim 2. Suppose the claim is wrong. Then there exist vertices $a, b \in \overline{I}$ such that (V, F) contains an a-b path P, but (V, F') does not. We choose a, b, and P such that the number of edges of P is minimum. Consequently, P contains no vertex of $\overline{I} \setminus \{a, b\}$. We now distinguish two cases.

Case 1: $X \cap E(P) = \emptyset$.
Then P is completely contained in a single set W_i for some $i \in \{0, \dots, p\}$, by definition of X. Hence, $a, b \in W_i \cap \overline{I} = I_i$ and a and b are connected by the path P in $(W_i, F[W_i])$.

Since Φ_i is the interface induced by (F, Φ) on W_i, the vertices a and b are contained in the same set of the partition $\mathcal{C}_i$ of I_i. This implies that every Φ_i-tour, and in particular F_i, must contain an a-b path, contradicting the assumption that (V, F') contains no a-b path.

Case 2: $X \cap E(P) \neq \emptyset$.
Recall $X = F \cap \bigcup_{i=0}^{p} \delta(W_i)$. For $i \in \{0, \dots, p\}$, the set I_i contains all vertices of W_i that are an endpoint of an edge in X, by definition of the induced interface Φ_i. Thus all endpoints of edges in X are contained in $\overline{I}$. Since P contains no vertex of $\overline{I} \setminus \{a, b\}$, we have $X \cap E(P) = \{\{a, b\}\}$, i.e. the path P consists only of a single edge that is contained in X and thus also in F'. This contradicts our assumption that (V, F') contains no a-b path. □

(proof of Claim 2)

To show point (iii) of Definition 9.6, we need to show that any two vertices a and b that are contained in the same set of the partition $\mathcal{C}$ of I are also contained in the same connected component of (V, F'). If a and b are contained in the same set of the partition $\mathcal{C}$, they are contained in the same connected component of (V, F) because F is a Φ-tour. Hence by Claim 2 and $I \subseteq \bar{I}$, also (V, F') contains an a-b path.

It remains to show point (ii) of Definition 9.6, i.e., we prove that $(V, F')/I$ is connected. First observe that if $p = 0$, then the result holds because then $F' = F_0$ is a Φ_0-tour and $I_0 = I$. Hence, assume from now on $p > 0$. In this case, we first observe that

$$I_i \neq \emptyset \quad \forall i \in \{0, \dots, p\}. \tag{9.11}$$

Indeed, because $(V, F)/I$ is connected, which follows from F being a Φ-tour, we have for each $i \in \{0, \dots, p\}$ that either $I \cap W_i \neq \emptyset$ or $\delta(W_i) \cap F \neq \emptyset$, both of which imply $I_i \neq \emptyset$.

To conclude that $(V, F')/I$ is connected, we will observe the following two properties, which immediately imply the result:

(i) For each $i \in \{0, \dots, p\}$, each vertex $v \in W_i$ is connected to a vertex in I_i in the graph (W_i, F_i).

(ii) All vertices in $\cup_{i=0}^{p} I_i$ are connected in $(V, F')/I$.

Notice that (i) is a consequence of (9.11) and the fact that $(W_i, F_i)/I_i$ is connected, which holds because F_i is a Φ_i-tour in $G[W_i]$. Finally, (ii) follows from Claim 2 due to the following. Either $I = \emptyset$, in which case $(V, F)/I = (V, F)$ is connected—because F is a Φ-tour—which implies (ii) by Claim 2. Or $I \neq \emptyset$, in which case the connectivity of $(V, F)/I$ implies that in (V, F) each vertex $v \in \cup_{i=0}^{p} I_i$ is connected to a vertex of I, again implying (ii) by Claim 2. □

9.4.3 The dynamic program

We now expand on the dynamic program used to show Theorem 9.17. The dynamic program is formally described by Algorithm 7 below. See also Figure 9.5 for an illustration. Before formally proving that Algorithm 7 indeed returns a Φ-tour implying Theorem 9.17, we provide a brief explanatory discussion outlining the core ideas of the algorithm and the line of reasoning we employ to show its correctness.

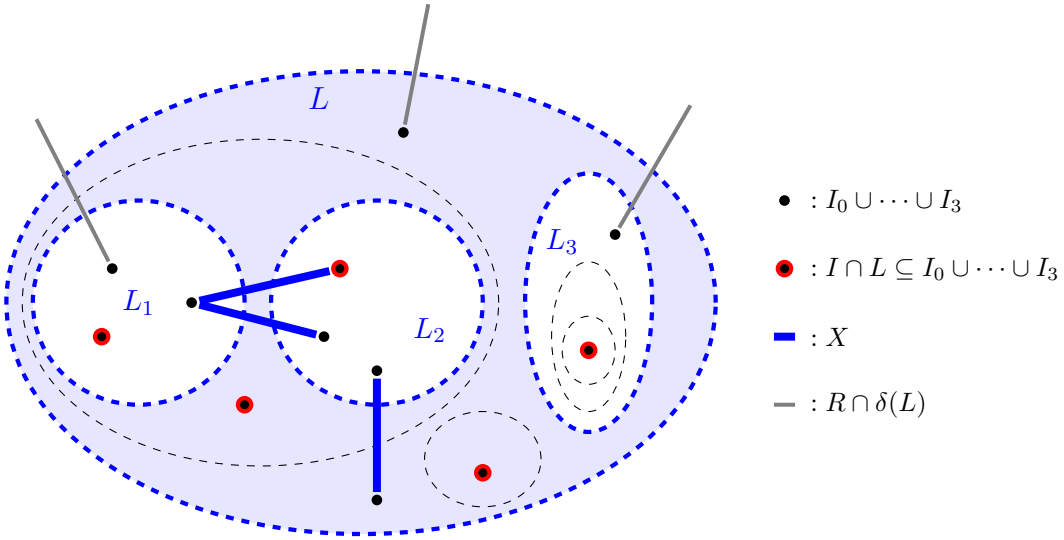

Figure 9.5: Illustration of Algorithm 7. The dashed ellipses show the laminar family $\mathcal{L}$; these sets are considered by the algorithm in an order of non-decreasing cardinality. Suppose we are considering $L \in \mathcal{L}(R,k)$; only subsets of L are shown in the figure. In the dynamic program we guess the children $L_1,\ldots,L_p$ of L in the laminar family $\mathcal{L}(R,k)$. The sets $L_1,\ldots,L_p$ are shown as blue ellipses with white interior. The light blue area shows the set $L_0 = L \setminus (L_1 \cup \cdots \cup L_p)$.
We also guess the set X of edges in $R[L] \cap (\delta(L_1) \cup \cdots \cup \delta(L_p))$; these are the thick blue edges. The thin gray edges are the edges in $R \cap \delta(L)$; these will be guessed only in a later step. However, we do guess the interface $\Phi_L = (I_L, T_L, \mathcal{C}_L)$ that (R,Φ) induces on L, where I_L consists of all vertices in $I \cap L$ (shown with a thick red boundary) and all vertices in L that are endpoints of gray edges. Moreover, we guess the interfaces $\Phi_i = (I_i, T_i, \mathcal{C}_i)$ that (R,Φ) induces on the sets L_i for $i \in \{0,\ldots,p\}$. The picture shows only the vertices in $I_0 \cup \cdots \cup I_p$; these are the vertices in L that are contained in the set I or are an endpoint of a thick blue or thin gray edge.
We compute a Φ_0-tour in $G[L_0]$ and combine it with X and the Φ_i-tours in $G[L_i]$ for $i \in \{1,\ldots,p\}$ that we have computed in previous steps of the dynamic program. This yields a Φ_L-tour in $G[L]$.

To this end, let R be a Φ-tour (unknown to the algorithm), and we will show that the dynamic program returns a Φ-tour $F \subseteq E$ such that $c(F) \leq \beta \cdot c(R) - (\beta - 1) \cdot c(R(\mathcal{L}, k))$. Conceptually, we want to consider the elements of the laminar family $\mathcal{L}(R,k) \subseteq \mathcal{L}$ from smaller to larger ones. Since we do not know the laminar family $\mathcal{L}(R,k)$, we consider all sets in $\mathcal{L}$ in an arbitrary fixed order of non-decreasing cardinality. We then guess, for every vertex set $L \in \mathcal{L}(R,k)$, the interface Φ_L induced by (R,Φ) on L. Now we compute a Φ_L-tour F_{L,Φ_L} in $G[L]$ as follows.

First, we guess the children $L_1, \ldots, L_p$ of L in the laminar family $\mathcal{L}(R,k)$. Then we guess the set $X \subseteq R[L]$ of edges that cross the cuts $\delta(L_1), \ldots, \delta(L_p)$. In other words, we guess all edges in $R(\mathcal{L},k)$ that are contained in L, but not in any child of L. Moreover, for each child L_i with $i \in \{1,\ldots,p\}$, we guess the interface Φ_i induced by (R,Φ) on L_i. Because we consider the elements of the laminar family $\mathcal{L}$ in an order of non-decreasing cardinality, we have already considered L_i before considering the current set L. Hence we have already computed some Φ_i-tour F_{L_i,Φ_i} for all $i \in \{1,\ldots,p\}$.

We now want to extend the union of these Φ_i-tours for all $i \in \{1,\ldots,p\}$ and the set X of edges crossing the boundaries of the children $L_1, \ldots, L_p$ to a Φ_L-tour in $G[L]$. To this end we define $L_0 \coloneqq L \setminus \cup_{i=1}^{p} L_i$. Then $L_0, \ldots, L_p$ is a partition of L. We also guess the interface Φ_0 that (R,Φ) induces on L_0. Then, by Lemma 9.22 applied to the graph $G[L]$, the union of X and arbitrary Φ_i-tours in $G[L_i]$ for $i = \{0,\ldots,p\}$ is a Φ_L-tour in $G[L]$. Here we use that Φ_i is the interface induced by $(R[L],\Phi_L)$ on L_i for $i = \{0,\ldots,p\}$ (cf. Lemma 9.21 (iii)). Finally, we use the given algorithm $\mathcal{B}$ to compute a β-approximation F_0 of a minimum length Φ_0-tour in the subgraph $G[L_0]$ and combine X, F_0, and the Φ_i-tours F_{L_i,Φ_i} for $i \in \{1,\ldots,p\}$ to a Φ_L-tour F_{L,Φ_L}.

Algorithm 7: Algorithm computing a Φ-tour as stated in Theorem 9.17

```
Let 𝓛̄ = 𝓛 ∪ {V};
for L ∈ 𝓛̄, in non-decreasing order of cardinality do
    for each interface Φ_L = (I_L, T_L, 𝒞_L) of G[L] with |I_L| ≤ |I| + k do
        Let F_{L,Φ_L} ≔ Nil;                          // No Φ_L-tour found yet.
                                                      // We set by convention: c(Nil) = ∞.
        for each subfamily {L_1, ..., L_p} ⊆ 𝓛̄ of disjoint proper subsets of L do
            Let L_0 ≔ L \ ∪_{i=1}^{p} L_i;
            for all X ⊆ (⋃_{i=1}^{p} δ(L_i)) ∩ E[L] with |X ∩ δ(L_i)| ≤ k ∀i ∈ {1,...,p} do
                For i ∈ {0,...,p}, let U_i ⊆ L_i be the set of all vertices in L_i that are an
                  endpoint of some edge in X;
                Let I_i ≔ (I_L ∩ L_i) ∪ U_i ∀i ∈ {0,...,p};
                for all T_i, 𝒞_i such that Φ_i = (I_i, T_i, 𝒞_i) is an interface of
                  G[L_i]  (i ∈ {0,...,p}) do
                    Use Algorithm 𝓑 to find a Φ_0-tour F_0;
                    Let F ≔ X ∪̇ F_0 ∪̇ ⋃_{i=1}^{p} F_{L_i,Φ_i};
                    if F is a Φ_L-tour and c(F) ≤ c(F_{L,Φ_L}) then
                        Set F_{L,Φ_L} = F;
return F_{V,Φ};
```

In what follows, we now provide a rigorous proof that Algorithm 7 implies Theorem 9.17 by leveraging the tools from Section 9.4.2.

9.4.4 Proof of Theorem 9.17

We start by showing that Algorithm 7 has indeed the claimed running time, before proving its correctness.

Running time

The running time of Algorithm 7 is dominated by the 5-fold nested for-loops. We first determine upper bounds on the number of iterations of each for-loop separately, whenever the algorithm reaches it.

1^{st} for-loop: It goes over all sets in $\overline{\mathcal{L}}$. Because $\overline{\mathcal{L}}$ is a laminar family over V, it contains $O(|V|)$ sets (by Lemma 2.6).

2^{nd} for-loop: It goes over all interfaces $\Phi_L = (I_L, T_L, \mathcal{C}_L)$ of $G[L]$ with $|I_L| \leq |I|+k$. There are no more than $(|L|+1)^{|I|+k} \leq (|V|+1)^{|I|+k}$ choices for choosing I_L. Moreover, there are at most $2^{|I_L|} \leq 2^{|I|+k}$ choices for $T_L \subseteq I_L$. Finally, the number of partitions $\mathcal{C}_L$ of I_L can be upper bounded by $|I_L|^{|I_L|} \leq |V|^{|I|+k}$. Overall, the number of iterations of any run of the second for-loop is bounded by $|V|^{O(|I|+k)}$.

3^{rd} for-loop: It iterates over subfamilies of $\overline{\mathcal{L}}$ of disjoint proper subsets of L. Because the sets are disjoint, such a family can have at most $\text{width}(\overline{\mathcal{L}}) \leq \text{width}(\mathcal{L}) + 1$ sets, and we can therefore bound the number of these subfamilies by $|\overline{\mathcal{L}}|^{\text{width}(\overline{\mathcal{L}})} = |V|^{O(\text{width}(\mathcal{L}))}$.

4^{th} for-loop: It iterates over edge sets $X \subseteq (\cup_{i=1}^{p} \delta(L_i)) \cap E[L]$ with $|X \cap \delta(L_i)| \leq k$ for all $i \in \{1, \ldots, p\}$, and can be bounded as follows. Notice that

$$|X| \leq \sum_{i=1}^{p} |X \cap \delta(L_i)| \leq p \cdot k \leq \text{width}(\mathcal{L}) \cdot k.$$

Hence, there are at most $(|E|+1)^{k \cdot \text{width}(\mathcal{L})} = |V|^{O(k \cdot \text{width}(\mathcal{L}))}$ options for X.

5^{th} for-loop: This loop runs for all $i \in \{0, \ldots, p\}$ over all interfaces $\Phi_i = (I_i, T_i, \mathcal{C}_i)$ of $G[L_i]$, where L_i and I_i are fixed. The number of interfaces Φ_i for a fixed $i \in \{0, \ldots, p\}$ is thus bounded by $(2|I_i|)^{|I_i|} \leq (2|V|)^{|I_i|}$ and, hence, the total number of combinations of such interfaces, and thus also on the number of iterations each time this for-loop is run, is bounded by

$$\prod_{i=0}^{p} (2|V|)^{|I_i|} = (2|V|)^{\sum_{i=0}^{p} |I_i|}. \tag{9.12}$$

Moreover, for $i \in \{1, \ldots, p\}$, we have $|I_i| \leq k + |I_L \cap L_i|$, which follows from the fact that each set I_i contains the elements of $I_L \cap L_i$ together

with at most k endpoints of edges from X because $|X \cap \delta(L_i)| \leq k$. This implies

$$\begin{aligned}\sum_{i=1}^{p} |I_i| \leq\ & p \cdot k + |I_L| \ \leq\ \text{width}(\mathcal{L}) \cdot k + (|I| + k) \\ =\ & O(|I| + k \cdot \text{width}(\mathcal{L})).\end{aligned} \tag{9.13}$$

Similarly,

$$\begin{aligned}|I_0| \leq\ & |I_L| + k \cdot \text{width}(\mathcal{L}) \ \leq\ |I| + k + k \cdot \text{width}(\mathcal{L}) \\ =\ & O(|I| + k \cdot \text{width}(\mathcal{L})).\end{aligned} \tag{9.14}$$

Combining (9.13) and (9.14) with (9.12), we can bound the number of iterations of the fifth for-loop by $|V|^{O(|I|+k\cdot\text{width}(\mathcal{L}))}$.

The most expensive single operation performed by Algorithm 7 is the call to Algorithm $\mathcal{B}$ to find a Φ_0-tour, which, by assumption, takes no more than $f_{\mathcal{B}}(G, |I_0|)$ time. Due to the bound on $|I_0| \leq |I| + k \cdot (\text{width}(\mathcal{L}) + 1)$ provided by (9.14), we have that the total running time is thus indeed bounded by

$$|V|^{O(|I|+k\cdot\text{width}(\mathcal{L}))} \cdot f_{\mathcal{B}}(G, |I| + k \cdot (\text{width}(\mathcal{L}) + 1)).$$

Correctness

We now show that, whenever G admits a Φ-tour, then Algorithm 7 will find a Φ-tour $F_{V,\Phi}$ with the length guarantee claimed by Theorem 9.17. So let R be a Φ-tour. We have to show that $F_{V,\Phi}$ computed by the algorithm is a Φ-tour (instead of Nil) and that it satisfies

$$c(F_{V,\Phi}) \leq \beta \cdot c(R \setminus R(\mathcal{L}, k)) + c(R(\mathcal{L}, k)). \tag{9.15}$$

We prove (9.15) by showing the following claim from smaller to larger sets $L \in \mathcal{L}(R, k) \cup \{V\}$.

Claim 3. *Let $L \in \mathcal{L}(R, k) \cup \{V\}$. If $L = V$, let $\Phi_L = \Phi$. Otherwise, let $\Phi_L = (I_L, T_L, \mathcal{C}_L)$ be the interface induced by (R, Φ) on L. Then Algorithm 7 computes a Φ_L-tour F_{L,Φ_L} such that*

$$c(F_{L,\Phi_L}) \leq \beta \cdot c\left(R[L] \setminus R(\mathcal{L}, k)\right) + c\left(R[L] \cap R(\mathcal{L}, k)\right).$$

Observe that the claim immediately implies Theorem 9.17 by choosing $L = V$. Hence, it remains to prove the claim.

Proof of Claim 3. We prove the claim by induction from smaller to larger sets in $\mathcal{L}(R, k) \cup \{V\}$. Hence, let $L \in \mathcal{L}(R, k) \cup \{V\}$ and assume that the claim holds for sets in $\mathcal{L}(R, k) \cup \{V\}$ of strictly smaller cardinality than L. In particular, it holds for the children $L_1, \ldots, L_p$ of L in the laminar family $\mathcal{L}(R, k) \cup \{V\}$. (Note that L may also not have any children.) Let $L_0 \coloneqq L \setminus \cup_{i=1}^{p} L_i$, and for $i \in \{0, \ldots, p\}$, let $\Phi_i = (I_i, T_i, \mathcal{C}_i)$ be the interface induced by (R, Φ) on L_i. By using Lemma 9.21 (iii) in the case $L \neq V$, we observe that Φ_i is also the interface induced by $(R[L], \Phi_L)$ on L_i. Let F_0 be a

Φ_0-tour obtained through Algorithm $\mathcal{B}$. Because $L_0, L_1, \ldots, L_p$ partitions L, we have by Lemma 9.22 that

$$F \coloneqq X \mathbin{\dot\cup} F_0 \mathbin{\dot\cup} \bigcup_{i=1}^{p} F_{L_i,\Phi_i}$$

is a Φ_L-tour, where

$$X \coloneqq R[L] \cap \bigcup_{i=1}^{p} \delta(L_i). \tag{9.16}$$

Before discussing that this Φ_L-tour F will indeed be considered by Algorithm 7, we bound its length. First, $c(F_0) \leq \beta \cdot c(R[L_0])$ because $\mathcal{B}$ computes β-approximate solutions and $R[L_0]$ is a Φ_0-tour by Lemma 9.21 (ii). Moreover, for $i \in \{1, \ldots, p\}$ we apply the induction hypothesis to L_i and Φ_i, which is possible because $L_i \in \mathcal{L}(R,k)$ has strictly smaller cardinality than L. Hence, F_{L_i,Φ_i} is a Φ_i-tour and fulfills the length bound stated in the claim. We therefore get

$$\begin{aligned} c(F) &= c(X) + c(F_0) + \sum_{i=1}^{p} c(F_{L_i,\Phi_i}) \\ &\leq c(X) + \beta \cdot c(R[L_0]) + \sum_{i=1}^{p} \left(\beta \cdot c(R[L_i] \setminus R(\mathcal{L},k)) + c(R[L_i] \cap R(\mathcal{L},k))\right) \\ &= \beta \cdot c(R[L] \setminus R(\mathcal{L},k)) + c(R[L] \cap R(\mathcal{L},k)), \end{aligned} \tag{9.17}$$

where the last equality follows by observing that

$$\begin{aligned} R[L_0],\ R[L_1] \setminus R(\mathcal{L},k), \ldots, R[L_p] \setminus R(\mathcal{L},k) \quad &\text{partitions} \quad R[L] \setminus R(\mathcal{L},k), \text{ and} \\ X,\ R[L_1] \cap R(\mathcal{L},k), \ldots, R[L_p] \cap R(\mathcal{L},k) \quad &\text{partitions} \quad R[L] \cap R(\mathcal{L},k). \end{aligned}$$

Due to (9.17), the Φ_L-tour F fulfills the length bound of the claim. It remains to show that the Φ_L-tour F will indeed be considered by Algorithm 7. For this, we show that the following quantities are considered in the five nested for-loops:

- 1^{st} for-loop: considers L,
- 2^{nd} for-loop: considers the interface $\Phi_L = (I_L, T_L, \mathcal{C}_L)$,
- 3^{rd} for-loop: considers the children $L_1, \ldots, L_p$ of the set L in the laminar family $\mathcal{L}(R,k) \cup \{V\}$,
- 4^{th} for-loop: considers the set X,
- 5^{th} for-loop: considers, for $i \in \{0, \ldots, p\}$, the interfaces Φ_i induced by $(R[L], \Phi_L)$ on L_i.

This run would indeed produce F. All that remains to be shown is that the above five quantities, to be considered within the five nested for-loops, fulfill the conditions set by the respective for-loops:

- 1^{st} for-loop: Algorithm 7 considers all sets in $\mathcal{L}$ and hence, also L.
- 2^{nd} for-loop: If $L = V$, the interface Φ is obviously considered. Otherwise $\Phi_L = (I_L, T_L, \mathcal{C}_L)$ is the interface induced by (R, Φ) on L, and we have

$I_L = (I \cap L) \cup U$, where U is the set of vertices in L connected by an edge of R to a vertex in $V \setminus L$. As $L \in \mathcal{L}(R,k) \cup \{V\}$, we have $|\delta(L) \cap R| \leq k$, and hence $|U| \leq k$, which implies $|I_L| \leq |I| + k$ and shows that the interface Φ_L is considered in the second for-loop.

3^{rd} for-loop: We have $\{L_1, \ldots, L_p\} \subseteq \overline{\mathcal{L}}$. Hence, the subfamily $\{L_1, \ldots, L_p\}$ will be considered in the third nested for-loop.

4^{th} for-loop: The set X we want to consider is given by (9.16). This set clearly satisfies $X \subseteq (\cup_{i=1}^{p} \delta(L_i)) \cap E[L]$ because $R[L] \subseteq E[L]$. Moreover, for each $i \in \{1, \ldots, p\}$ we have

$$|X \cap \delta(L_i)| = |R[L] \cap \delta(L_i)| \leq |R \cap \delta(L_i)| \leq k,$$

where the last inequality follows from $L_i \in \mathcal{L}(R,k)$. Hence, the set X will be considered during the fourth nested for-loop of the algorithm.

5^{th} for-loop: For $i \in \{0, \ldots, p\}$ we have that $\Phi_i = (I_i, T_i, \mathcal{C}_i)$ is the interface of $G[L_i]$ induced by $(R[L], \Phi_L)$ on L_i. Hence, $I_i = (I_L \cap L_i) \cup U_i$, where U_i are all vertices in L_i connected by an edge of $R[L]$ to a vertex in $L \setminus L_i$. We have $R[L] \cap \delta(L_i) = X \cap \delta(L_i)$ by our choice of X as described in (9.16) and because $\{L_0, \ldots, L_p\}$ is a partition of L. Therefore, $I_i \coloneqq (I_L \cap L_i) \cup U_i$, as desired. Hence, the interfaces Φ_i for $i \in \{0, \ldots, p\}$ indeed get considered in the fifth nested for-loop of the algorithm.

□

As said, Claim 3 implies (9.15), completing the proof of Theorem 9.17.

We remark that Claim 3 can be slightly strengthened as follows. The statement also holds when replacing the induced interface $\Phi_L = (I_L, T_L, \mathcal{C}_L)$ by any interface $\Phi'_L = (I_L, T_L, \mathcal{C}'_L)$ where $\mathcal{C}'_L$ is a refinement of $\mathcal{C}_L$. However, we do not need this for our purposes.

9.5 Proof of the main theorem

We finally prove that the Boosting Theorem (Theorem 9.10) implies Theorem 9.1.

In fact, we prove a generalization, stated below as Theorem 9.23, which, for $k = 2$ and $\Phi = (I, T, \mathcal{C})$ with $I = T = \{s, t\}$ and $\mathcal{C} = \{\{s, t\}\}$, yields Theorem 9.1.

Theorem 9.23. *Let $\alpha > 1$. Let $\mathcal{A}$ be an algorithm that computes α-approximate solutions for TSP. Then, for any $\varepsilon > 0$ and any integer k, there is an $(\alpha+\varepsilon)$-approximation algorithm for Φ-TSP restricted to instances with $|I_\Phi| \leq k$ that, for any instance (G, Φ), calls $\mathcal{A}$ a polynomial number of times on TSP instances defined on subgraphs of G, and performs further operations taking polynomial time.*

Proof. We obtain the result by repeatedly applying the Boosting Theorem, i.e. Theorem 9.10, to strengthen the 4-approximation algorithm for Φ-TSP guaranteed by Theorem 9.9 through the α-approximation algorithm for TSP which we assume to exist.

Without loss of generality $\varepsilon \leq 1$. The Boosting Theorem will be repeated $i_{\max}$ many times with error parameter given by $\varepsilon' = \frac{\varepsilon}{\alpha}$, where

$$i_{\max} := \left\lceil \frac{4 - (\alpha + \varepsilon)}{\alpha - 1} \cdot \frac{8\alpha}{\varepsilon} \right\rceil.$$

Notice that $i_{\max}$ is constant, because both ε and α are fixed.

Let $\beta_0 := 4$ be the approximation ratio for Φ-TSP before applying the Boosting Theorem. We assume $\alpha \leq 1.5 < \beta_0$ because Christofides' algorithm is a 1.5-approximation algorithm for TSP. Let $i \in \{1, \ldots, i_{\max}\}$. After i applications of the Boosting Theorem we obtain an algorithm $\mathcal{B}_i$ that computes β_i-approximate solutions for Φ-TSP with

$$\begin{aligned} \beta_i &:= \max\left\{(1+\varepsilon')\alpha,\ \beta_{i-1} - \frac{\varepsilon'}{8} \cdot (\beta_{i-1} - 1)\right\} \\ &= \max\left\{\alpha + \varepsilon,\ \beta_{i-1} - \frac{\varepsilon}{8\alpha} \cdot (\beta_{i-1} - 1)\right\}, \end{aligned}$$

where we used $\varepsilon' = \frac{\varepsilon}{\alpha}$. We therefore have

$$\beta_i \leq \max\left\{\alpha + \varepsilon,\ \beta_{i-1} - \frac{\varepsilon}{8\alpha}(\alpha - 1)\right\} \leq \max\left\{\alpha + \varepsilon,\ \beta_0 - i \cdot \frac{\varepsilon}{8\alpha}(\alpha - 1)\right\},$$

where the last inequality follows by induction on i. Hence,

$$\begin{aligned} \beta_{i_{\max}} &\leq \max\left\{\alpha + \varepsilon,\ 4 - i_{\max} \cdot \frac{\varepsilon}{8\alpha}(\alpha - 1)\right\} \\ &= \max\left\{\alpha + \varepsilon,\ 4 - \left\lceil \frac{4 - (\alpha + \varepsilon)}{\alpha - 1} \cdot \frac{8\alpha}{\varepsilon} \right\rceil \cdot \frac{\varepsilon}{8\alpha}(\alpha - 1)\right\} \\ &= \alpha + \varepsilon. \end{aligned}$$

Moreover, we define real numbers $k_i > 0$ for $i \in \{0, \ldots, i_{\max}\}$ to upper bound the size of the interfaces we have to be able to handle after i boosting steps. We want the algorithm $\mathcal{B}_{i_{\max}}$, obtained after $i_{\max}$ many applications of the Boosting Theorem, to handle interfaces of size $k_{i_{\max}} := k$. Because $\mathcal{B}_{i_{\max}}$ was obtained by applying the Boosting Theorem to $\mathcal{B}_{i_{\max}-1}$, we obtain that $\mathcal{B}_{i_{\max}-1}$ needs to handle interfaces of size bounded by $k_{i_{\max}-1} := \frac{9}{\varepsilon'} \cdot k_{i_{\max}}$. Repeating this reasoning, we obtain upper bounds k_i on the size of the interfaces that we have to handle with $\mathcal{B}_i$ that satisfy

$$k_i := \frac{9}{\varepsilon'} \cdot k_{i+1} \quad \forall i \in \{i_{\max} - 1, \ldots, 1, 0\},$$

which implies

$$k_i = k \cdot \left(\frac{9}{\varepsilon'}\right)^{i_{\max} - i} \qquad \forall i \in \{0, \ldots, i_{\max}\}.$$

Notice that because $i_{\max}$, k, and ε' are constant, also k_0 is constant.

For $i = i_{\max}$, the following claim implies Theorem 9.23 because $\beta_{i_{\max}} = \alpha + \varepsilon$ and $i_{\max}$, k_0, and ε' are constant and $\mathcal{B}_0$ is a polynomial-time algorithm.

Claim 4. *Let $c > 0$ be the hidden constant in the big-O notation in the runtime bound in Theorem 9.10. Let $i \in \{0, \ldots, i_{\max}\}$ and let $\mathcal{A}$ be the given α-approximation algorithm*

for TSP. Then there is an algorithm $\mathcal{B}_i$ that computes β_i-approximate solutions for Φ-TSP and for every weighted graph G, runs in time at most

$$f_i(G) \coloneqq |V|^{i \cdot c \cdot \frac{k_0}{\varepsilon'}} \cdot \left(i \cdot f_{\mathcal{A}}(G) + f_{\mathcal{B}_0}(G)\right) \tag{9.18}$$

on any instance (G', Φ), where G' is a subgraph of G and $|I_\Phi| \leq k_i$.

We prove the claim by induction on i. By Theorem 9.9 we have a β_0-approximation algorithm $\mathcal{B}_0$ for Φ-TSP, implying the claim for $i = 0$.

Now let $i \in \{1, \ldots, i_{\max}\}$. By our induction hypothesis, there exists an algorithm $\mathcal{B}_{i-1}$ that computes β_{i-1}-approximate solutions and runs in time $f_{i-1}(G)$ on every weighted graph G and every interface Φ of G with $|I_\Phi| \leq k_{i-1}$. Applying Theorem 9.10 to the algorithms $\mathcal{A}$ and $\mathcal{B}_{i-1}$ then yields an algorithm $\mathcal{B}_i$ that computes β_i-approximate solutions for Φ-TSP and runs on every graph G and every interface Φ with $|I_\Phi| \leq k_i$ in time at most

$$\begin{aligned}
&|V|^{c \cdot \frac{k_i}{\varepsilon'}} \cdot \left(f_{\mathcal{A}}(G) + f_{\mathcal{B}_{i-1}}\left(G, \tfrac{9 \cdot k_i}{\varepsilon'}\right)\right) \\
&\leq |V|^{c \cdot \frac{k_0}{\varepsilon'}} \cdot \left(f_{\mathcal{A}}(G) + f_{\mathcal{B}_{i-1}}\left(G, k_{i-1}\right)\right) \\
&\leq |V|^{c \cdot \frac{k_0}{\varepsilon'}} \cdot \left(f_{\mathcal{A}}(G) + f_{i-1}(G)\right) \\
&= |V|^{c \cdot \frac{k_0}{\varepsilon'}} \cdot f_{\mathcal{A}}(G) + |V|^{i \cdot c \cdot \frac{k_0}{\varepsilon'}} \cdot \left((i-1) \cdot f_{\mathcal{A}}(G) + f_{\mathcal{B}_0}(G)\right) \\
&\leq f_i(G).
\end{aligned}$$

□

9.6 A 4-approximation algorithm for Φ-TSP

Theorem 9.9. *There is a 4-approximation algorithm for Φ-TSP.*

Proof. Let $\Phi = (I, T, \mathcal{C})$ be an interface of $G = (V, E)$. By Lemma 9.8, we can assume that Φ is feasible. The main component of our algorithm is to obtain a 2-approximation algorithm for the problem of finding a set (not a multi-set) $F \subseteq E$ of minimum length $c(F)$ that satisfies the following three conditions:

(i) $(V, F)/I$ is connected;

(ii) (V, F) connects all vertices within any $C \in \mathcal{C}$;

(iii) each connected component of (V, F) contains an even number of vertices in T.

We will achieve this through an application of Jain's iterative rounding method for the Generalized Steiner Network Problem [Jai01].

Before we discuss the details of Jain's method in our setting, we first assume that we can indeed find in polynomial time a set $F \subseteq E$ fulfilling (i), (ii), and (iii) of length no larger than twice the length of a shortest edge set fulfilling these three conditions. Because a shortest Φ-tour F^* must fulfill these conditions, and removing parallel edges does not destroy them, there is a subset of F^* that contains no parallel edges and satisfies (i), (ii), and (iii). Therefore, $c(F) \leq 2 \cdot \mathrm{OPT}$.

Due to property (iii), the set F contains a T-join $J \subseteq F$, which we can find in linear time through standard techniques. We then return $F \dot\cup (F \setminus J)$, which is indeed a Φ-tour and satisfies

$$c(F \dot\cup (F \setminus J)) \leq 2c(F) \leq 4 \cdot \mathrm{OPT},$$

as desired. It remains to show how to obtain a polynomial 2-approximation algorithm for finding a shortest edge set fulfilling (i), (ii), and (iii).

To this end, observe that a set $F \subseteq E$ satisfies (i), (ii), and (iii) if and only if

$$|F \cap \delta(S)| \geq f(S) \qquad \forall S \subseteq V, \tag{9.19}$$

where the function $f : 2^V \to \{0, 1\}$ is defined as follows. For $S \subsetneq V$ with $S \neq \emptyset$, we set $f(S) = 1$ if at least one of the following three properties holds:

(a) $S \cap I = \emptyset$;

(b) $\exists\, C \in \mathcal{C}$ s.t. $S \cap C \neq \emptyset$ and $C \setminus S \neq \emptyset$;

(c) $|S \cap T|$ is odd.

Otherwise we set $f(S) = 0$. (In particular, $f(\emptyset) = f(V) = 0$.) Indeed, the properties (a), (b), and (c) are just reformulations of (i), (ii), and (iii), respectively.

Jain's technique [Jai01] leads to a 2-approximation algorithm for finding a shortest edge set F satisfying (9.19) if, first, the function f is *weakly supermodular*, which means that for all $X, Y \subseteq V$

$$f(X) + f(Y) \leq \max\left\{f(X \cup Y) + f(X \cap Y),\, f(X \setminus Y) + f(Y \setminus X)\right\}, \tag{9.20}$$

and, second, one can separate over the polytope

$$P = \left\{x \in [0,1]^E : x(\delta(S)) \geq f(S)\ \ \forall S \subseteq V\right\}. \tag{9.21}$$

in polynomial time.

We start by showing (9.20). Notice that (9.20) clearly holds if $X \subseteq Y$, because in this case we have $\{X, Y\} = \{X \cup Y, X \cap Y\}$. Hence, in what follows, we always assume that $X \setminus Y \neq \emptyset$ and $Y \setminus X \neq \emptyset$.

Let f_a, f_b, and f_c be the functions from 2^V to $\{0, 1\}$ that take a value of 1 precisely for sets $S \subsetneq V, S \neq \emptyset$ that satisfy (a), (b), or (c), respectively. Hence, $f(S) = \max\{f_a(S), f_b(S), f_c(S)\}$. First, one can observe that each of the functions f_a, f_b, and f_c is weakly supermodular. Consider first f_a and let $X, Y \subseteq V$ with $X \setminus Y \neq \emptyset$ and $Y \setminus X \neq \emptyset$. If $f_a(X) = 1$ then $f_a(X \setminus Y) = 1$. Similarly, if $f_a(Y) = 1$, then $f_a(Y \setminus X) = 1$. Hence, f_a satisfies (9.20). The function f_b corresponds to pairwise connectivity requirements and, as shown in [Jai01], is therefore weakly supermodular. The function f_c is easily seen to be a so-called *proper* function, which means that $f_c(V) = 0$, f_c is symmetric, and $f_c(S_1 \cup S_2) \leq \max\{f_c(S_1), f_c(S_2)\}$ for any pair of disjoint sets $S_1, S_2 \subseteq V$. Finally, it is well-known that any proper function is weakly supermodular (see [GGP+94]).

We say that a set $S \subsetneq V$ with $S \neq \emptyset$ is of type (a), (b), or (c), if it satisfies (a), (b), or (c), respectively. Because each of the functions f_a, f_b, and f_c is weakly supermodular, the inequality (9.20) holds whenever the sets X and Y are of the same type, or if X or

Y is none of the three types. Hence, it remains to consider sets X and Y of two different types among the types (a), (b), and (c). Let $S_a, S_b, S_c \subseteq V$ be sets of type (a), (b), and (c), respectively. Thus, we need to show that (9.20) holds for the three cases where (X, Y) is either (S_a, S_b), (S_a, S_c), or (S_b, S_c). Moreover, let $C \in \mathcal{C}$ be a set such that $S_b \cap C \neq \emptyset$ and $C \setminus S_b \neq \emptyset$, which exists because S_b is of type (b).

We start by considering the case $(X, Y) = (S_a, S_b)$. As discussed, we assume that $S_a \not\subseteq S_b$ and $S_b \not\subseteq S_a$; for otherwise, (9.20) holds trivially. Notice that in this case we have

$$2 = f(S_a) + f(S_b) \leq f(S_a \setminus S_b) + f(S_b \setminus S_a) = 2,$$

because $(S_a \setminus S_b) \cap I \subseteq S_a \cap I = \emptyset$, as well as $(S_b \setminus S_a) \cap I = S_b \cap I$ and $C \subseteq I$. Hence, $S_a \setminus S_b$ is of type (a) and $S_b \setminus S_a$ is of type (b).

Consider now the case $(X, Y) = (S_a, S_c)$. Here, we have

$$2 = f(S_a) + f(S_c) \leq f(S_a \setminus S_c) + f(S_c \setminus S_a) = 2,$$

because $(S_a \setminus S_c) \cap I \subseteq S_a \cap I = \emptyset$, implying that $S_a \setminus S_c$ is of type (a), and $|(S_c \setminus S_a) \cap T| = |S_c \cap T|$ due to $S_a \cap T \subseteq S_a \cap I = \emptyset$, which implies that $S_c \setminus S_a$ is of type (c).

It remains to consider the case $(X, Y) = (S_b, S_c)$. We first observe that

$$\max\{f(S_b \setminus S_c),\ f(S_b \cup S_c)\} \geq 1, \text{ and} \tag{9.22}$$
$$\max\{f(S_b \cap S_c),\ f(S_c \setminus S_b)\} \geq 1, \tag{9.23}$$

due to the following. Inequality (9.22) holds because $S_b \cup S_c$ can be partitioned into S_c and $S_b \setminus S_c$. Because $|S_c \cap T|$ is odd, either $S_b \cup S_c$ or $S_b \setminus S_c$ must also have an odd intersection with T and is thus of type (c). Inequality (9.23) follows from an analogous reasoning using the partition of S_c into $S_b \cap S_c$ and $S_c \setminus S_b$. Moreover, we have

$$\max\{f(S_b \setminus S_c),\ f(S_b \cap S_c)\} \geq 1, \text{ and} \tag{9.24}$$
$$\max\{f(S_b \cup S_c),\ f(S_c \setminus S_b)\} \geq 1, \tag{9.25}$$

because S_b is of type (b), i.e., $S_b \cap C \neq \emptyset$ and $C \setminus S_b \neq \emptyset$. Indeed, even without any assumptions on $S_c \subseteq V$, we have that either $S_b \setminus S_c$ or $S_b \cap S_c$ is also of type (b). The same holds for either $S_b \cup S_c$ or $S_c \setminus S_b$. Among the four expressions $f(S_b \cup S_c)$, $f(S_b \cap S_c)$, $f(S_b \setminus S_c)$, and $f(S_c \setminus S_b)$, consider any one of minimum value and sum up the two inequalities among (9.22), (9.23), (9.24), and (9.25) containing that expression. This gives the desired result. For example, if $f(S_b \setminus S_c)$ achieves minimum value among the four, then (9.22) implies $f(S_b \cup S_c) = 1$ and (9.24) implies $f(S_b \cap S_c) = 1$. Hence,

$$2 = f(S_b) + f(S_c) \leq f(S_b \cup S_c) + f(S_b \cap S_c) = 2,$$

as desired. This completes the proof that f is weakly supermodular.

To apply Jain's method, it remains to show that we can separate over P, and we will in fact give a polynomial algorithm. Given $y \in [0, 1]^E$, we will either show that all constraints $y(\delta(S)) \geq f(S)$ for $S \subseteq V$ are fulfilled or return one of these constraints that is violated. Notice that, because $y \geq 0$, a constraint $y(\delta(S)) \geq f(S)$ can only be violated if $f(S) = 1$, i.e., S is either of type (a), (b), (c). Hence, we can check these constraints for each type separately.

Whether there is a violated constraints of type (a) reduces to finding a minimizer of

$$\min \{y(\delta(S)) : S \subseteq V \text{ with } S \cap I = \emptyset\}.$$

This can be solved through a global minimum cut algorithm applied to the graph G/I with edge weights y. Indeed, this either leads to a cut S with $S \cap I = \emptyset$ as desired or one where $I \subseteq S$, in which we can replace S by $V \setminus S$.

To check whether there is a violated constraint of type (b) reduces to

$$\min \{y(\delta(S)) : \exists C \in \mathcal{C} \text{ with } S \cap C \neq \emptyset \text{ and } C \setminus S \neq \emptyset\}.$$

This can be solved by performing the following for all $C \in \mathcal{C}$ with $|C| \geq 2$. Number the vertices in C arbitrarily $C = \{c_1, \ldots, c_k\}$, and solve a minimum c_i-c_{i+1} cut problem in G with edge weights y for each $i \in \{1, \ldots, k-1\}$. If any of these s-t-cut problems leads to a cut of value strictly smaller than 1, then the minimizing cut corresponds to a violated constraint. Otherwise, there is no violated constraints of type $y(\delta(S)) \geq f(S)$ for any set S of type (b).

Finally, checking whether there is a violated constraint of type (c) reduces to

$$\min \{y(\delta(S)) : S \subseteq V, |S \cap T| \text{ is odd}\}.$$

This is a minimum weight T-cut problem, for which polynomial-time algorithms are well known (see, e.g. [Sch03]).

In summary, the separation problem over P can be solved in polynomial time, and we can therefore apply Jain's technique as claimed.

□

Chapter 10

Conclusions and open questions

In this thesis we studied TSP, ATSP and their path versions, as well as their unit-weight special cases. We gave new approximation algorithms and better upper bounds on the integrality ratio of the classical LP relaxations for many of these problems. For a summary of what we now know on these topics see Section 1.2. In this chapter we will discuss some open questions and possible directions for further research.

10.1 ATSP and its path version

In Chapter 4 we gave a $(22 + \varepsilon)$-approximation algorithm for ATSP for every fixed $\varepsilon > 0$ and proved that the integrality ratio of (ATSP LP) is at most 22. The best known lower bound on the integrality ratio is 2 [CGK06]. So determining the exact integrality ratio of (ATSP LP) remains an open question.

Open question 1
What is the integrality ratio of (ATSP LP)?

This question is also open for graph ATSP. We showed a lower bound of 2 in Section 4.6 and the best known upper bound is 13 [Sve15].

For the path version of ATSP we proved the first constant upper bound on the integrality ratio of (ATSPP LP); see Chapter 5. We also showed that the integrality ratio of (ATSPP LP) is less than four times larger than the integrality ratio of (ATSP LP). While the integrality ratio of (ATSPP LP) is clearly not smaller than the integrality ratio of (ATSP LP), it might be possible that it is equal.

Open question 2
Are the integrality ratios of (ATSP LP) for ATSP and (ATSPP LP) for s-t-path ATSP different?

At the moment, the best known lower bounds on the integrality ratios of these two LP relaxations are both 2 [CGK06]. However, for (ATSPP LP), Friggstad, Gupta, and Singh [FGM16] gave a simpler family of examples proving this lower bound; see Figure 5.5.

In Chapter 9 we proved that in the symmetric case the path version is not substantially harder to approximate than TSP. A natural question is whether the same holds for the asymmetric versions.

Open question 3
Is s-t-path ATSP substantially harder to approximate than ATSP?

Feige and Singh [FS07] showed that any α-approximation algorithm for ATSP implies that there is a $(2\alpha+\varepsilon)$-approximation algorithm for its path version (for any fixed $\varepsilon > 0$). One might try to extend the techniques that we used for our reduction in the symmetric case (Chapter 9) to the asymmetric setting in order to avoid the loss of a factor 2 in the approximation ratio.

If the distance from s to t is large, one can still apply dynamic programming as in Section 8.8 and Chapter 9. Otherwise, the distance from s to t is small, but in the asymmetric setting the distance from t to s could still be large. In this situation we do not know how to reduce the problem to ATSP or guess edges of significant weight by a dynamic program.

Another obstacle is the following. Our reduction from the s-t-path TSP to TSP used a constant-factor approximation algorithm for Φ-TSP. An asymmetric analogue of Φ-TSP should certainly generalize ATSP and its path version. For these two problems there are constant-factor approximation algorithms; see [STV18a, FS07], Chapter 4, and Chapter 5. However, we do not know how to generalize these algorithms to an analogue of Φ-TSP.

10.2 TSP and some generalizations

For the symmetric TSP the probably most important open question is to improve on Christofides' algorithm.

Open question 4
Is there an approximation algorithm for TSP with approximation ratio less than $\frac{3}{2}$?

This question is open even for node-weighted instances, i.e. instances with node-weights $c_v > 0$ for every vertex v and edge cost defined by $c(\{v, w\}) = c_v + c_w$. In contrast to ATSP, where node-weighted instances are essentially equivalent to unit-weight instances (see Section 4.6), for the symmetric TSP node-weighted instances seem to be more general. Although we know better algorithms than Christofides' algorithm for graph TSP (see [OSS11, MS16, Muc14, SV14] and Chapter 8), for node-weighted TSP this is not the case.

Another important question is whether the so-called "$\frac{4}{3}$-conjecture" is true, i.e. the integrality ratio of (TSP LP) is $\frac{4}{3}$. We know that the integrality ratio is at least $\frac{4}{3}$ (see Figure 6.1), but the best known upper bound is $\frac{3}{2}$ [Wol80]; see Theorem 6.1. For instances with up to twelve vertices Boyd and Elliott-Magwood [BEM07] verified the $\frac{4}{3}$-conjecture computationally.

Open question 5
Is the integrality ratio of (TSP LP) $\frac{4}{3}$?

This is open also for graph TSP, i.e. the special case $c \equiv 1$.

In Chapter 7 we gave an improved upper bound of 1.5284 on the integrality ratio of the relaxation (TSPP LP) for the s-t-path TSP. The best known lower bound, which is widely believed to be tight, is $\frac{3}{2}$; see Figure 6.2.

Open question 6
Is the integrality ratio of the relaxation (TSPP LP) for the s-t-path TSP $\frac{3}{2}$?

For the special case of unit weights this question was answered affirmatively by Sebő and Vygen [SV14]. We proved a strengthening of this in Chapter 8 (Lemma 8.39).

The proof by Sebő and Vygen [SV14] also extends to the T-tour problem in graphs, where we we are looking for a T-join J in an undirected graph $G = (V, E)$ such that (V, J) is connected and $|J|$ is minimum. The s-t-path graph TSP with $s \neq t$ is the special case where $T = \{s, t\}$ and graph TSP is the special case where $T = \emptyset$. For the T-tour problem in weighted graphs, no $\frac{3}{2}$-approximation algorithm is known.

Open question 7
Is there a $\frac{3}{2}$-approximation algorithm for the T-tour problem?

Our black-box reduction from s-t-path TSP to TSP also applies to the special case of the T-tour problem where $|T|$ is bounded by a constant; see Theorem 9.23. Also Zenklusen's [Zen19] recent $\frac{3}{2}$-approximation algorithm for the s-t-path TSP can be generalized to this special case of the T-tour problem. For general T, the best known approximation ratio is $\frac{8}{5}$, due to Sebő [Seb13].

Finally, one could consider Φ-TSP, which further generalizes the T-tour problem; see Chapter 9. We gave a 4-approximation algorithm, but a better approximation ratio might be possible.

Bibliography

[ABCC06] D.L. Applegate, R. Bixby, V. Chvátal, and W.J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006.

[AGM+17] A. Asadpour, M.X. Goemans, A. Mądry, S. Oveis Gharan, and A. Saberi. An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. *Operations Research*, 65:1043–1061, 2017.

[AKS15] H.-C. An, R. Kleinberg, and D.B. Shmoys. Improving Christofides' algorithm for the *s-t* path TSP. *Journal of the ACM*, 62(Article 34), 2015.

[AOG15] N. Anari and S. Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric TSP. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 20–39, 2015.

[Aro98] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45:753–782, 1998.

[BBCM04] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 166–174, 2004.

[BCK+07] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing*, 37:653–670, 2007.

[BEM05] S. Boyd and P. Elliott-Magwood. Computing the integrality gap of the asymmetric traveling salesman problem. *Electronic Notes in Discrete Mathematics*, 19:241–247, 2005.

[BEM07] S. Boyd and P. Elliott-Magwood. Structure of the extreme points of the subtour elimination polytope of the STSP. Technical Report TR-2007-09, SITE, University of Ottawa, 2007.

[BG13] Y. Bartal and L.-A. Gottlieb. A linear time approximation scheme for Euclidean TSP. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 698–706, 2013.

[Blä03] M. Bläser. A new approximation algorithm for the asymmetric TSP with triangle inequality. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 638–645, 2003.

[Bry73] T.H. Brylawski. Some properties of basic families of subsets. *Discrete Mathematics*, 6:333–341, 1973.

[BSvdSS14] S. Boyd, R. Sitters, S. van der Ster, and L. Stougie. The traveling salesman problem on cubic and subcubic graphs. *Mathematical Programming*, 144:227–245, 2014.

[CFG15] J. Cheriyan, Z. Friggstad, and Z. Gao. Approximating minimum-cost connected T-joins. *Algorithmica*, 72:126–147, 2015.

[CGK06] M. Charikar, M.X. Goemans, and H. Karloff. On the integrality ratio for the asymmetric traveling salesman problem. *Mathematics of Operations Research*, 31:245–252, 2006.

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the Travelling Salesman Problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.

[CKP12] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms*, 8(3):23:1–23:27, 2012.

[CLS15] J. Correa, O. Larré, and J.A. Soto. TSP tours in cubic graphs: beyond 4/3. *SIAM Journal on Discrete Mathematics*, 29:915–939, 2015.

[CSS01] J. Cheriyan, A. Sebő, and Z. Szigeti. Improving on the 1.5-approximation of a smallest 2-edge connected spanning subgraph. *SIAM Journal on Discrete Mathematics*, 14:170–180, 2001.

[DFJ54] G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

[DHK11] E.D. Demaine, M. Hajiaghayi, and K. Kawarabayashi. Contraction decomposition in H-minor free graphs and algorithmic applications. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 441 – 450, 2011.

[DHM10] E.D. Demaine, M. Hajiaghayi, and B. Mohar. Approximation algorithms via contraction decomposition. *Combinatorica*, 30:533–552, 2010.

[DKM17] Z. Dvořák, D. Král', and B. Mohar. Graphic TSP in cubic graphs. In *34th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 66, pages 27:1–27:13, 2017.

[DL18] B. Duník and R. Lukotka. Cubic TSP: A 1.3-approximation. *SIAM Journal on Discrete Mathematics*, 32(3):2094–2114, 2018.

[Edm67] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71:233–240, 1967.

[Edm68] J. Edmonds. Matroid partition. *Mathematics of the Decision Sciences, Part 1*, pages 335–345, 1968.

[Edm70] J. Edmonds. Submodular functions, matroids and certain polyhedra. In *Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969*, pages 69–87, 1970.

[EJ73] J. Edmonds and E.L. Johnson. Matching, Euler tours and the Chinese postman. *Mathematical Programming*, 5(1):88–124, 1973.

[FGM82] A.M. Frieze, G. Galbiati, and F. Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12:23–39, 1982.

[FGM16] Z. Friggstad, A. Gupta, and Singh. M. An improved integrality gap for asymmetric TSP paths. *Mathematics of Operations Research*, 41:745–757, 2016.

[Fra93] A. Frank. Conservative weightings and ear-decompositions of graphs. *Combinatorica*, 13:65–81, 1993.

[FS07] U. Feige and M. Singh. Improved approximation algorithms for traveling salesperson tours and paths in directed graphs. In *Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 104–118, 2007.

[FSS13] Z. Friggstad, M.R. Salavatipour, and Z. Svitkina. Asymmetric traveling salesman path and directed latency problems. *SIAM Journal on Computing*, 42:1596–1619, 2013.

[Gao13] Z. Gao. An LP-based 3/2–approximation algorithm for the *s-t* path graph traveling salesman problem. *Operations Research Letters*, 41:615–617, 2013.

[GB93] M.X. Goemans and D.J. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming*, 60:145–166, 1993.

[GGP+94] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, É. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 223–232, 1994.

[GLS81] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[GLS05] D. Gamarnik, M. Lewenstein, and M. Sviridenko. An improved upper bound for the TSP in cubic 3-edge-connected graphs. *Operations Research Letters*, 33:467–474, 2005.

[Got13] C. Gottschalk. Approximation algorithms for the traveling salesman problem in graphs and digraphs. Master's Thesis, Research Institute for Discrete Mathematics, University of Bonn, 2013.

[Gre73] C. Greene. A multiple exchange property for bases. *Proceedings of the American Mathematical Society*, 39:45–50, 1973.

[GV18] C. Gottschalk and J. Vygen. Better s-t-tours by Gao trees. *Mathematical Programming B*, 172:191–207, 2018.

[Hoo91] J.A. Hoogeveen. Analysis of Christofides' heuristic: some paths are more difficult than cycles. *Operations Research Letters*, 10(5):291–295, 1991.

[HV17] K. Heeger and J. Vygen. Two-connected spanning subgraphs with at most $\frac{10}{7}$ OPT edges. *SIAM Journal on Discrete Mathematics*, 31:1820–1835, 2017.

[Jai01] K. Jain. A factor 2 approximation algorithm for the generalized Steiner Network Problem. *Combinatorica*, 21:39–60, 2001.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103, 1972.

[Kar96] A.V. Karzanov. How to tidy up a symmetric set-system by use of uncrossing operations. *Theoretical Computer Science*, 157:215–225, 1996.

[Kha79] L. Khachiyan. A polynomial algorithm in linear programming [in Russian]. In *Doklady Academii Nauk SSSR*, volume 244, pages 1093–1096, 1979.

[Kle08] P.N. Klein. A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights. *SIAM Journal on Computing*, 37:1926–1952, 2008.

[KLS15] M. Karpinski, M. Lampis, and R. Schmied. New inapproximability bounds for TSP. *Journal of Computer and System Sciences*, 81:1665–1677, 2015.

[KLSS05] H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *Journal of the ACM*, 52:602–626, 2005.

[KTV19] A. Köhne, V. Traub, and J. Vygen. The asymmetric traveling salesman path LP has constant integrality ratio. In *Proceedings of 20th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 288–298, 2019. Full version: arXiv:1808.06542v2.

[KV18] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms.* Springer, Berlin, Sixth Edition, 2018.

[Lov76] L. Lovász. On some connectivity properties of Eulerian graphs. *Acta Mathematica Academiae Scientiarum Hungaricae*, 28:129–138, 1976.

[LWN17] H. Le and C. Wulff-Nilsen. Minor-free graphs have light spanners. In *Proceedings of the 58nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 767 – 778, 2017.

[Mad82] W. Mader. Konstruktion aller n-fach kantenzusammenhängender Digraphen. *European Journal of Combinatorics*, 3:63 – 67, 1982.

[Mit99] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: a simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28:1298–1309, 1999.

[MMP90] C.L. Monma, B.S. Munson, and W.R. Pulleyblank. Minimum-weight two-connected spanning networks. *Mathematical Programming*, 46:153–171, 1990.

[MS16] T. Mömke and O. Svensson. Removing and adding edges for the Traveling Salesman Problem. *Journal of the ACM*, 63(1):2:1–2:28, 2016.

[Muc14] M. Mucha. $\frac{13}{9}$-approximation for graphic TSP. *Theory of Computing Systems*, 55(4):640–657, 2014.

[NR07] V. Nagarajan and R. Ravi. Poly-logarithmic approximation algorithms for directed vehicle routing problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: APPROX/RANDOM 2007*, pages 257–270, 2007.

[NR08] V. Nagarajan and R. Ravi. The directed minimum latency problem. In *Proceedings of the 11th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 193–206, 2008.

[NW67] C.S.J.A. Nash-Williams. An application of matroids to graph theory. In *Theory of Graphs; Proceedings of an International Symposium in Rome 1966*, pages 263–265, 1967.

[NZ19] M. Nägele and R. Zenklusen. A new dynamic programming approach for spanning trees with chain constraints and beyond. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1550–1569, 2019.

[OSS11] S. Oveis Gharan, A. Saberi, and M. Singh. A randomized rounding approach to the Traveling Salesman Problem. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2011.

[RS93] S.B. Rao and W.D. Smith. Approximating geometric graphs via "spanners" and "banyans". In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 540–550, 1993.

[Sch03] A. Schrijver. *Combinatorial Optimization, Polyhedra and Efficiency.* Springer, 2003.

[Seb97] A. Sebő. Potentials in undirected graphs and planar multiflows. *SIAM Journal on Computing*, 26(2):582–603, 1997.

[Seb13] A. Sebő. Eight-fifth approximation for the Path TSP. In *Proceedings of 16th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 263–373, 2013.

[Ser78] A. I. Serdjukov. Some extremal bypasses in graphs [in Russian]. *Upravlyaemye Sistemy*, 17:76–79, 1978.

[SSTvZ18] F. Schalekamp, A. Sebö, V. Traub, and A. van Zuylen. Layers and matroids for the traveling salesman's paths. *Operations Research Letters*, 46(1):60–63, 2018.

[STV18a] O. Svensson, J. Tarnawski, and L. Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 204–213, 2018.

[STV18b] O. Svensson, J. Tarnawski, and L. Végh. Constant factor approximation for ATSP with two edge weights. *Mathematical Programming*, 172(1-2):371–397, 2018.

[STV19] O. Svensson, J. Tarnawski, and L. Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. arXiv:1708.04215v3, 2019.

[SV14] A. Sebő and J. 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] O. Svensson. Approximating ATSP by relaxing connectivity. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–19, 2015.

[SvZ19] A. Sebő and A. van Zuylen. The salesman's improved paths through forests. *Journal of the ACM*, 66(4):28:1–28:16, 2019.

[SW90] D. B. Shmoys and D. P. Williamson. Analyzing the Held-Karp TSP bound: a monotonicity property with application. *Information Processing Letters*, 35:281–285, 1990.

[Tra17] V. Traub. Approximating the s-t-path TSP. Master's Thesis, Research Institute for Discrete Mathematics, University of Bonn, 2017.

[TV18] V. Traub and J. Vygen. Beating the integrality ratio for s-t-tours in graphs. In *Proceedings of 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 766–777, 2018. Full version: arXiv:1804.03112v2.

[TV19a] V. Traub and J. Vygen. Approaching 3/2 for the s-t-path TSP. *Journal of the ACM*, 66(14), 2019.

[TV19b] V. Traub and J. Vygen. An improved upper bound on the integrality ratio for the s-t-path TSP. *Operations Research Letters*, 47:225–228, 2019.

[TVZ19] V. Traub, J. Vygen, and R. Zenklusen. Reducing Path TSP to TSP. arXiv:1907.10376v1, 2019.

[Vyg16] J. Vygen. Reassembling trees for the traveling salesman. *SIAM Journal on Discrete Mathematics*, 30(2):875–894, 2016.

[Whi32] H. Whitney. Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34:339–362, 1932.

[Wol80] L.A. Wolsey. Heuristic analysis, linear programming and branch and bound. *Mathematical Programming Study*, 13:121–134, 1980.

[Woo74] D.R. Woodall. An exchange theorem for bases of matroids. *Journal of Combinatorial Theory, Series B*, pages 227–228, 1974.

[XR15] Z. Xu and B Rodrigues. A 3/2-approximation algorithm for the multiple TSP with a fixed number of depots. *INFORMS Journal on Computing*, 27(4):636–645, 2015.

[Zen19] R. Zenklusen. A 1.5-approximation for Path TSP. In *Proceedings of 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1539–1549, 2019.

Summary

The traveling salesman problem is the probably most famous problem in combinatorial optimization. Given a graph G and nonnegative edge costs, we want to find a closed walk in G that visits every vertex at least once and has minimum cost. When G is an undirected graph, we call this problem the symmetric traveling salesman problem (TSP). When G is a directed graph, we call it the asymmetric traveling salesman problem (ATSP).

We also consider the more general path versions, where we do not require the walk to be closed, but to start and end in prescribed vertices s and t. Moreover, we investigate the unit-weight special cases, where all edges have the same cost. These are of particular interest because the best known lower bounds on the integrality ratio of the standard linear programming relaxations can be obtained from unit-weight instances.

In this thesis we give improved approximation algorithms and better upper bounds on the integrality ratio of the classical linear programming relaxations for several of these traveling salesman problems. For this we use techniques arising from various parts of combinatorial optimization such as linear programming, network flows, ear-decompositions, matroids, and T-joins.

In a recent breakthrough, Svensson, Tarnawski, and Végh gave the first constant-factor approximation algorithm for ATSP. Building on their result, we present a simpler algorithm with a much better approximation guarantee. We prove an approximation ratio of $22 + \varepsilon$ (for any fixed $\varepsilon > 0$), improving on the previously best bound of 506. We also improve the upper bound on the integrality ratio of the classical LP relaxation from 319 to 22.

For the path version of ATSP we give the first constant upper bound on the integrality ratio of its classical LP relaxation. We show that the integrality ratio for the path version is at most four times larger than for ATSP.

Then we turn to the symmetric case. For the path version of TSP we prove a new upper bound on the integrality ratio of its classical LP relaxation. We achieve this by an improved analysis of an algorithm by Sebő and van Zuylen.

For the s-t-path TSP with unit weights, the integrality ratio is known to be exactly $\frac{3}{2}$. We show that the classical instances with integrality ratio close to $\frac{3}{2}$ are essentially the only such instances (up to small local differences). From the proof of this result we derive the first approximation algorithm that has an approximation ratio below the integrality ratio of the classical LP relaxation.

Finally, we prove that the s-t-path TSP is not substantially harder to approximate than its special case TSP: if there exists an α-approximation algorithm for TSP, then for any fixed $\varepsilon > 0$ there is an $(\alpha + \varepsilon)$-approximation algorithm for the s-t-path TSP.

Until very recently the best known approximation ratios for TSP and s-t-path TSP differed significantly. Our result avoids such discrepancies in the future.

Moreover, our result also holds for the case of unit weights. By applying this to an algorithm by Sebő and Vygen, we obtain an improved approximation algorithm for the s-t-path TSP with unit weights.